2017 年臺灣國際科學展覽會
優勝作品專輯

作品編號　**190005**

參展科別　電腦科學與資訊工程科

作品名稱　英文篇章難易度自動分級之研究

得獎獎項　大會獎：三等獎

就讀學校　臺北市立第一女子高級中學

指導教師　黃芳蘭、陳信希

作者姓名　許湘鈴、任恬儀

關鍵字　語法分析、程度分級、語言學習

作者簡介

許湘鈴(左)：喜歡專研、喜歡科展、喜歡公假，也喜歡音樂、社團和朋友，珍惜身邊的一切。我可能是個有點瘋狂的人。找到喜歡的事情，就會一不小心廢寢忘食，通霄趕工不眠不休。

任恬儀(右)：從小我就是一個喜歡數學的女孩，也對電腦的各種功能很有興趣，高一專研選組的時候，我選擇了資訊組，一開始我最先接觸的語言是 c ，之後為了專題研究所需，也學習了 python。

摘要

以製作適合高中生的英文篇章難易度自動分級為初衷，本研究採高中英文課文為語料，針對「如何分級」，意即從文章萃取哪些特徵、利用何工具或語料協助萃取特徵、以何工具分級等因素，進行研究與實驗，並建立一套新方法。首先進行前處理，再嘗試以單字、句型的數量或比例、句長、音節長、整合以上分析等各式特徵，支持向量機(Support Vector Machines)、隨機森林分類器(Random Forest Classifier)、決策樹分類器(Decision Tree Classifier)、卷積神經網路句分類器(Convolutional Neural Networks for Sentence Classification)等工具，進行將篇章分為高中一、二、三年級等三個難易度等級的測試，建立自動分級模型。最後製作成可供大眾使用的自動分級網頁。各項測試之中，最佳分類效能為整合各項特徵時得到的分類正確率65.04%，經模擬得知，此效能較過去研究，已有所提升。

Abstract

To help students whose native languages are not English enhance English reading ability, we take passages in English textbooks to do a research on making an automatic classification of English passages by the degrees of difficulty and write a website for people to make use of. We respectively extract and test a series of features from the preprocessed passages, including Surface Feature, Word Feature, Sentence Pattern Feature, and the integration of the former ones. Testing the features by using them to classify the passages with Support Vector Machine, Random Forest Classifier, Decision Tree Classifier, and Convolutional Neural Networks for Sentence Classification,we get the best classification effectiveness with the integrated features and Supported Vector Machine, with the accuracy rate up to 65.04%, which is higher than several previous researches.

一、 前言

(一) 研究動機

處於這個國際交流頻繁的時代，英文成了不可或缺的能力。然而，臺灣學生的英語程度落差頗大。經過研究後，我們瞭解到教材難易度對學習者具有非常大的影響力[4]，學習者閱讀的文章難易度是否適當，可能成為其學習是否順利的關鍵。市面上雖有許多以難易度分級的英語學習雜誌，但雜誌中的文章有限，還必須付出一定的金錢購買；而同樣有分級的坊間參考書，卻因為沒有經過任何正式的檢核程序，經常可以發現許多錯誤；在網際網路普及的現今，網路上的英文文章雖然比比皆是，其難易度卻十分駁雜。因此，我們想設計出一套難易度的判別方式，輔助英文學習者自主挑選適合自己程度的文章閱讀。

從最切身的角度來看，學校英文老師為了鼓勵我們多閱讀原文書，自行選書撰寫閱讀心得成為一項例行作業。其中，選書卻成為一個頗為困難的問題。許多同學們都曾受無法事先辨別文本難易度所困擾，閱讀過的書不是太過簡單，便是充斥古字、稀有字，恐怕都不是培養英語閱讀之興趣與習慣的最佳方法。而這也成為了我們希望能夠以電腦事先判別英文篇章難易度的動機之一。

綜合以上種種，我們想設計一套能夠「輔助學習者篩選出難易度合適的文章來閱讀」的判別方式和系統，並且撰寫成網頁以提供大眾直接利用。

(二) 研究目的

製作判定高中英文篇章難易度的的機器學習模型：

1. 由單字判定篇章難易度
2. 由句型判定篇章難易度
3. 由表面特徵判定篇章難易度
4. 整合單字、句型、表面特徵，判定篇章難易度

並撰寫可供大眾利用的英文篇章難易度自動分級網頁。

二、 研究過程與方法

以下首先探討相關研究，介紹研究設備及器材，接著敘述前處理、特徵萃取、機器學習、網頁撰寫等研究過程。

(一) 相關研究探討

閱讀是讀者與文本互動的過程，意即讀者、文本、讀者與文本的關係三項因素，都會影響閱讀過程的順利與否。其中，讀者因素因人而異（如：語文程度、如何閱讀等等），文本的難易度則是既有的，能夠掌握的因素便是讀者與文本的關係──意即，根據難易度分類文章，讓讀者變能依自身程度與情況，自行選擇文章閱讀。

過去有許多研究曾經探討如何分析文章的適讀性（readability），也已經發展出許多適讀性公式，最廣為使用的如 1968 年的富萊適讀性指數（Fry Readability Graph），考量了句子的密度和單詞音節數等因素[4]。國內也有學者嘗試以各式適讀性公式，判別英文讀本的適讀性級別[5]。

然而 2010 年，黃昭憲等認為早期的研究（適讀性公式）基於當時技術與資源的限制，大多只考慮文字資料中的詞彙難易度、句子數目和長句資訊，不但不能滿足實際需求，連學理上都有可議之處。例如：適讀性公式沒有考量是否以英文做為母語，因此，對於國內學生來說，適讀性分級便不等於難易度分級，雖並非無可取之處，但也不貼近需求。

而近年來國內對於英文文章難易度分類的研究[3,4,6]，雖考量了更深入的資訊，如子句結構等，但多主要採用全民英檢（GEPT）進行難易度分級，且僅取全民英檢的初、中、中高三個級別，進行相當粗略的分級。對於英語學習者來說，由於難易度級別差距相當大，恐怕也不敷使用。

因此，無論採用國外的分級制或是全民英檢的分級，或許都不是太實用。於是，針對此問題，我們直接以 103 學年度國內各版本的高級中學英文課文作為語料。而相較於楊子儀、黃孝慈採用的 C5.0 決策樹(Decision Tree)分類器、許珀豪採用的 k-最鄰近演算法（KNN）分類技術，我們應用新的分類工具，期能達到更好的分類效果：

1. 支持向量機（Support Vector Classification, SVM）[2]

SVM 是一個實用的資料分類技術。先將資料分成訓練資料和測試資料，訓練資料中的，每個例子必須包含一個「目標值(target value)」和數個「屬性(attributes)」，而 SVM 的目標就是基於這些訓練資料，建立一個給定目標值便能預測屬性的模型。

本實驗使用 SVM 的徑向基函數核(Radial basis function kernel, RBF)，也是支持向量機最常使用的核函數，它可以非線性地將樣本映射到更高維的空間，因此能夠處理目標值和屬性之間並非線性關係的狀況。RBF 有 C 和 γ 兩個參數，實驗前無法預知參數數值，必須進行一系列的測試以找到能夠達到最佳分類效果的參數。其中，C 是誤差項的懲罰參數(penalty parameter)，預設為 1.0，γ 則是核係數(kernel coefficient)，預設為自動(auto)。

SVM 之公式如下[2]：

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned}$$

其中，(xi, yi)表示（資料, 標籤）；$K(x_i, x_j) \equiv \varphi(x_i)^T \varphi(x_j)$即核函數。

2. 卷積神經網路句分類器（Convolutional Neural Networks for Sentence Classification, CNN sentence）[1]

近來，深度學習的模型在電腦視覺及語音識別都已經達到了顯著的效果。隨著自然語言的發展，大部分以深度學習完成的工作，都包含了透過神經語言模型學習字詞向量(word vectors)表徵的過程，也發表了和學習字詞向量分類相關的文章。

CNN 模型原先是為了計算機視覺領域而發明，後來被證明是有效的自然語言處理工具，並已在語意分析的搜索查詢系統、句子建模，以及其他的傳統自然語言任務得到了優異的結果。因此，我們將使用此新穎的工具，並測試其是否也能為文章難易度的判斷實驗帶來更好的效果。

我們整理了過去類似難易度分類文章的研究。由於研究多是基於過往之成果，故在此以「新特徵」表示該研究新使用的特徵。

研究	語料／新特徵／方法	探討
適讀性公式早期國外研究、[5]	語料：英文讀本等	1. 學理上有可議之處[4] 2. 未考量是否以英文為母語
	新特徵：文長、句數等	
	方法：適讀性公式、Microsoft Word[5]	
全民英檢(GEPT)[3,4,6]	語料：全民英檢(GEPT)	僅概分為初、中、中高級，無法對應國內學生實況。
	新特徵：子句結構等	
	方法：C5.0 決策樹、KNN 算法	
我們的構想	**語料：高中英文課文** **新特徵：單字向量、句型結構、單字頻率等** **方法：支持向量機(SVM)、卷積神經網路(CNN)**	

表一：相關研究之探討與整理

(二) 研究設備及器材

1. 硬體

(1) 筆記型電腦（寫程式。CPU: Intel Core i5-4210U with Turbo Boost up to 2.7 GHz）

(2) 工作站電腦(執行程式。CPU: Intel(R) Core(TM) i7-2600 CPU @3.40GHz 四核心）

2. 軟體及工具

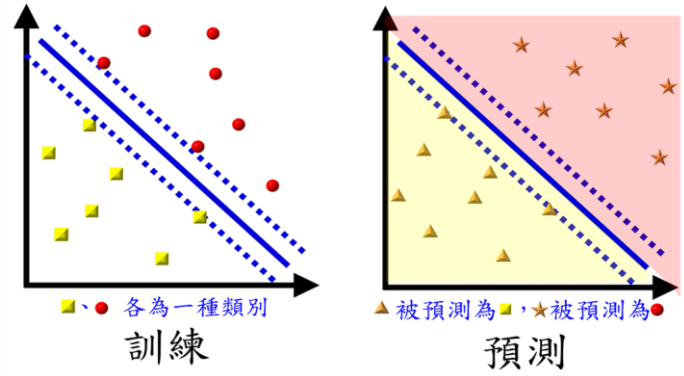
(1) Python（程式語言）

(2) Django（Web 應用框架）

(3) Stanford Parser（句型剖析工具）

(4) Natural Language Toolkit （NLTK，自然語言工具）

(5) 支持向量機（Support Vector Machine, SVM）[2]（**監督式學習**領域工具）



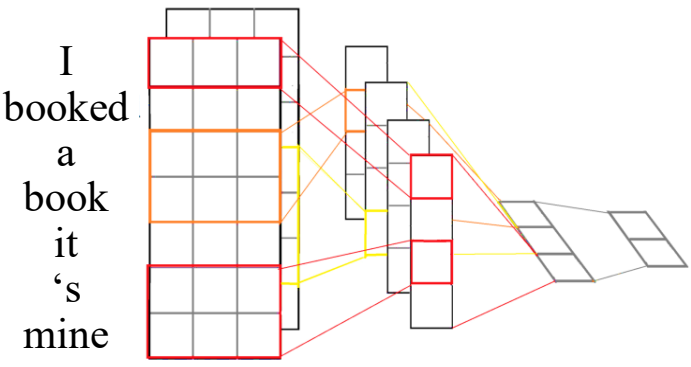
圖一、SVM 原理示意圖

建構高維「超平面」分類資料點。如圖，以二維示意 SVM：得到特徵，以機器學習在類別間劃界線（圖中藍實線）、建立模型，用以預測類別。SVM 有 C、γ 兩影響分類效果之參數，須逐個嘗試 2 的次方以求最佳參數組合。

(6) 隨機森林分類器（Random Forest Classifier, RFC）（**監督式學習**領域工具）

(7) 決策樹分類器（Decision Tree Classifier, DTC）（**監督式學習**領域工具）

(8) 卷積神經網路句分類器（Convolutional Neural Networks for Sentence Classification, CNN sentence）[1]（**深度學習**領域工具）



圖二、CNN 原理示意圖

卷積神經網路(CNN)基於試圖使用多個處理層，對數據進行高維抽象。本圖以 I booked a book. It’s mine.為例。CNN sentence 利用此概念，使用 word2vec 的向量表，以 300 個維度表示每個向量，判斷句意。技術較新穎。

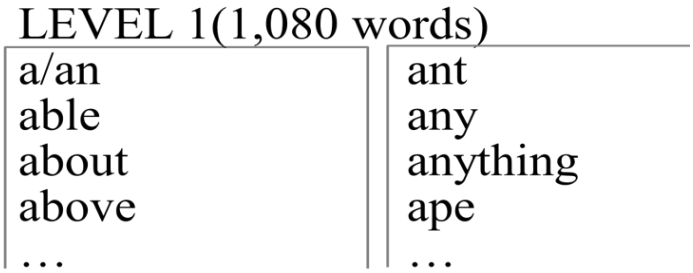
3. 語料：

(1) 美國國家語料庫（American National Corpus, **ANC**）

一個美式英語的大型電子彙集庫，對於任何用途完全公開而無限制。本實驗使用書面（Written）部分的頻率資料（Frequency Data），以在 ANC 語料庫中出現過的次數（Count，以下簡稱計數）排序前 24,521 行的資料。

圖三、ANC 節錄
如圖，每行有四項資料（綠字），包括詞和其原形（如橘框），詞性的代號，以及計數。採依計數排序（籃框）的版本。

(2) 高中英文詞彙參考表（The English Reference Word List for Senior High School, **ERWL**）



圖四、ERWL 節錄
將高中英文詞彙分為六個等級，每個等級分別包含 1080 個詞彙，並依字典序排序。

(3) 各版本 103 學年度高中英文課本電子檔

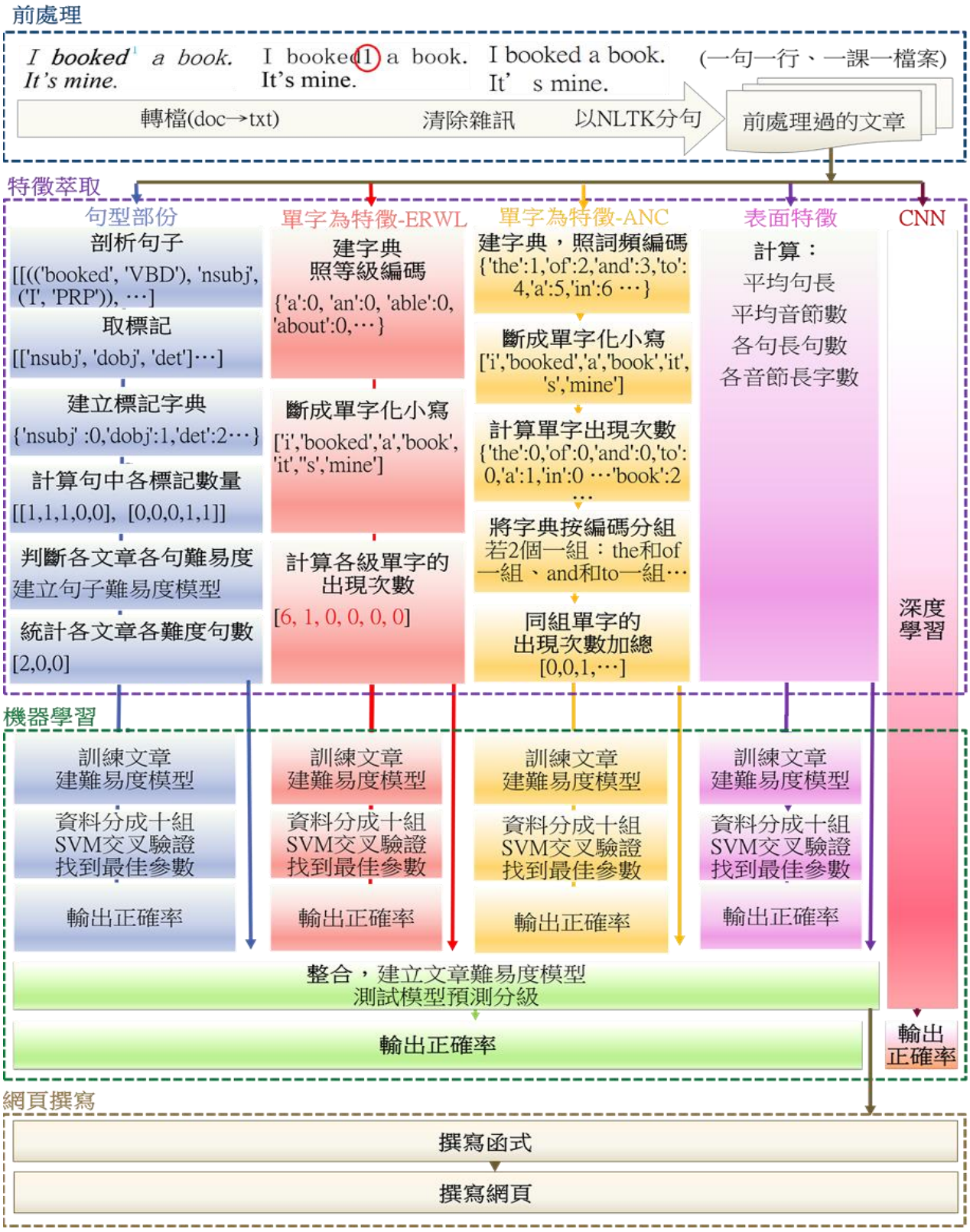
	三民	龍騰	南一	遠東 施	遠東 陳	總和
高一	24	24	24	24	23	119
高二	24	24	24	30	24	126
高三	22	20	23	26	20	111
總和	70	68	71	80	67	**356**

表二、課文來源分布（單位：篇）
採三民乙版，每冊比甲版多兩課，其他課文內容無異；遠東施版係由施玉惠、林茂松、黃崇術、Sarah Brooks 編著；遠東陳版則由陳純音主編。

(三) 實驗整體流程

本研究的整體流程可以概分為前處理、特徵萃取、機器學習、網頁撰寫四個部分。

下圖為本研究的實驗整體流程圖，以”I booked a book. It’s mine.”為例。



圖五、實驗整體流程圖

在實驗的整體流程中，前處理為其他部分之基礎；特徵萃取部分，我們發想、嘗試並探討以各式新、舊特徵，以作為自動分級之材料；機器學習部分，我們嘗試了包括監督式學習和深度學習兩個領域的數種方法：SVM、RFC、DTC 等分類工具；深度學習則測試 CNN sentence。由於 CNN sentence 性質特殊，和前述實驗分開進行；最後完成整合模型，並撰寫成方便大眾使用的網頁。

(四) 前處理

1. 撰寫程式，處理各版本高中英文課本電子檔(doc)

１. 將**課文**部份取出，儲存為純文字文件(txt)

２. 去除上、下標（部分課文中有標註生字的下標編號，轉成 txt 時會變成普通數字，影響判讀）、亂碼（doc 排版符號等）等雜訊

３. 使用 NLTK 中的 Punkt Sentence Tokenizer 將課文分句，儲存（一個純文字文件放一課、一行放一句）

2. 撰寫程式，處理欲使用之**字彙庫**

１. 整理字彙內容（如：將”a (n)”分成”a”和”an”兩字）

２. 去除雜訊（避免編碼問題，去除含非英文語言字母之單字）

３. 改寫成程式容易讀取之格式（如：加入跳格、換行符號）

(五) 特徵萃取

相較於過去研究使用過的各式特徵，我們企圖**萃取出更簡單卻更有效的特徵**，因此，我們試圖從生活經驗發想，並使用更具效益的語料、資料、工具等實作，得到，如單字、句型之相關特徵；此外，我們亦基於舊有想法，發想出新方法，如表面特徵的部分。

中學階段課文之學習重點在於單字、句型，應是辨別難易度的重要特徵，因此我們構思了針對單字、句型，及較淺顯的各式表面特徵，進行量化的一系列方式，以萃取特徵、進行機器學習。

為了應用於處理長文（如：長篇小說等），避免文長影響模型判別其難易度，**我們對於以下「計算出現次數」的實驗，進行了「計算出現頻率」的版本**，意即每個數據都除以該數據總和。這也是我們首創的想法之一。

以下為了方便舉例，我們對所有的例子做一個簡單的假設：

假設”My dog also likes eating sausage.”是一篇高一課文。

1. 單字特徵

(1)字彙集來源

若欲以各個單字出現過的次數為特徵，可先取得一個字彙集，以字彙集建立字典（dic）。我們測試了兩個字彙集。較容易取得、屬於完全公開資源的字彙集是**美國國家語料庫（ANC）**。ANC 有以詞頻排序過的資料，並且包含單字變化形-原形-詞性-詞頻的對應關係，但以「英語為母語者產出之書面資料」的詞頻排序，和學生最熟悉的英文課文略有出入（如表三）。我國大學入學考試中心公布之**《高中英文詞彙參考表》（ERWL）**的分級可能相對貼近我國學生，但 ERWL 僅分將單字分為六級。

表三、ANC 與課文之詞頻序比較節錄

我們計算英文課文的詞頻序高低，和 ANC 略有出入。

單字	ANC	課文
of	2	3
and	3	4
to	4	2
is	7	8
for	8	10

(2)**轉換成原形**

過去研究[3,4,6]在處理單字時，多會先轉換成原形。但由於許多單字是以複數形、過去式等形態出現在文章中，可能造成影響。因此，我們嘗試兩種方式，一是直接以單字的各式變化形為索引，計算在課文中出現的次數；二是利用 ANC 中的單字－原形詞對應關係，將文章中出現的詞全部轉換成原形詞，再行處理。

(3)依詞頻排序**分組、加總**出現次數

一篇課文僅數百字，但建立能夠容下所有課文中之多數單字的字典，其單字數量達上萬，而將數百字的課文，分散進數萬個元素的特徵矩陣中，數據相當稀疏。若詞頻高低和該詞的難易度具有一定程度的關聯性，應可將單字以詞頻排序過後，每一定數量的單字的出現次數加總，放進一個新的矩陣，再進行機器學習。至於究竟是否分組、加總，或每多少個字詞加總一次，即探討之處所在。

(4)**計次數或頻率**

我們也試了改計詞頻的版本，將所有「字的出現次數」數據除以總字數。

2. 單字特徵、ANC 為語料之步驟

(1) 從課文讀出單字。

> 斷字處理，得['my', 'dog', 'also', **'likes'**, 'eating', 'sausage']

(2) （**轉換成原形**才須進行）用 ANC 的「變化形-原形」對應關係，建「變化形-原形」之對應字典，將課文中讀出的單字都轉成原形。

> {'the': 'the', 'of': 'of', 'a': 'a',…, **'likes': 'like'**,…}（表示 likes 的原形是 like）
>
> 轉換得['my', 'dog', 'also', **'like'**, 'eat', 'sausage']

(3) 建立單字和 ANC 詞頻序對應的字典，以及和其字典等長之零矩陣。計算**單字在課文中出現的次數（即「計數」）**於詞頻序對應之矩陣位置，完成其新計數矩陣。

> {'the':0, 'of':1, 'a':2,…, **'like':**,…}（表示 likes 的詞頻序 62）
>
> [0, 0, 0, …, **1,**…] （表示詞頻序 62 的單字 **like 總共出現 1 次**）

(4) （**分組、加總**才須進行）將單字按照 ANC 計數排序後分組，再利用前一步完成之新計數矩陣，加總各組的新計數，成為一個新的、濃縮過的矩陣。

> 若 500 個單字一組。第 0~499 個單字的**出現次數總和為 3**、第 500~999 個為 1、第 1000~1499 個為 0…，則得特徵矩陣[**3**, 1, 0, …]

(5) （**頻率版本**才須進行）將前一步驟所得矩陣除以總字數。

> 原矩陣[3, 1, 0, …]除以總字數 6 字，得新矩陣[**0.5**, 0.1667, 0, …]

(6) 得 x、y 矩陣，放進分類器訓練、測試。

> 特徵矩陣[**0.5**, 0.1667, 0, …]屬高一難易度，記為[1]，則：
>
> **輸入資料**：x 矩陣（特徵）為 **[0.5, 0.1667, 0, …]**，y 矩陣（難易度）為 **[1]**

3. 單字特徵、ERWL 為語料之步驟

(1) 以 ERWL 製作「單字-難度等級」對應字典。

> {'a':1, 'an':1, 'able':1, 'about':1, 'above':1,…, **'like':1**,…}（這些字皆屬**難度 1**）

(2) 從課文讀出單字、建立字典並計算數量。和 ANC 為語料之步驟類似，但改計「各級單字出現的次數」。詳見單字特徵、ANC 為語料。

> [**6**,0,0,0,0,0]（**難度 1** 的字記在第 1 格，共出現 **6** 次）

(3) （**頻率版本**才須進行）將前一步驟所得矩陣除以總字數。

> 原矩陣[**6**,0,0,0,0,0]除以總字數 6 字，得新矩陣[**1**,0,0,0,0,0]

(4) 得 x、y 矩陣，放進分類器訓練、測試。

> 特徵矩陣[**1**,0,0,0,0,0]屬高一難易度，記為[1]，則：
>
> **輸入資料**：x 矩陣（特徵）為 **[1,0,0,0,0,0]**，y 矩陣（難易度）為 **[1]**

4. 句型特徵

句型特徵我們採用詞之間的**關係（Dependency）資料**，也就是各詞之間的關係。亦是一**首創**作法。

(1) 以句為單位

以句型為特徵時，必須將資料斷句，以句子為單位剖析。對每篇課文，我們對當中的各個句子剖析，得到第一層特徵──關係資料。各個句子依其「各式關係出現過的次數」為特徵，以分類器進行第一層分類，分成三個難易度。此次分類，得到以句子為單位的分級結果。

(2) 以篇為單位

再以各篇課文「第一層分類時，被分成各個難易度的句子分別有幾句」為特徵，以分類器進行第二層分類，得到以篇章為單位的分級結果。

5. 句型特徵──以句為單位之步驟

(1) 將標籤編碼，製成字典。

> {**"advmod": 0**, "advcl": 1, "mark": 2, "nsubj": 3, …}（表示 **adbmod** 編碼 0）

(2) 使用 Stanford Parser 剖析每一個句子，得到其中的關係資料。

> (('likes', 'VBZ'), '**nsubj**', ('dog', 'NN'))
>
> (('dog', 'NN'), '**nmod:poss**', ('My', 'PRP$'))
>
> (('likes', 'VBZ'), '**advmod**', ('also', 'RB')) ……

(3) 取這之中表示兩者關係的**關係標記**，組成句型矩陣，得到**句子架構**。

> ['nsubj', 'nmod:poss', '**advmod**',……]

(4) 統計每個句子**各標記出現的次數**，儲存於矩陣中該標籤的字典編碼位置。

[**1**,0,0,1,1,0,….]（表示編碼 0 的 **adbmod** 共出現 **1 次**）

(5) 得 x、y 矩陣，分別在 SVM、RFC 、DTC 訓練、測試。

特徵矩陣[**1**,0,0,1,1,0,….]屬高一難易度，記為[1]，則：

輸入資料：x 矩陣（特徵）為 **[1,0,0,1,1,0,….]**，y 矩陣（難易度）為 **[1]**

6. 句型特徵──以篇為單位之步驟

(1) 分別讀出每篇文章、計算文章中分屬三個年級難易度的句子各有幾句，作為特徵。

[1,0,0]（表示有 **1** 個句子被判成了**記在第 1 格的高一難度**）

(2) （**頻率版本**才須進行）將前一步驟所得矩陣除以總句數。

原矩陣[**1**,0,0]除以總句數 1 句，得新矩陣[**1**,0,0]

(3) 各文章輪流當作測試資料，其他文章當作訓練資料，計算正確率。

特徵矩陣[**1**,0,0]屬高一難易度，記為[1]，則：

輸入資料：x 矩陣（特徵）為 **[1,0,0]**，y 矩陣（難易度）為 **[1]**

7. 表面特徵

表面特徵當中，為了實用考量，我們依舊秉持著「**避免受文長影響**」的原則。因此單純的總字數、總句數等會被文長影響的特徵，我們皆不採用。不受影響的特徵如平均句長、平均音節數被過去諸多研究不斷使用[5]，故我們將不多加探討。而我們的**各句長句數**、**各音節長字數**兩特徵雖不複雜，過去研究中卻未見到。

(1) 平均句長

$$平均句長 = \frac{文章總字數}{文章總句數}$$

(2) 平均音節數

$$字平均音節數 = \frac{文章總音節數}{文章總字數}$$

$$句平均音節數 = \frac{文章總音節數}{文章總句數}$$

(3) 各句長句數

$$i\ 句長句數 = 文中句長\ i\ 的句子數\ \ \forall 1 \le i \le 40, i \in N$$

但考量短文句數不多，資料較分散，將 n 個句長的句子數加總，達到類似平滑的效果，形成新的特徵矩陣。

$$i\ 句長句數特徵 = \sum_{k=i}^{i+n} 文中句長\ k\ 的句子數\ \ \forall 1 \le i \le 40 - n, i \in N$$

(4) 各音節長字數

$$i\ 音節長字數 = 文中音節長\ i\ 的字數\ \ \forall 1 \le i \le 10, i \in N$$

(六) 機器學習

本實驗以 **10 次交叉驗證**（10-fold cross validation）方式，將前特徵萃取得到之 x 矩陣（特徵）及 y 矩陣（難易度）輸入分類器訓練和測試。主要以 SVM 分級，其有 C、γ 兩參數，須從 $2^{-5}, 2^{-4}, 2^{-3},\ldots,2^{5}$ 開始，逐步縮小範圍暴力嘗試，尋找能夠得到最佳效果的參數組合。以 SVM 分析各項特徵之效能，並測試 RFC 和 DTC 分類工具。

(七) CNN sentence

由於該工具使用的資料為句子，所以和其他實驗分開介紹。

1. 測試資料

(1) 工具內部會有自動分成訓練與測試資料，因此只將資料分成一、二、三年級。

(2) 因考量來自同一篇文章的句子可能會有相近的文意，故將各句子依文章分類。

2. 程式流程

(1) process_data.py

- 功能：以 Google Word2Vec 及測試資料的每一個句子，建立名為 mr.p 的 pickle 檔。
- 修改部份：設定路徑將原本的「反向句子設定為編號 0，正向句子設定為編號 1」，改成「一、二、三年級分別設定為 0、1、2」。原為第幾份資料的 0~9 隨機數字，改至每篇文章隨機一次，確保文章不會同時出現在訓練資料和測試資料中。

(2) Conv_sentence.py

- 功能：讀取 mr.p 的資料，及模式的引數，整理出形式，及依 Word2Vec 整理出多維向量，引用 Theano 模組，進行深度學習。
- 修改部分：max_len 為一個句子的最大長度，init 為分類的數量，由正面負面兩項，改為一、二、三年級三項。

(八) **整合**

整合前，我們將程式**重新撰寫**，力求**模型未來的維護與擴增更加容易**。如：將前處理程式碼嵌進主程式、所有特徵萃取的程式重寫成函式等等。模型流程簡述如下：

1. 輸入分句、經過前處理的文章。
2. 分析**單字特徵**：ANC 各個單字、ERWL 各級單字出現次數。
3. 分析**句型特徵**：被判成一、二、三年級難度的句子個數。
4. 分析**表面特徵**。
5. 將分析得到之特徵矩陣相接，以先前訓練出之 SVM 模型預測分級並輸出。

(九) 網頁撰寫

為了**應用於長文的判別**，我們亦建置了不會被文長影響的**頻率版本之整合模型**，並以 Django **撰寫網頁**，**直接應用**此模型。

三、 研究結果與討論

以下將先闡述本研究之效能評估方式；再聚焦於本研究兩大重要過程：特徵萃取、機器學習，討論實驗結果。最後呈現各項實驗與研究之整理與比較，進行綜合性地討論。

(一) 效能評估方式

下表為一般判斷時的情況。其中，「人工判斷」相當於我們現有的「課文實際上是高一、二、三」，而「系統判斷」則相當於「分類器預測為高一、二、三」。

若以「從高中課文中判斷高一課文」為例，「相關」相當於「被認為是高一課文」，「不相關」相當於「被認為不是高一課文」。因此有了判斷「正確」、判斷「錯誤」，以及「正例」、「負例」之分。

		系統判斷	
		相關	不相關
人工判斷	相關	正確正例（true positive, TP）	錯誤負例（false negative, FN）
	不相關	錯誤正例（false positive, FP）	正確負例（true negative, TN）

表四、系統判斷情形的種類

1. 準確率（Precision）

$$準確率(\text{Precision}) = \frac{正確正例(\text{TP})}{正確正例(\text{TP}) + 錯誤正例(\text{FP})}$$

2. 召回率（Recall）

$$召回率(\text{Recall}) = \frac{正確正例(\text{TP})}{正確正例(\text{TP}) + 錯誤負例(\text{FN})}$$

3. **F 度量（F-measure）**（為方便闡釋，後悉以「**正確率**」稱之。）

$$\text{F 度量} = \frac{2 * 準確率 * 召回率}{準確率 + 召回率}$$

(二) 特徵萃取

特徵萃取部分，包含單字特徵、句型特徵、表面特徵三大主軸，三大主軸以下又各有數種變因、不同特徵。以下將針對不同之特徵萃取結果進行討論與闡述。

1. 單字特徵

單字特徵的變因包含字彙集來源、計次數或頻率、是否轉換成原形、是否分組加總等等。由於變因繁多，為了避免繁雜，暫以「**標準過程**」稱此特徵一系列實驗中**得到最高正確率之過程**，再就不同的變因比較差異。若未特別註明，變因即與該過程相同。

單字特徵最高正確率 64.43%出現於：ANC 為語料、計次數、不轉換成原形、不分組加總。係以這些條件作為標準過程。

(1) 字彙集來源

若直接以 ERWL 最高正確率 50.84%和 ANC 之最高正確率 64.43%做比較，可知 ANC 之**最佳分類效能**顯著高於 ERWL。應是由於 ERWL 中僅收錄 6 個級別，因此僅有 6 個特徵值，僅能粗略評估單字難易程度；而本實驗使用 ANC 前 24521 筆資料，因此特徵值高達 24521 個，將各個單字分別統計，有助於較細微之判別。**特徵值數量差異**懸殊，故分類效能差異亦相對顯著。

字彙集來源	ANC	ERWL
最高正確率(%)	64.43	50.84

表五、使用不同字彙集來源之正確率(%)比較

(2) 轉換成原形

從一系列的實驗結果可知，是否事先將單字轉換成原形，**對正確率之影響皆不大**（<1%）。甚至各個實驗當中，於單字不轉換成原形時，得到的正確率，都相當輕微地高於轉換成原形時的結果。因此，**過去研究**[3][4][6]**多有之「先將單字全部轉換成原形」一步驟**，其**效益有待商榷**。以下且節錄部分實驗結果以示意。

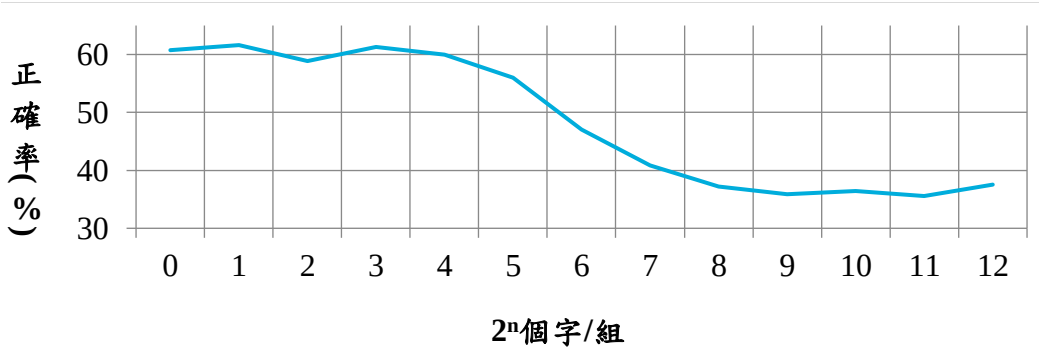
變因	轉換成原形	不轉換成原形
標準過程	63.80	64.43
分組加總(2^1 個詞/組)	61.62	61.91
分組加總(2^3 個詞/組)	61.34	61.91
分組加總(2^5 個詞/組)	56.02	56.86
分組加總(2^7 個詞/組)	40.90	41.18

表六、是否轉換成原形之正確率(%)比較（節錄）

(3) 分組、加總

在分組、加總之部分，基於實驗效率考量，本研究先嘗試 2 的各個指數，以將不同數量的詞劃分為一組、加總。

在 2^1~2^4 個詞一組時，正確率在 60%上下波動，故進一步以 2^1 到 2^4 的每個整數逐一測試。得到於每 7 個詞為一組時，有最高正確率 64.15%。



圖六、分組加總(2^n個字/組)之正確率(%)比較

雖然此正確率 64.15%略低於不分組（即各個單字分別計算出現次數）時的正確率 64.43%，但透過實驗結果，我們可以推測，在**小範圍（大約 n<=16）的情況下，將數個詞劃分為一組**，對於分類效能的影響並不大。因此，或許可以透過此法**濃縮特徵值數量**。

以此實驗為例。若將 24521 個特徵值濃縮成 24521 / 7 = 3503 個特徵值，對於正確率的影響僅 64.43% - 64.15% = 0.28%，卻可以大幅減少特徵值，縮短機器學習所需之時間。意即，未來在應用時，可以利用此法，以更快捷地建立新的難易度分級模型、判別篇章難易度。

(4) 計次數或頻率

計次數或頻率對於分級效能的影響，和是否分組加總有關；而是否分組加總，又和字彙集來源有關。

多個字為一組時，頻率版本效能較佳：由於 ERWL 僅分 6 級，每一級有 1,080 個單字，故在計算「單字出現次數」時得到的數據，受文長之干擾較大。但若以頻率計，資料更能聚焦於「各難度字所佔比例」，則可以消弭此問題。前後正確率差異顯著，達 50.84% - 35.67% = 15.17%。

為了驗證上述推測，我們將 ANC 依照詞頻排序，亦以 1,080 個單字為一組，以模仿 ERWL，而「計次數」所得之正確率 34.83%，的確低於「計頻率」之 40.45%。

然而，ANC 模仿 ERWL 分組加總所得正確率，無論計次數或計頻率，皆不及 ERWL。可能和語言使用習慣有關。因為 ERWL 為我國大考中心所編[7]，高中英文課文之編輯過程應有參考之；而 ANC 則為美國蒐集美式英語語料、依美式英語使用習慣（詞頻）排序而成。因此，同樣 1,080 個字為一組、分類我國英文課文的實驗中，ERWL 之分級效能便較 ANC 佳。

字彙集來源	分組加總*	計次數	計頻率
ERWL	是	35.67	**50.84**
ANC	是	34.83	40.45
	否	**64.43**	52.25

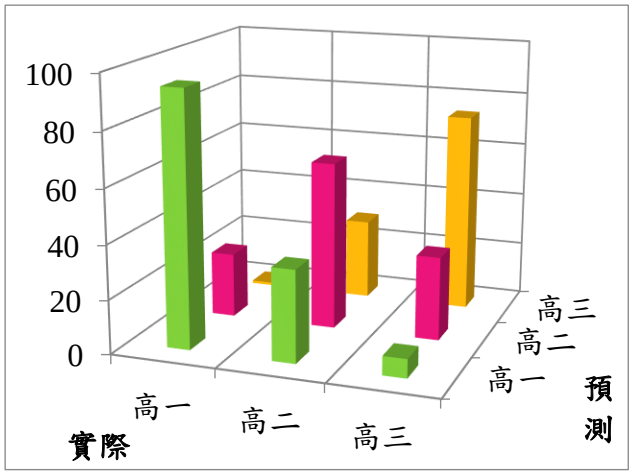
表七、單字特徵正確率(%)綜合比較

*註：由於 ERWL 中每一等級各有 1,080 個單字，為了 ERWL 與 ANC 之比較，在此 ANC 之「分組加總」結果係採用以 1,080 個單字為一組之結果。

(5) 分級情形

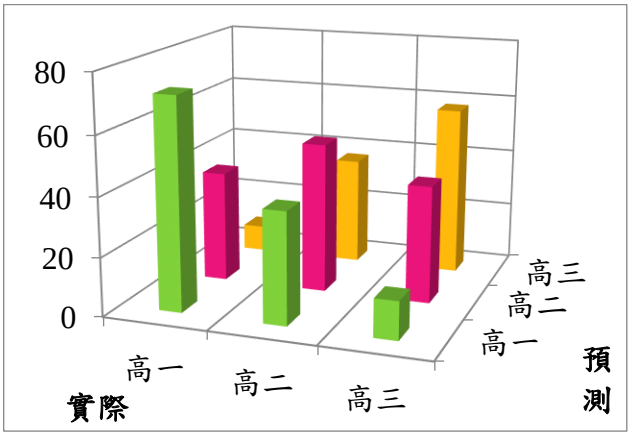
由分級情形表格，可以更進一步地分析確切的分級效果。以 ANC 之分級情形為例。實際是**高一**的文章，不太會被預測成**高三**文章（**誤判率 0.84%**）；實際是**高三**的文章，也較不易被預測成**高一**的文章（**誤判率 6.25%**）。故此分級已**具相當判別能力**。

其中，**高一最易判別**：高一課文不常出現高二、三的單字，會有清晰的「沒有高二高三難易度的單字」的特徵。句型亦是同理。而**高二則最難判別**：高一、二，以及高二、三之間的文章，皆較容易混淆在一起，高一、三之間則不太容易混淆。因為課文的難易度往往循序漸進地增加，因此前後兩年級之間的難易度差異度較小。這個結果符合預期。



表八、ANC 最佳分級情形（單位：篇）

預測 / 實際	高一	高二	高三	正確率 (%)
高一	**94**	24	1	78.33
高二	34	**62**	30	48.44
高三	7	31	**74**	64.35
正確率 (%)	69.12	52.10	68.52	**64.43**



表九、ERWL 最佳分級情形（單位：篇）

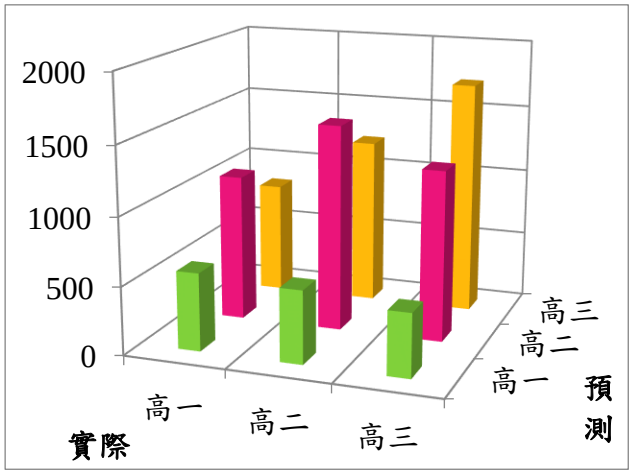
以最佳分級情形來說，ERWL 之效能不及 ANC，故 ERWL 柱狀圖之高低，亦較 ANC 為不分明。

2. 句型特徵

句型特徵的部分，包含以句為單位、以篇為單位之比較與分析。

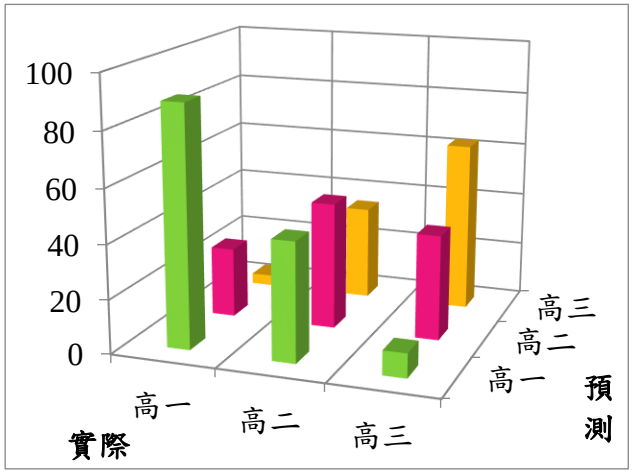
以篇為單位較易判別：相較於以句為單位，以篇為單位較易判別。尤其是一年級，且一三年級有較理想的判斷分布，雖然一年級的句子，易被誤判成二三年級，但少了會被判斷為三年級的難句，所以會有較明顯的句子難易度分布特徵。

以句為單位，難句較易判別：相較於簡單的句子（如高一句子），難句反而較易判別。因為簡單的句子在簡單、困難的文章都會出現，沒有清楚特點，所以較難判別；反之，較難的句子有明顯「在簡單文章中不會出現」的特徵，因此高三的句子預測數量呈現高三>高二>高一的理想情形。



表十、句型特徵分級情形（單位：句）

實際＼預測	高一	高二	高三	正確率(%)
高一	**573**	1078	828	23.11
高二	536	**1518**	1231	46.21
高三	466	1253	**1722**	50.04
正確率(%)	36.38	39.44	45.54	**41.42**



表十一、句型特徵分級情形（單位：篇）

實際＼預測	高一	高二	高三	正確率(%)
高一	**89**	26	4	74.79
高二	44	**47**	35	37.30
高三	9	39	**63**	56.76
正確率(%)	62.68	41.96	61.76	**55.90**

3. 表面特徵

表面特徵包含平均句長、平均音節數、各句長句數、各音節數字數。

由於過去相關研究使用的表面特徵，多為平均句長、平均音節數[5]，而經實驗知此二特徵對於本研究採用的語料之分級正確率並不理想（44.38%及 42.69%），故我們**基於此二舊特徵，發想出了兩項改良的特徵：各句長句數、各音節長字數**。預期透過直接統計不同長度之句數或字數之出現次數、頻率，再以簡單移動平均等方法修飾數據（公式請參照 16 各句長句數），以**改良舊特徵**（平均值）**無法看出分布**的缺陷。

經過實驗得知，各句長句數之分級正確率，的確較原特徵平均句長為佳，相差達 56.74% - 44.38% = 12.36%。

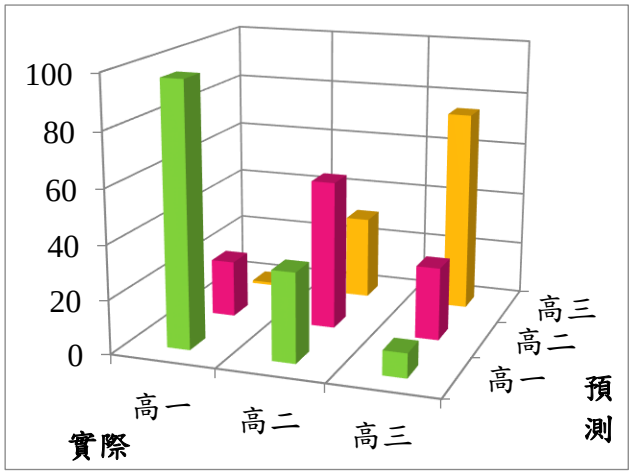
表面特徵內容	計次數	計頻率
平均句長	44.38	
平均音節數	42.69	
各句長句數	56.74	49.71
各音節長字數	36.23	37.64

表十二、表面特徵正確率（%）綜合比較

4. 整合

整合單字、句型、表面特徵，可以得到本研究**最高**之分類正確率 **65.04%**。由此可以推測各特徵具有一定的互補性，因此整合時正確率得以提升。

由下表中判斷正確之三個突出的高柱（實際為高一，預測為高一）、（實際為高二，預測為高二）、（實際為高三，預測為高三），以及高一、三之間亦不太會混淆（誤判率為 0.55%和 4.95%），可以看出**本分級的確具有相當的判別能力**。



表十三、整合特徵分級情形（單位：篇）

預測 / 實際	高一	高二	高三	正確率 (%)
高一	**97**	21	1	81.51
高二	34	**60**	32	47.62
高三	9	27	**75**	67.57
正確率 (%)	69.29	55.56	69.44	**65.04**

5. 特徵萃取小結

從整體的比較可以看出**單字特徵較易判別**：相較於句型特徵，單字特徵較容易判別。可能是因為一篇文章中，字數通常比句數多出許多，甚至是數十倍以上，故可以擷取的特徵亦較多，對於判別便比句型特徵有利。或是因為高中階段難易度差異，單字大於句型。

特徵萃取方式			計數量	計頻率
句型特徵	以句為單位		41.42	
	以篇為單位		56.46	48.42
單字特徵	ERWL		35.67	50.84
	ANC	分組*	34.83	40.45
		不分組	64.43	52.25
表面特徵	平均句長		44.38	
	平均音節數		42.69	
	各句長句數		56.74	49.71
	各音節長字數		36.23	37.64
整合特徵			65.04	

表十四、特徵萃取正確率（%）綜合比較

(三) 機器學習

這部分將以不同工具：，進行討論。經過實驗與探討，我們知道 SVM 應是相對最適合本研究之分類器，故對於 SVM 之分類效能等將再進行更進一步之實驗與探討，以確保得到該過程應得之最佳正確率。

1. SVM、RFC 和 DTC

在本研究的各項實驗中，以 SVM 分級所得之正確率，或多或少皆大於 RFC 和 DTC，因此，SVM 應比另兩項工具更適合篇章難易度分級。故前述實驗結果悉為以 SVM 分級之結果，以下也將再針對 SVM 進行延伸探討，而最後整合、應用成網頁時，亦是使用 SVM。

使用工具 / 特徵	SVM	RFC	DTC
單字特徵-ANC	64.43	46.80	51.37
句型特徵-句	39.50	37.84	35.83
句型特徵-篇(數量)	55.62	53.67	50.64
句型特徵-篇(頻率)	48.42	41.32	37.85

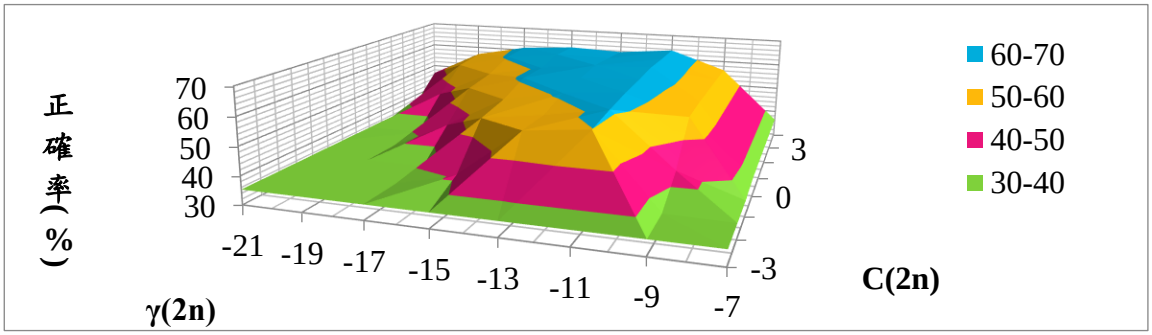
表十五、SVM、RFC、DTC 三項分類工具的正確率(%)比較

2. SVM 的參數

SVM 有 C、γ 兩個必須由使用者自行嘗試，才能得到最佳分類效果的參數[2]。因此，前述所有 SVM 實驗，在得到正確率之前，都必須進行 C、γ 參數之嘗試，即格點搜尋（Grid Search）[2]，而 C 和 γ 的嘗試意味著每個實驗都必須重複數十次以上，才能得到最佳參數組合，相當耗時。於是，我們亦對 SVM 參數對於正確率的影響，進行了簡單的探討。

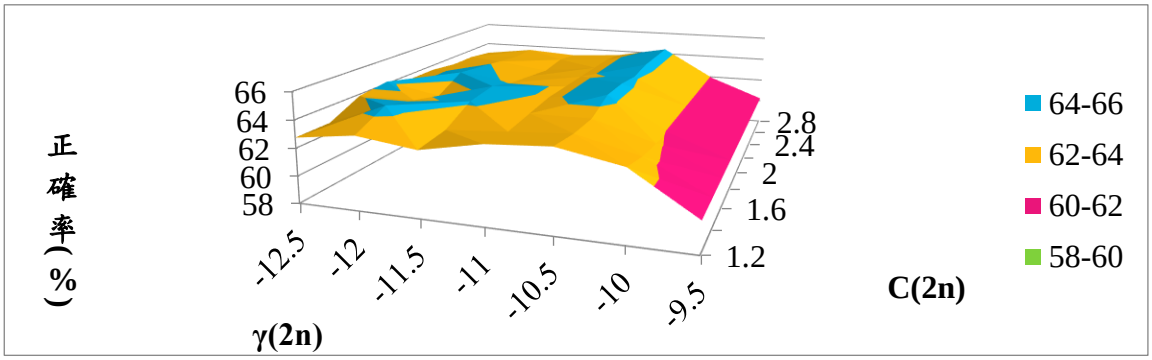
以單字特徵的實驗為例，在不調整參數時，SVM 工具預設為'auto'，亦會自行挑選參數，會得到一可能不及最佳結果 64.43%的粗略正確率，59.10%。

以 2 的次方測試 C 參數和 γ 參數。則當 $C=2^2, \gamma=2^{-11}$ 時，得到正確率 63.87%。下圖為部分測試所得正確率與兩參數之關係。



圖七、2^n 格點搜尋與正確率（一）

再進一步測試C參數和γ參數。取 $C=2^2, \gamma=2^{-10.5}$ 時，可以得到最高正確率 64.43%。和使用預設值所得之 59.10%相差達五個百分點，故調整參數為耗時但必要之步驟。



圖八、2^n 格點搜尋與正確率（二）

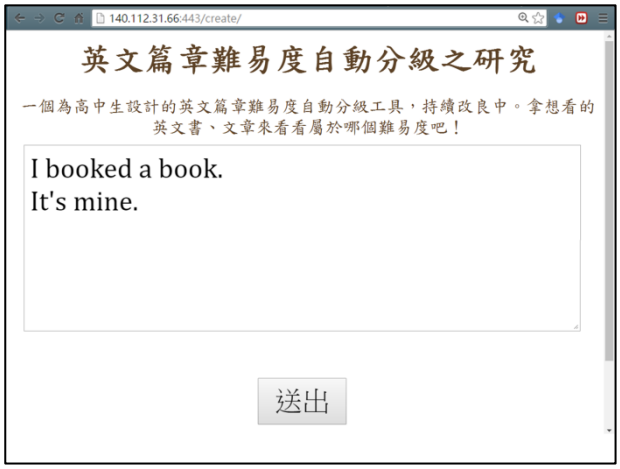
3. CNN sentence

由 CNN sentence 測試結果：內部資料的測試（訓練資料中含有測試資料）的正確率是 98.98%，外部資料的測試（訓練資料中不含測試資料）正確率則是 38.25%。

雖然 CNN sentence 在某些領域上能達到亮眼的分類效果[1]，但對本研究之分類效能似乎不盡理想，故僅能摒之不用。

(四) 網頁撰寫

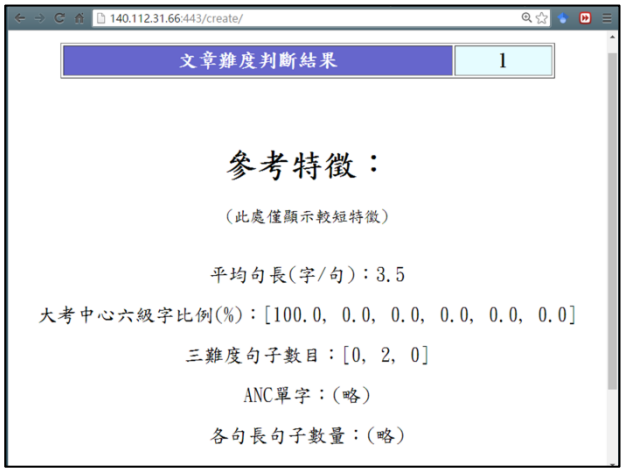
本研究亦包含完成可供大眾利用之英文篇章難易度自動分級網站，目前有中、英文兩個版本的網站，皆已在網路上公開供大眾使用。



圖九、中文網頁（一）

為方便國內學習者直接應用，故撰寫中文版網站。介面設計以清楚明瞭、容易使用為原則。

輸入欲分類的篇章，按下送出鍵，即可得到分級結果。

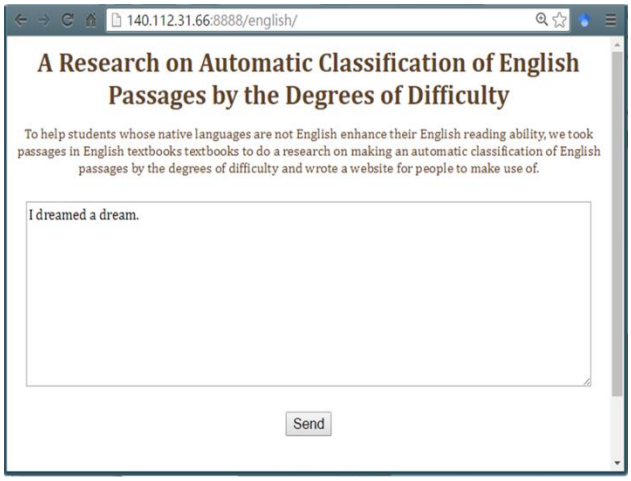


圖十、中文網頁（二）

按下送出鍵後會出現結果畫面。

上有「文章難度判斷結果」，以數字 1, 2, 3 代表該篇章被判斷為高一、二、三的難易度。

頁面其餘部分呈現一部分判斷時參考之特徵。由於某些特徵過於繁雜，故以省略。



圖十一、英文網頁

為與國際接軌，本研究亦撰寫英文版網站。使用方式悉同中文版，惟一切以英文呈現。

(五) 效能分析

為了評估效能，我們模擬過去相關研究的方法，用以分級我們的語料。經研究得知，過去國外對於英文篇章難易度分級，多採用適讀性公式[5]；而國內則多以全民英檢為分級準則[3,4,6]，故以下就這兩者進行模擬和討論。

1. 適讀性公式[5]

目前最久也最廣為使用的是 Flesch 適讀性公式（The Flesch Reading Ease Formula），採用音節數和平均句長來評量文章難易度。公式如下：

$$RE = 206.835 - (1.015 \times ASL) - (84.6 \times ASW)$$

其中，RE 為閱讀容易度（Reading Ease），ASL 為平均句長（Average Sentence Length），ASW 為字平均音節數（Average Number of Syllables per Word）。

由於適讀性公式之相關研究並未分析分級效能，故我們以適讀性公式使用的特徵，採 SVM 分級以模擬分級效能。

研究	採用／模擬特徵	分級／模擬方式
適讀性公式 [5]	ASL、ASW	公式計算
本研究	ASL、ASW、RE	SVM

表十六、適讀性公式與本研究模擬分級之比較

2. 全民英檢（GEPT）[3,4]

全民英檢相關研究使用的特徵以表面特徵為主（字數、句數、平均字長、ASL、ASW），再加上子句結構。表面特徵定義如下：

(1) 字數

篇章單字總數

(2) 句數

篇章句子總數

(3) 平均字長

$$\text{平均字長} = \frac{\text{單字字母總數}}{\text{單字數}}$$

(4) ASL、ASW（同適讀性公式[5]採用之定義）

至於子句結構，我們則以 Standford Parser 剖析句型以模擬。

我們使用 Scikit-learn 的決策樹分類器（Decision Tree Classifier, DTC），模擬過去研究使用之 C5.0 決策樹。

研究	採用／模擬特徵	分級／模擬方式
全民英檢	字數、句數、平均字長、ASL、ASW、子句結構	C5.0 決策樹
本研究	字數、句數、平均字長、ASL、ASW、子句結構	DTC（決策樹分類器）

表十七、全民英檢相關研究與本研究模擬分級之比較

經由上述過程，我們**模擬過去研究採用的方式，對我們的語料進行一樣的分級**（高中英文課文、分三級），其分類效能不盡理想，可見**本研究實際上相當具有挑戰性**，而我們**提出新方法所得到的 65.04%**，確實較於過去研究**效能更佳**。

模擬之研究	**該研究所採方法／我們模擬的方式**	**(模擬) 正確率**
適讀性公式[5]	方法：Flesch 適讀性公式（該研究未分析分級效能）	**42.22%**
	模擬：（相同特徵）SVM 分級	
全民英檢 （GEPT）[3,4]	方法：表面特徵、子句結構等，C5.0 決策樹	**57.31%**
	模擬：（相同特徵）Scikit-learn 決策樹工具	
本研究	**方法：單字及句型之計數與頻率、改良表面特徵， 支持向量機(SVM)、卷積神經網路(CNN)**	**65.04%**

表十八、過去類似分級、研究之模擬與比較

四、 結論與應用

將篇章單字特徵的部分，和句型特徵的部分整合後，使用 SVM 和 RFC 兩項工具，皆能得到更高的正確率。可知單字、句型兩種特徵應具有互補性，相結合使得分類效果提升。未來可以考慮將 CNN sentence 分類納入整合，使自動分級多考量文意等更深入之資訊，對於提升正確率應該頗有幫助。

因為課文的難易度往往循序漸進地增加，因此前後兩年級之間的難易度差異度較小，高一、二，以及高二、三之間的文章，皆較容易混淆在一起，高一、三之間則不太容易混淆。高二文章最難判斷，而這樣的結果符合預期。

將多個字劃為一個難度等級、同等級出現次數加總，以建立濃縮過的特徵矩陣時，將整個特徵矩陣除以總字數，能去除文長的影響，使特徵更單純，正確率也大幅提升。

就篇章難易度的分級來說，RFC 或 DTC 的分級效果皆不及 SVM。因此，SVM 應比另兩項工具更適合篇章難易度分級。

目前已將現階段研究完成之分級，呈現於網頁上，直接供大眾利用。未來若取得其他年段之語料（課文），亦可使用相同方式擴大分級範圍，增加應用範圍。亦可考慮建置其他語言之版本，對於各語言學習應有相當之助益。

未來若結合網路代理人技術，定期從網際網路蒐集大量英文文章並依難易度分級，甚至再結合現有之文章推薦系統，架構成一個完整的線上英語閱讀學習平台，讓來自不同地域與環境的英語學習者能夠善加利用，進而減少各地英語學習資源多寡之懸殊差異。

篇章難度的自動判斷研究有一定的難度，和過去的研究相比，本研究提出了許多新的特徵萃取方式，最後也確實達到較佳的分級效果。未來將嘗試更多方向，找出能夠讓難易度更明顯的特徵，並和目前方法加以結合改善。

五、 參考資料及其他

（一）、 參考資料

1. Yoon Kim (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Retrieved from http://emnlp2014.org/

2. Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin (2010). A Practical Guide to Support Vector Classification. Retrieved from https://www.csie.ntu.edu.tw/~cjlin/

3. 黃孝慈（2010）。利用字彙與子句結構進行全民英檢閱讀文章難易度分類之研究。長榮大學資訊管理學系碩士學位論文，未出版。

4. 許珀豪（2013）。應用文字探勘技術於英文文章難易度分類。國立政治大學資訊管理學系碩士學位論文，未出版。

5. 陳海泓（2013）。以適讀性公式挑選英文讀本之探究。教育資料與圖書館學，50(2)。

6. 楊子儀（2009）。基於代理人計數之適性化英文閱讀文章推薦系統。長榮大學資訊管理學系碩士學位論文，未出版。

7. 鄭恆雄、張郇慧、程玉秀、顧英秀、許秀玲、黃莉琪、劉欣潔、黃麗華（2001）。《大考中心高中英文參考詞彙表》編修研究計畫報告（第二期）。臺北市：大考中心高中英文參考詞彙表研究計畫小組。取自：http://www.ceec.edu.tw/

（二）、 附錄

1. 參數測試（節錄）

不轉換成原形、不排序分組，C 和 γ 參數的正確率測試(節錄) (單位：%)

$C(2^n)\backslash\gamma(2^n)$	-21	-19	-17	-15	-13	-11	-9	-7
-3	35.29	35.29	35.29	35.29	35.29	35.29	35.29	35.29
-2	35.29	35.29	35.29	40.90	57.70	59.94	35.57	35.29
-1	35.29	35.29	35.29	54.90	57.70	61.35	42.86	35.29
0	35.29	35.29	45.38	56.86	60.78	62.47	55.74	36.42
1	35.29	35.57	55.18	60.78	62.19	63.03	56.30	36.42
2	35.29	46.78	56.86	60.78	63.31	**63.87**	56.30	36.42
3	35.57	55.46	60.50	62.75	62.47	63.59	56.30	36.42
4	46.50	56.86	61.06	62.47	61.63	63.59	56.30	36.42

不轉換成原形、不排序分組的正確率測試結果。(單位：%)

$C(2^n)\backslash\gamma(2^n)$	-12.5	-12.0	-11.5	-11.0	-10.5	-10.0	-9.5
1.2	62.75	63.31	62.75	63.59	63.87	63.03	60.22
1.4	62.75	**64.43**	63.03	63.31	63.59	62.47	60.22
1.6	63.87	63.87	64.15	63.59	63.59	62.19	60.22
1.8	64.15	63.31	**64.43**	63.87	**64.43**	61.91	60.22
2.0	63.59	**64.43**	63.87	63.87	**64.43**	61.91	60.22
2.2	63.59	64.15	63.59	63.87	64.15	61.91	60.22
2.4	63.31	63.87	63.31	63.87	**64.43**	61.91	60.22
2.6	63.31	63.59	63.59	63.87	**64.43**	61.91	60.22
2.8	63.03	63.59	63.31	63.59	**64.43**	61.91	60.22

不轉換成原形、每 5 個一組，C 和 γ 參數正確率測試 (單位：%)

$C(2^n)\backslash\gamma(2^n)$	-18	-16	-14	-12	-10	-8	-6	-4
-4	35.29	35.29	44.82	59.66	35.57	35.29	35.29	35.29
-2	35.29	57.42	61.63	61.63	60.22	35.29	35.29	35.29
0	58.26	61.06	61.91	**62.75**	61.63	42.86	36.42	35.85
2	60.78	61.91	62.47	59.94	60.50	44.26	36.42	35.85
4	61.63	62.19	58.26	60.22	60.50	44.26	36.42	35.85
6	62.19	58.26	55.74	60.22	60.50	44.26	36.42	35.85
8	58.82	54.62	55.74	60.22	60.50	33.26	36.42	35.85
10	55.46	55.46	55.74	60.22	60.50	44.26	36.42	35.85

不轉換成原形、每 5 個一組，C 和 γ 參數正確率測試(縮小範圍) (單位：%)

C(2^n)\γ(2^n)	-13.5	-13.0	-12.5	-12.0	-11.5	-11.0	-10.5
-0.8	62.47	62.47	62.75	63.03	63.03	62.19	60.22
-0.6	62.47	62.47	62.75	62.75	**63.31**	62.19	61.63
-0.4	62.19	62.75	63.03	63.03	62.75	62.47	62.19
-0.2	62.75	63.03	62.75	62.75	62.75	63.03	62.47
0	62.75	63.03	62.47	62.75	63.03	**63.31**	**63.31**
0.2	63.03	62.47	62.47	63.03	62.47	**63.31**	62.75
0.4	62.75	62.19	62.75	**63.31**	61.91	62.47	62.19

不轉換成原形、排序分組測試。(測試 2 的 0 次方到 12 次方。)

2^n 個詞/組	0	1	2	3	4	5	6	7	8	9	10	11	12
個詞/組	1	2	4	8	16	32	64	128	256	512	1024	2048	4096
正確率(%)	59.10	**61.91**	61.06	**61.91**	57.98	56.86	46.78	41.18	37.26	35.85	35.85	35.85	35.85

不轉換成原形、排序分組測試。(2^0~2^5 逐數測試。)

個詞/組	1	2	3	4	5	6	7	8
正確率(%)	59.10	61.91	59.94	61.06	**62.75**	60.78	61.63	61.91
個詞/組	9	10	11	12	13	14	15	16
正確率(%)	61.06	59.94	61.06	58.82	58.54	59.10	57.98	57.98
個詞/組	17	18	19	20	21	22	23	24
正確率(%)	59.10	59.38	60.78	60.50	60.22	59.94	59.38	57.98
個詞/組	25	26	27	28	29	30	31	32
正確率(%)	58.82	58.54	57.70	57.70	57.98	57.98	57.42	56.86

轉換成原形，C 和 γ 參數的正確率測試(節錄) (單位：%)

C\γ(2^n)	-12	-13	-14	-15	-16
3.50	62.75	63.03	63.87	61.06	59.94
3.57	62.75	63.03	63.87	61.06	59.94
3.65	62.75	62.75	63.87	60.79	60.50
3.73	62.19	62.47	63.59	60.79	60.50
3.82	61.91	62.75	63.59	60.79	60.50
3.91	61.63	63.31	63.59	61.06	61.06
4.02	61.35	63.31	63.87	61.06	61.06
4.13	61.35	63.03	**64.15**	61.35	61.06

4.24	61.06	63.31	**64.15**	61.35	61.06
4.37	60.78	63.03	63.87	61.35	61.35

轉換成原形，範圍較小的 C 和 γ 參數正確率測試 (單位：%)

C\γ(2^n)	-14.8	-14.6	-14.2	-14.0	-13.8	-13.6	-13.4	-13.2
4.10	63.87	**64.15**	63.31	63.59	63.59	63.59	63.31	62.75
4.12	63.59	**64.15**	63.31	63.59	63.59	63.59	63.03	62.75
4.14	63.59	**64.15**	63.31	63.59	63.59	63.59	63.03	62.75
4.16	63.59	**64.15**	63.59	63.59	63.59	63.59	63.03	63.03
4.18	63.59	**64.15**	63.59	63.59	**64.15**	63.59	63.03	63.03
4.20	63.59	**64.15**	63.59	63.59	**64.15**	63.59	63.03	63.03
4.22	63.87	**64.15**	63.59	63.59	**64.15**	63.59	63.03	63.03
4.24	63.87	**64.15**	63.59	63.59	**64.15**	63.59	63.03	62.75
4.26	63.87	**64.15**	63.59	63.59	**64.15**	63.59	63.03	63.03

轉換成原形，範圍較小的 C 和 γ 參數正確率測試 (單位：%)

C\γ(2^n)	-14.8	-14.6	-14.2	-14.0	-13.8	-13.6	-13.4	-13.2
4.10	63.87	**64.15**	63.31	63.59	63.59	63.59	63.31	62.75
4.12	63.59	**64.15**	63.31	63.59	63.59	63.59	63.03	62.75
4.14	63.59	**64.15**	63.31	63.59	63.59	63.59	63.03	62.75
4.16	63.59	**64.15**	63.59	63.59	63.59	63.59	63.03	63.03
4.18	63.59	**64.15**	63.59	63.59	**64.15**	63.59	63.03	63.03
4.20	63.59	**64.15**	63.59	63.59	**64.15**	63.59	63.03	63.03
4.22	63.87	**64.15**	63.59	63.59	**64.15**	63.59	63.03	63.03
4.24	63.87	**64.15**	63.59	63.59	**64.15**	63.59	63.03	62.75
4.26	63.87	**64.15**	63.59	63.59	**64.15**	63.59	63.03	63.03

轉換成原形、排序分組、不調整參數之正確率

2^n 個詞/組	0	1	2	3	4	5	6	7	8	9	10	11	12
個詞/組	1	2	4	8	16	32	64	128	256	512	1024	2048	4096
正確率(%)	60.78	**61.62**	58.82	61.34	59.94	56.02	47.05	40.9	37.25	35.85	36.41	35.57	37.53

轉換成原形、排序分組、不調整參數之正確率(2^0~2^4)

個詞/組	1	2	3	4	5	6	7	8
正確率(%)	60.78	61.62	58.82	58.82	61.06	62.18	**63.03**	61.34
個詞/組	9	10	11	12	13	14	15	16
正確率(%)	57.98	57.7	59.38	56.58	58.54	57.14	56.58	59.94

轉換成原形，每 7 個詞為一組，得到的正確率(單位：%)

$C(2^n)\backslash\gamma(2^n)$	-16	-14	-12	-10	-8	-6
-14	35.29	35.29	35.29	35.29	35.29	35.29
-12	35.29	35.29	35.29	35.29	35.29	35.29
-10	35.29	35.29	35.29	35.29	35.29	35.29
-8	35.29	35.29	35.29	35.29	35.29	35.29
-6	35.29	35.29	35.29	35.29	35.29	35.29
-4	35.29	58.26	63.03	37.82	35.29	35.29
-2	61.06	**64.15**	62.18	57.98	35.29	35.29
0	63.59	62.47	62.47	61.34	42.02	36.41
2	61.34	61.06	59.66	61.34	43.42	36.41
4	61.62	59.66	60.78	61.34	43.42	36.41
6	59.10	56.58	60.78	61.34	43.42	36.41
8	57.42	56.86	60.78	61.34	43.42	36.41

不轉換成原形，使用 2 的指數測試參數得到的正確率(節錄)(單位：%)

$C(2^n)\backslash\gamma(2^n)$	-18.0	-17.5	-17.0	-16.5	-16.0	-15.5	-15.0	-14.5	-14.0	-13.5
-2.0	35.57	38.94	43.70	56.58	61.06	61.34	61.87	63.03	64.15	62.47
-1.5	38.66	44.26	56.86	61.06	61.06	63.87	62.75	63.87	62.75	62.75
-1.0	44.26	57.42	60.50	61.34	**64.43**	62.75	63.59	62.18	62.75	62.47
-0.5	57.70	60.50	61.34	64.15	62.47	63.59	62.18	62.75	63.03	62.47
0	60.50	60.78	63.87	62.18	63.59	61.90	62.47	63.03	62.47	62.47
0.5	60.78	63.87	62.18	63.59	61.90	62.75	62.75	62.18	61.34	61.62
1.0	63.59	62.18	63.87	61.90	62.75	61.90	62.18	61.34	61.34	61.90
1.5	62.47	63.87	62.18	62.75	62.18	61.90	61.34	61.06	61.62	61.34
2.0	63.87	62.18	62.75	61.90	61.34	60.50	60.78	61.34	61.06	61.06
2.5	62.18	62.75	61.90	61.06	60.50	61.06	61.90	61.34	61.06	61.62

2. 整合特徵程式碼

```
'''整合 1 (單字與句型)'''
# -*- coding: utf-8 -*-
import os
from sklearn import svm
from sklearn import ensemble
import pickle
import re
import json

anc_dic=dict()
anc_count=dict()
anc_post=dict()
dic=json.load(open('dic.json','r'))
table=[[0,0,0],[0,0,0],[0,0,0]]
dependency_list=json.load(open('depen.json','r'))
for line in open("anc.txt",'r'):
    text_list=line.split('\t')
    if text_list[0].isalpha()==False:
        if text_list[0].find('-')==-1:
            continue
    anc_dic[text_list[0]]=text_list[1]
    anc_count[text_list[1]]=0
for index_c in [1]:
    c=1
    #print("C = "+str(c))
    r_file.write("C = "+str(c)+"\n")
    for index_gamma in range(-2,10):
        r=[[0,0,0],[0,0,0],[0,0,0]]
        g=0.01*(2**index_gamma)
        print(g)
        right=list()
        total=list()
        right_sum=0.0
        total_sum=0.0
        for test in range(10):
            x=[]
            y=[]
            tx=[]
            ty=[]
            for grade in [1,2,3]:
                for num in range(1,120):
                    try:
                        fo=open('paragraph/%d/%d.t'%(grade,num),'r')
                    except:

                        break
                    lines=fo.readlines()
                    for item in anc_count:
                        anc_count[item]=0
                    temp_list=list()
                    for line in lines:
                        words= re.sub("[^\w]", " ",  line).split()
                        temp_list.extend(words)
                    if len(temp_list)<20: continue
                    for item in temp_list:
                        if anc_dic.get(item)!=None:
                            item=anc_dic[item]
```

```
                        anc_count[item]+=1
                record=[]
                for item in anc_count:
                    record.append(anc_count[item])
                ls=[]
                for i in range(len(record)/8+1):
                    s_sum=0
                    for j in range(0,8):
                        try:
                            s_sum+=record[i*8+j]
                        except:
                            if((i*8+j)>=len(record)):
                                break
                            else:
                                print('false_record')
                    ls.append(s_sum)
                al=[]
                al.extend(ls)
                al.extend(dependency_list[grade-1][num-1])
                if num%10 == test:
                    tx.append(al)
                    ty.append(grade-1)
                else:
                    x.append(al)
                    y.append(grade-1)
                fo.close()
            clf = svm.SVC(C=c)      #
            clf.fit(x,y)

            pre=clf.predict(tx)
            count=0
            correct=0

            for item in pre:
                r[ty[count]][item]+=1
                if ty[count]==item:
                    correct+=1

                count+=1
            right_sum+=correct
            total_sum+=count
        print(right_sum/total_sum)
        for gr in r:
            print(gr)

# -*- coding: utf-8 -*-
'''整合2(ANC+平均句長+大考中心單字)'''
import Word
import Len
import Vex
import nltk
import os
from sklearn import svm
import pickle

r=[[0,0,0,0],    #[y][pre]
   [0,0,0,0],
   [0,0,0,0],
   [0,0,0,0]]
```

```
max AVE=0
max c=0
max g=0
max r=[[0,0,0,0],    #[y][pre]
       [0,0,0,0],
       [0,0,0,0],
       [0,0,0,0]]

fl=open(" WLV Wfqc=T Vfqc=T.txt",'w')

for index c in range(9):
    index c = index c + 4
    c = 2.0**index c
    print("c = " + str(c))
    fl.write("c = " + str(c) + "\n")
    for index gamma in range(9):
        index gamma = index gamma - 13
        g = 2.0**index gamma
        print("g = " + str(g))
        fl.write("g = " + str(g) + "\n")
        right=list()
        total=list()
        for R in range(10):
            x=[]
            y=[]
            for label in [1,2,3]:
                for root, dirs, files in os.walk("paragraph/%d/"%label):
                    for f in files:
                        str f=str(f)
                        temp=str f.split(".")
                        if temp[0] == 'desktop':
                            continue
                        num=int(temp[0])
                        if num%10==R:
                            continue
                        o="paragraph/"+str(label)+"/"+str(f)
                        record = Word.Count(open(o,'r'),fqc=True)
                        record.extend(Len.tkn(open(o,'r')))
                        record.extend(Vex.Count(open(o,'r'),fqc=True))
                        x.append(record)
                        y.append(label)
                        #fo.close()
            clf = svm.SVC(C=c,gamma=g)
            clf.fit(x,y)
            x=[]
            y=[]
            for label in [1,2,3]:
                for root, dirs, files in os.walk("paragraph/%d/"%label):
                    for f in files:
                        str f=str(f)
                        temp=str f.split(".")
                        if temp[0] == 'desktop':
                            continue
                        num=int(temp[0])
                        if num%10!=R:
                            continue
                        o="paragraph/"+str(label)+"/"+str(f)
                        record = Word.Count(open(o,'r'),fqc=True)
                        record.extend(Len.tkn(open(o,'r')))
                        record.extend(Vex.Count(open(o,'r'),fqc=True))
```

```
                    x.append(record)
                    y.append(label)
                    #fo.close()
            pre=clf.predict(x)
            count=0
            correct=0

            for item in pre:
                if y[count]==item:
                    correct+=1
                    if item==1:
                        r[1][1]+=1
                    if item==2:
                        r[2][2]+=1
                    if item==3:
                        r[3][3]+=1
                else:
                    if y[count]==1:
                        if item==2:
                            r[1][2]+=1
                        if item==3:
                            r[1][3]+=1
                    if y[count]==2:
                        if item==1:
                            r[2][1]+=1
                        if item==3:
                            r[2][3]+=1
                    if y[count]==3:
                        if item==1:
                            r[3][1]+=1
                        if item==2:
                            r[3][2]+=1
                    count+=1
            right.append(correct)
            total.append(count)

        r sum=0
        for item in right:
            r sum+=item
        t sum=0
        for item in total:
            t sum+=item
        rate=float(r sum)/float(t sum)
        print("AVE = %5.3f%%"%(rate*100))
        fl.write("AVE   = %5.3f\n"%(rate*100))

        if rate>max AVE:
                max AVE=rate
                max r=r
                max c=c
                max g=g
        r=[[0,0,0,0],    #[y][pre]
           [0,0,0,0],
           [0,0,0,0],
           [0,0,0,0]]
    print("\n")
    fl.write("\n")

print("max c = "+str(max c))
print("max_g = "+str(max_g))
```

```
print("max AVE = %5.3f%%"%(max AVE*100))
fl.write("max c = "+str(c)+'\n')
fl.write("max g = "+str(g)+'\n')
fl.write("max AVE = %5.3f%%"%(max AVE*100)+"\n")

for item in max r:
    print(item[0], item[1], item[2], item[3])
    fl.write(str(item[0])+" "+str(item[1])+" "+str(item[2])+" "+str(item[3])+"\n")

fl.close()

'''整合 3(ANC+大考中心+平均句長+各句長句數)'''
from sklearn import svm
import nltk
import re
import json
import pickle
import Word
import Len
import Vex
dp dic=json.load(open('dic.json','r'))
def clf make(test):
    print(test)
    x=[]
    y=[]
    for grade in range(1,4):
        for num in range(1,127):
            if num%10 ==test:
                continue
            try:

                a=[]
                f=open('paragraph/%d/%d.t'%(grade,num),'r')
            except:
                continue

            text=f.read()
            text=text.replace('\n',' ')
            a.extend(Word.Count([text]))
            a.extend(Vex.Count([text],fqc=True))
            sents=nltk.sent tokenize(text)
            k=Len.tkn(sents)
            a.extend(k)
            a.extend(Len.Test(sents,range(2,4)))

            f1=open('paragraph/%d/%d.list'%(grade,num),'w')
            json.dump(a,f1)
            x.append(a)
            y.append(grade-1)

    clf p=svm.SVC()
    clf p.fit(x,y)
    f=open('clf p v.p','wb')
    pickle.dump(clf p,f)
    f.close()


def judge(text,grade=0,num=0):
```

```
    dl=[0,0,0]
    sl=[]

    a=[]
    text=text.replace('\n',' ')
    a.extend(Word.Count([text]))
    a.extend(Vex.Count([text],fqc=True))
    sents=nltk.sent_tokenize(text)
    k=Len.tkn(sents)
    a.extend(k)
    a.extend(Len.Test(sents,range(2,4)))

    clf_p=pickle.load(open('clf_p_v.p','rb'))
    pr=clf_p.predict([a])
    return pr[0]
if __name__=='__main__':
    table=[[0,0,0],[0,0,0],[0,0,0]]
    wrong=[[],[],[]]
    for test in range(10):
        clf_make(test)
        for grade in range(1,4):
            for num in range(1,127):
                if num%10==test:
                    try:
                        f= open('paragraph/%d/%d.t'%(grade,num),'r')
                    except:
                        print("%d %d "%(grade,num))
                        continue
                    text=f.read()
                    f.close()
                    text=text.replace('\n',' ')
                    ans=judge(text,grade,num)
                    table[grade-1][ans]+=1
                    if grade-1 !=ans:
                        wrong[grade-1].append(num)


    print(table)
    print(wrong)
'''整合 4(句型+平均句長+ANC 單字+大考中心單字'''
from sklearn import svm
from sklearn.externals import joblib
import re
import json
import pickle
import nltk
import Word
import Len
import Vex
import os
dp_dic=json.load(open('dic.json','r'))
def dependency(text,grade=None,sent=None):
    stanfordp='/nfs/cache/hhhuang/stanford-parser-full-2015-12-09/lexparser.sh
temp.txt'
    text=text.replace('\n',' ')
    f1=open('temp.txt','w')
    f1.write(text.strip())
    f1.close()
    rt=[]
    rd=os.popen(stanfordp).read()
```

```
    if grade != None:
        fw=open('paragraph/%d/%d.parse'%(grade,num),'w')
        fw.write(rd)
        fw.close()
    par=rd.split('\n')
    k=False
    sl=[]
    rt=[]
    for i,t in enumerate(par):
        if k:
            rt.append(t.split('(')[0])
        if len(t)==0:
            if len(rt)!=0:
                sl.append(rt)
                rt=[]
            k=k^True
    return sl
def sent make(ts):

    for test in [ts]:
        sx=[]
        sy=[]
        for grade in range(1,4):
            #print('grade'+str(grade))
            for num in range(1,128):
                pl=[]
                if num%10 ==test:
                    continue

                try:
                    f=open('paragraph/%d/%d.sent'%(grade,num),'r')
                    sentences=json.load(f)
                    f.close()
                except:
                    break
                for s in sentences:
                    if len(s)>5:
                        l=[0]*len(dp dic)
                        for mark in s:
                            if mark in dp dic:
                                l[dp dic[mark]]+=1
                        sx.append(l)
                        sy.append(grade-1)
                        s.append(l)
                        pl.append(s)

        clf s=svm.SVC(C=1,gamma=0.125)
        clf s.fit(sx,sy)
        f=open('clf s.p','wb')
        pickle.dump(clf s,f)
        f.close()

def clf make(test):
    print(test)
    x=[]
    y=[]
    for grade in range(1,4):
        for num in range(1,127):
            if num%10 ==test:
                continue
```

```
            try:
                f=open('paragraph/%d/%d.prdl'%(grade,num),'r')
                sentences=json.load(f)
                f.close()
            except:
                break

            f=open('paragraph/%d/%d.t'%(grade,num),'r')
            lines=f.readlines()
            lt=[]
            a=[]
            a.extend(sentences)
            lt.append(len(a))
            a.extend(Word.Count(lines))
            lt.append(len(a))
            k=Len.tkn(lines)

            a.extend(k)
            lt.append(len(a))
            v=Vex.Count(lines,fqc=True)
            a.extend(v)
            lt.append(len(a))
            print('a')
            with open('list.json','w')as fw:
                json.dump(lt,fw)

            x.append(a)
            y.append(grade-1)

    clf p=svm.SVC()
    clf p.fit(x,y)
    joblib.dump(clf p,'clf/clf p.p')


def judge(text,grade=None,num=None):

    dl=[0,0,0]
    sl=[]
    sent=dependency(text,grade,num)

    for rt in sent:
        l=[0]*len(dp dic)
        for tag in rt:
            if tag in dp dic:
                l[dp dic[tag]]+=1
        sl.append(l)
    if grade != None:
        fw=open('paragraph/%d/%d.sl'%(grade,num),'w')
        json.dump(sl,fw)
        fw.close()

    clf s=pickle.load(open('clf s.p','rb'))
    pr=clf s.predict(sl)
    for ans in pr:
        dl[ans]+=1
    if grade != None:
        fw=open('paragraph/%d/%d.prdl'%(grade,num),'w')
        json.dump(dl,fw)
        fw.close()
    a=[]
```

```
    a.extend(dl)
    a.extend(Word.Count([text],fqc=False))
    k=Len.tkn([text])
    v=Vex.Count([text],fqc=True)
    a.extend(k)
    a.extend(v)
    if grade != None:
        fw=open('paragraph/%d/%d.judge'%(grade,num),'w')
        json.dump(a,fw)
        fw.close()
    clf p=joblib.load('clf p.p')
    pr=clf p.predict([a])
    #return pr[0],v,k
    return pr[0]
if  name  ==' main  ':
    table=[[0,0,0],[0,0,0],[0,0,0]]

    for test in range(10):
        clf make(test)
        sent make(test)
        for grade in range(1,4):
            for num in range(1,127):
                if num%10==test:

                    try:
                        f=open('paragraph/%d/%d.t'%(grade,num),'r')
                    except:
                        break
                    text=f.read()
                    f.close()
                    text=text.replace('\n',' ')
                    #ans,v,k=judge(text,grade,num)
                    ans=judge(text)
                    table[grade-1][ans]+=1
    print(table)
```

3. 文獻探討

文獻探討完整整理

研究		工具	特徵	探討
適讀性公式	早期國外諸研究	（受限於技術）	詞彙難易度、句子數目等資訊	一、學理上有可議之處（黃昭憲等，2010） 二、適度性公式未考量是否以英文作為母語，對國內學生來說不等於難度分級，不貼近需求
	陳海泓，2012	Microsoft Word 內建工具		
近年國內關於英文文章難易度分類的研究	楊子儀，2009	C5.0 決策樹	較深入的資訊，如子句結構、片語結構等	以全民英檢（GEPT）作為語料，僅採其中的初、中、中高三個級別進行難易度分級： (一) 無法對應國內學生實況 (二) 分級過於粗略
	黃孝慈，2010			
	許珀豪，2013	k-最鄰近演算法		
	黃昭憲等 2010	WEKA 內建分類器（J48 決策樹、LMT 決策樹、ANN 類神經網路和 Ridor 規則(rule)）	較深入的資訊、中文提示	語料為高中英文一到四冊附的閱讀測驗短文，最高達到 53.6% 的分類正確性。不適合判別英文原文的難易度： 一、著重題型難易度分析 二、以閱讀測驗附的中文提示作為特徵之一
我們的構想		較新工具	詞彙向量、句型結構等較深入的資訊	一、直接以高中英文課文作為語料（對應國內學生實況） 二、應用較新的工具

4. SVM 補充整理[2]

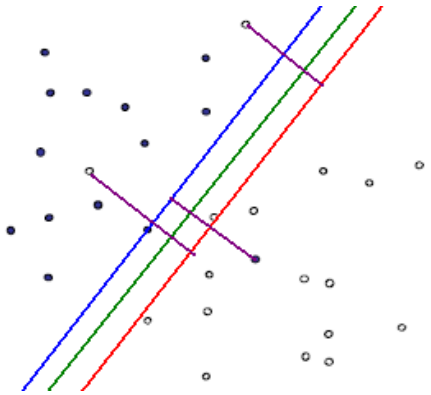
C 是誤差項的懲罰參數(penalty parameter)，預設為 1.0(對一個分錯的點的懲罰函數就是這個點到其正確位置的距離)；γ 則是核係數(kernel coefficient)，預設為自動(auto)。

尋找適當的 C 參數和 gamma 參數最好的方法是逐個嘗試 2 的指數。(例如：$2^{-15}, 2^{-14}, 2^{-13},\ldots, 2^{14}, 2^{15}$)

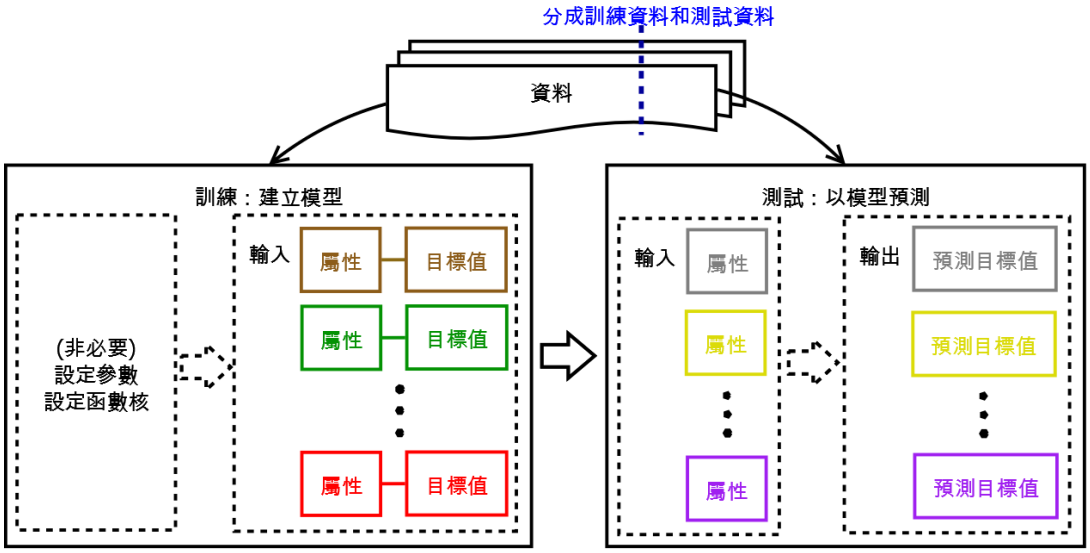
$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned}$$

上式為 SVM 基本公式[2]。

下圖為懲罰參數的圖解。(圖片來源：http://leftnoteasy.cnblogs.com)



SVM 基本運作方式圖解。RFC 及 DTC 同理。



5. 自建詞頻排序

(1) 以訓練資料（即課文）建立單字的「出現次數表」，和 ANC 有些微差異。

(2) 以訓練資料建立的出現次數表，取出現次數大於 10 的單字(意即不考慮太少見的單字)，當 8 個單字為一組，不調整 C 和 gamma，即可得到 64.71%的正確率。

關於詞彙的排序分組，由於排序時使用的語料 ANC 來自美國，依據語言習慣的不同，其詞頻和我國人編纂的英文課文之詞頻可能有異，因此其排序或許不是最契合難易度，可能是影響結果的原因。於是，我們試著以英文課文作為語料，計算新的計數取代 ANC 原先的計數，再依計數由高至低排序，發現其排序真的和 ANC 的排序有部分差異，如"that"的排序由第 12 升至第 7，可能和關係代名詞在中學階段屬於重要文法，在課文中會時常出現有關。由此可知，我國英文課文使用英文詞彙的習慣，的確和以英文作為母語的人相異。但以我們製作英文課文計數排序後得到的正確率，在不轉成原形、不調整參數的情況下，最高為 23 個詞彙一組時的 64.71%，僅較以相同方式實驗但使用 ANC 得到的 61.91%，高出不到 3%，並不是非常顯著，為了不使實驗太過複雜，仍以 ANC 作為排序所使用的語料。日後可以再針對詞彙部分的英文課文編纂習慣，進行更多嘗試提升正確率的實驗。

the	the
of	to
and	of
to	and
a	a
in	in
is	that
for	is
i	you
it	for
with	was
that	are
was	i
on	it
's	as
as	on
by	with
be	be
are	have
at	he
he	his
but	they
from	can
this	not
not	from
you	or
or	people
an	their
n't	your
were	at
they	this
his	by

（**左：ANC 詞頻排序；右：訓練資料詞頻排序**）

【評語】190005

1. 本件作品完成度高，且有不錯的實驗用以驗證其理論基礎。

2. 建議可多增加 training data size 及確保 ground truth 的正確性，以提升研究成果的實用性。

Intel 特別獎評語：

1. Focus 在文法上，字的結構分析來決定難易無法辨識內容和學生的可理解的程度（分級後無法幫助學好英文）

2. 難易的程度和興趣有很高的關聯性，但這是一個不錯的分析和研究，但是對台灣學生幫助有限，present 的技巧更以再大方有自信一些

3. 建議以同樣的處理手法，但重新定義題目例如辨識文章作者／作者寫作風格／etc.