隊長：資工二 B06705057 黃資翔
隊員：資管二 B06705058 劉品枘
資管二 B06705001 楊力行
資管二 B06705009 任恬儀
教授：李弘毅　教授

新聞立場分析 News Retrieval

問題介紹：

具爭議性議題的新聞一直是閱聽人關注與討論的焦點，例如：美國牛肉開放進口、死刑廢除、多元成家等。不論是政治、經濟、教育、兩性、能源、環保等公共議題，新聞媒體常需報導不同的立場。若能從大量的新聞文件裡，快速搜尋各種爭議性議題中具特定立場的新聞，不但有助於人們理解不同立場對這些議題的認知與價值觀，對制定決策的過程而言，也相當有參考價值。 (取自新聞立場檢索比賽簡介)

動機：

經過投票表決，新聞立場分析和肺炎X光呈現 2:2，但是評估組員們的顯示卡效能後，加上在 8 次的作業中，我們皆對自然語言蠻有興趣 (HW6)，因此我們選擇了新聞立場分析作為我們的期末專案主題。也希望可以利用期末專題，來分析台灣各大新聞台的立場是否客觀，或是我們可以找出某些新聞台的立場是否單一。

資料預處理與資訊擷取：

我們用若干種方式來處理我們的資料，即新聞內容。第一，我們利用 bag of word 來統計每一篇新聞 (共十萬篇) 與問題 (共二十個) 的詞頻，用一個 8000 維度的向量來表示。第二，我們統計每一個詞總共出現在幾篇文章之中，這部分的資料會應用在 inverse document frequency (IDF)。第三，我們利用 word to vec 來將每一個詞對應到一個向量。資訊擷取部分，我們特別處理每篇文章的最後一段，希望可以藉由最後一段文章來分析該立場。

實作方法：

我們分別介紹 simple baseline 與 strong baseline 的實作方法，其中 strong baseline 的方法是基於 simple baseline 的方法做些修改，改善。

simple baseline實作：

一開始我們只有採用 tf-idf 演算法，就能夠過simple baseline了。首先必須要忽略一些特殊字元，例如標點符號、+、-、空白鍵以及換行字元，因此我們將這些特殊字元存在一個 list 中。接著讀入助教給的 json 檔，並使用 jieba 及 hw6 中助教給的 dict.txt.big 對其進行斷詞處理。

接下來要觀察一個詞有多常被使用，先設置兩個字典 word_to_freq 以及 word_terms_freq，前者是負責記錄在所有文章中這個詞總共出現了幾次，而後者則是記錄某個詞出現在多少文章之中 (也就是說同一個文章重複出現也只會算一次，IDF)。

然後再設置一個字典 word_to_index，利用剛剛的 word_to_freq 來讓整個字典由出現次數多排到出現次數低，而第8000名以後的字詞由於出現次數太少所以就不放進word_to_index裡面。接著就將 word_to_index 以及 word_terms_freq 存入 json 檔案之後使用。(排序過的詞總出現次數、詞出現在多少文章中)。

接著另外存一個二維陣列 news_bag，即紀錄每個文章的 bag of word。然後我們對 query 做一樣的事情，但我們手動將 QS_1.csv 中的 20 個標題切割並保留重要的詞，例如第一個標題我們就存成陣列 [“通姦", ”刑罰”, ”除罪”]。這麼做其實是因為 jieba 斷詞沒有那麼精確，由於只有二十個問題，這部份我們就自己額外處理。接著我們直接算出每一個問題對應到每一篇文章的 tf-idf 為多少，取前 300 名當作最後答案，其中分數的算法如下，

$$query[i][j] = \frac{query\_bag[i][j]}{query\_bag\_length} * log\frac{document\_num}{word\_terms\_freq[j]}$$

$$news[i][j] = \frac{news\_bag[i][j]}{news\_bag\_length} * log\frac{document\_num}{word\_terms\_freq[j]}$$

$$score[i][j] = \frac{query[i] * news[j]}{||query[i]|| * ||news[j]||}$$

query$[i][j]$ 與 news$[i][j]$: 意思為第 i 篇文章 (問題) 中的第 j index 的字

document_num : 新聞數量，在這邊是十萬

Score$[i][j]$: 第 i 個問題與第 j 個問題的分數

Simple baseline 的分數以及問題：

1. 由於只進行了 tf-idf，只在乎詞頻，權重不在乎文法
2. tf-idf 的演算法還是會被一些詞頻低卻不重要的詞影響甚大
3. tf-idf 無法了解支持與反對
4. tf-idf 假設了詞頻較低的詞較重要，但實際上不一定

只使用 tf-idf 的分數只能達到 0.158，離 strong 仍有一段距離。

Strong baseline 的實作：

為了補足上述的問題，我們要使用 word2vec 進行更多的判斷，並且手動增加參數來調整權重。首先，我們使用所有的文章進行 word2vec 的訓練，並將 model 存好。我們額外新增兩個二維陣列query_vec、news_vec，負責的是將每一個詞的 word2vec 與 IDF 相乘來得到加權平均的分數，而非只看 BAG 與 IDF。其中 query_vec 是針對 20 個標題，而 news_vec 則是針對全部的文章，算分方式與 simple baseline 中三個式子類似。由於我們是利用加權平均的 vector 來表示某一篇文章或問題，因此在前 300 名中時常會找到連關鍵字都沒出現的文章，這顯然不是我們要的，因此我們限定該文章必須出現至少一個問題的關鍵字，即 $query\_bag[i] * news\_bag[j] > 0$ 我們才考慮該文章為最終答案。雖然 word2vec 某種程度上可以分析文章的立場，但效果有限，以下我們介紹 RNN 模型。

RNN 的嘗試與架構：

針對最後一段話，使用我們之前 train 好的 word2vec 的模型進行 RNN，將 padding length 設為 500，並且將 embedding layer 設為True，讓詞可以與我們的 RNN 一起訓練。

使用 keras 實作，堆疊一層 LSTM (128, dropout = 0.5)，一次 batchnormalization，最後一層是sigmoid，使用 binary crossentropy 當作loss function、adam 為optimizer。如果sigmoid > 0.5，代表這個文章是支持立場，反之則為反對立場。

模型其實表現得沒有很好，因為訓練資料稍顯過少，且資料在標記上也有些問題，因為若問題是支持立場，則標記為 0 (當 TD.csv 裡的 relevance 是 0)，1 (當 TD.csv 裡的 relevance 是 1, 2, 3)，若問題立場是反對，則標記相反，如此才可以讓輸出大於 0.5 時代表文章是支持。因為效果不是很好，我們有嘗試讓變數減少，但其

實差不多。事實上由於有五題的訓練資料與問題一樣，因此在這五個問題中 RNN 的表現很好，但其他的十五個問題就很普通，在這十五題中整體分數沒有多大的進步。

Strong baseline 的分數以及問題：

1. RNN 對於未看過的問題表現普通
2. 由於主要仍是 tf-idf 為基準，問題會被一些不重要的詞影響，例如 Q_09：支持中國學生納入健保，很容易找到跟中國有關卻跟健保無關的文章，若把 “中國” 的關鍵字刪除也不見得比較好，因為 “中國” 這個詞也需要佔一部分比例。與 simple baseline 有一樣的問題，tf-idf 假設了低詞頻較重要，但實際上不見得如此，如果能找到方法替代 IDF 對於每個詞的重要性，應該可以改善此問題
3. Word2vec 中與 “支持” 有關的詞與 “反對” 有關的詞向量太相近，因此在立場判斷輔助上效果有限 (但仍有幫助)

進行 Word2Vec 與 tf-idf 與 RNN 之後，由於判斷的基準變多了，分數上升到了 0.35。

實驗與討論

我們實驗了各種方法，來進一步提升準確率

1. TF-IDF 演算法

   原本的算法列在 simple baseline，以下為修改的算法

$$query[i][j] = \frac{query\_bag[i][j]}{query\_bag\_length} * (log \frac{document\_num}{word\_terms\_freq[j]})^{p_1}$$

$$news[i][j] = \frac{news\_bag[i][j]}{news\_bag\_length} * (log \frac{document\_num}{word\_terms\_freq[j]})^{p_2}$$

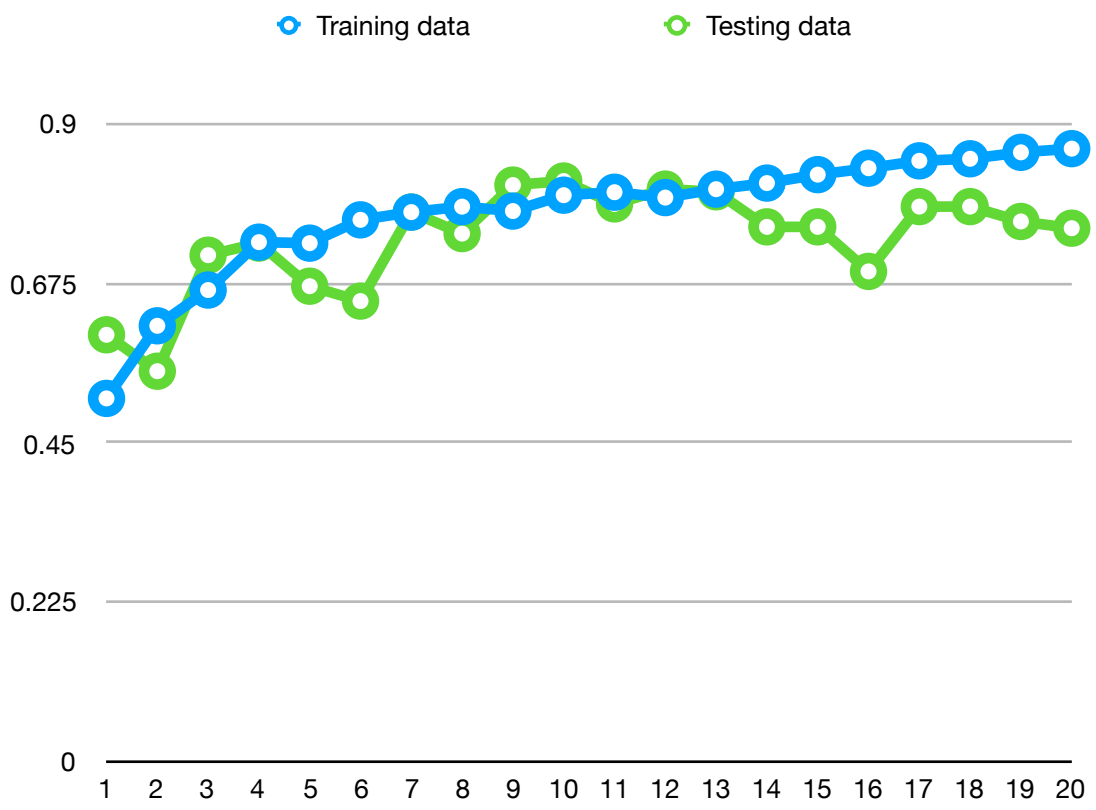
   p_1 與 p_2 決定了 IDF 的權重要多少，這兩個值越大表示我們越傾向認為詞頻較低的詞較重要，經過實驗後找到 $p_1 = 0.9$，$p_2 = 1.1$ 時分數最高，我認為這蠻合理的，因為我們不完全相信詞頻越低的詞越重要，我們也無法完全依靠 terms frequency 而不考慮 inverse document frequency，因此他們數值不應該太小或太大。

2. Word to vector 向量長度

向量長度	分數
200	0.3770
1000	0.3840
1500	0.3844

將一個中文詞轉換成向量，若向量越大，某種程度上來說我們把每一個詞分得比較細 (如果每一個維度都有用到的話)，因此理論上向量長度不能太短，但太長似乎就沒那麼有意義，甚至有可能效果更差。因為我們有十萬筆文章，每一個文章又有很多個字，因此選擇約 1000 維度恰恰好，訓練資料也非常足夠。

3. RNN 訓練曲線



我們可以發現 testing data 的準確率起起伏伏，可能是訓練資料過於少，因此 testing data 不是很穩定。降低訓練變數的個數也無法讓 testing accuracy 收斂，由此可知，利用深度學習有點不太穩定，我們未來會繼續思考如何利用少量的資料來訓練出穩定的模型。

4. 同義詞

很多字其實代表同一個意思，如果兩個同義詞卻使用不同的 IDF 蠻奇怪的，因此我們實驗將同義詞合併使用同一個 IDF 。例如：[“同意", "贊同", “支持”, “認同”] 為同義詞，只要文章出現四個同義詞中的任何一個就算進他們的 word_terms_freq。我們是利用 word to vector 來找同義詞的。

	分數
不考慮同義詞	0.371
考慮同義詞	0.377

5. 支持與反對 (word to vector)

Word to vector 中我們發現支持與反對兩者的向量接近，因此這是word to vector 無法準確判斷立場的主要原因。我們找出所有的反對詞語與支持詞語，我們人工讓 向量(支持) = -向量(反對)。首先我們先定義 “向量(支持)” 為所有與 ”支持” 相關的詞的平均，而 ”向量(反對)” 為其加負好，如此一來我們將支持與反對兩個詞拆得非常遠。理論上這樣做應該能較精確的判斷立場，但實際上分數不增反降。可能是因為我們只是純粹讓 向量(支持) 與 向量(反對) 離非常遠，但我們沒有考慮到其他詞語，且 向量(反對) 所在的位置再也沒有任何意義了。

結論：

這個比賽的目的是為了要找出特定立場的新聞，但其實是非常困難的，因為主辦單位給的訓練資料有點少 (只有四千多筆)，因此要利用深度學習來分析文章的立場有點困難。在訓練資料少的情況下也許只能用簡單的線性模型 (但我覺得線性模型應該無法解決這複雜的問題) 或是我們可以試著使用半監督式學習或無監督式學習。這兩種因為我們尚未想出實際的方法，因此未能實現。在立場分析這一塊我們仍有非常大的進步空間。在相關性中，我們約能找到 8 成的文章是有相關的，在排名後面的文章，事實上常常是與問題毫無關係，這部份可能是文章數太少 (十萬個)，在比賽的第二階段有一百萬筆文章，也許可以改善這種情況。而我們最後的模型是利用 TF-IDF，word2vec，RNN 的結合，可以達到 0.39 的準確率。

心得：

黃資翔

這次期末專案我體會到了，簡單就是最好，簡單的模型往往可以得到蠻不錯的結果，而這次題目中複雜的模型反而無法做得很好，這讓我想到我之前修的機器學習基石，林軒田教授在最後一堂課強調 simple, simple, simple。這門課讓我學到很多機器學習的方法，給我一個概略性的導覽，最後能在期末報告中用出上課學過的一些技巧覺得很棒。

劉品枘

立場分析除了機器學習之外，還需要結合其他的舊有知識、甚至是其他領域的演算法，讓本來只是train model的作業變得更能貼近實務。本來執著於只使用RNN，但是一整篇文章過長、冗言過多，只使用RNN的方法準確度很差。TF-IDF對我來說是新的演算法，很驚訝能夠用這麼簡單的方式來判別文章與給定標題的關係，準確率也不算低，是除了RNN以外非常有效的演算法，如果有夠好的電腦以及夠好的RNN，結合TF-IDF應該能有非常大的效用。

楊力行

在這次的專題中， 我發現了有時候簡單的模型如tf idf 會比複雜的模型如rnn 還要更合適,得到的分數也更高，因此，我了解到了，並不是越複雜的模型就越好，而是要根據所做的題目，以及該題目所提供的資料來選擇最合適的方法。

任恬儀

比較大的困難點，算是要把文章的所有word2vec 求出來取平均，然後把所有word2vec 去和quest算距離，過程中常常會超出記憶體的限制，導制整個程式崩潰，跑很多次，然後 early simple 就飛走了。因為word2vec很大的程度上代表著語意，所以取word2vec的平均能一定程度的代表文意。這學期我們學了很多深度學習相關的理論，但是有時候實作上，卻發現一般的計算或是經驗、直覺衍生出的模型，可以在更好的效能下達到一定的效果。

Reference:

TF-IDF : https://zh.wikipedia.org/wiki/Tf-idf

Google 搜尋演算法：https://www.google.com/intl/zh-TW/search/howsearchworks/algorithms/

詞性標註：https://www.itread01.com/content/1544598122.html