2022 年臺灣國際科學展覽會
優勝作品專輯

作品編號　**190025**

參展科別　電腦科學與資訊工程

作品名稱　摘要演算法和語句分析之關聯性

得獎獎項　二等獎

就讀學校　臺北市私立薇閣高級中學

　　臺北市私立復興實驗高級中學

指導教師　陳信希

作者姓名　張舒晴、高翊庭

關鍵詞　摘要演算法、文章分析

作者簡介

我是張舒晴，目前就讀於薇閣高級中學國際班的十一年級，喜歡滑雪和英文辯論，也對於軟體開發有興趣。這次的研究讓我有機會接觸到資訊科學一個全新的領域，也激發了我對深度學習的好奇心，我也很高興能夠和同學以及老師合作，參加此屆的國際科展。

我是高翊庭，就讀復興實驗高級中學雙語部十一年級，喜歡旅遊以及攝影，也對人工智慧與機器學習頗感興趣。在這次研究中，很高興能有機會體會真正科學研究的精神與實驗過程，也非常感謝我的夥伴與一路上幫助我們的老師與同學，我們未來也會持續努力，繼續朝我們的夢想邁進。

壹、前言

一、研究動機

在這個資訊發達的時代，網路充滿著五花八門的資訊，導致我們在查詢資料時會因為這些雜亂且未經過濾的資料浪費許多時間，其中最為氾濫的便是點擊誘餌（clickbait），此種新聞常常有著吸引人的標題，而內容卻不會與主題相符，人們也常常在讀完整篇文章後才意識到自己浪費了許多時間在無意義的資訊上面。解決此問題很常用的方法之一便是運用摘要演算法來讓讀者先對新聞有一個大概的理解，不過，雖然摘要演算法越來越普及，但產生出來的摘要仍會和人為判斷的結果有所差距，進而造成閱讀理解上的錯誤以及偏差，所以我們想要藉由這次研究，從一個嶄新的角度切入，探討摘要演算法和句型分析之間的關係，融合原本向量建構的方式以及語句結構的分析來測試摘要的準確度，並且由結果研發出一個可以產生出更為精確的主旨之摘要演算法，除此之外，我們也會融合實地調查以及搜集意見的方式來更進一步探討人們思模式與產生出的摘要之關聯性。

二、研究目的

此研究將藉由四步驟 – 製作摘要演算法、分析語句、合併兩者並且比較輸出、以及進行實際訪問與調查 – 將複雜的英文文章化繁為簡，減少閱讀的時間，並且產生一個最接近人們思考方式的摘要法。首先，此研究將比較不同摘要法之利與弊，並改進摘要演算法使它能產生最精確的摘要。接下來，將文章解剖分析，藉由關鍵字提取等方法辨別出文章中的語句關聯性。再者，此研究會將上述兩步驟得出的結果整合，編寫出一套能產生出更精確之摘要的演算法。最終，我們會將摘要模型生產出的摘要和其他摘要演算法比較，得出大眾較為偏好的摘要法，並且分析原因和統整結果。對於未來的發展，我們希望將最終產生出的演算法呈現於網站上，供大眾使用。

貳、研究方法或過程

一、研究設備

(一)、硬體設備

1. 筆記型電腦：Intel Core i7-6500U 2.50 GHz、Intel Core i5 1.6 GHz

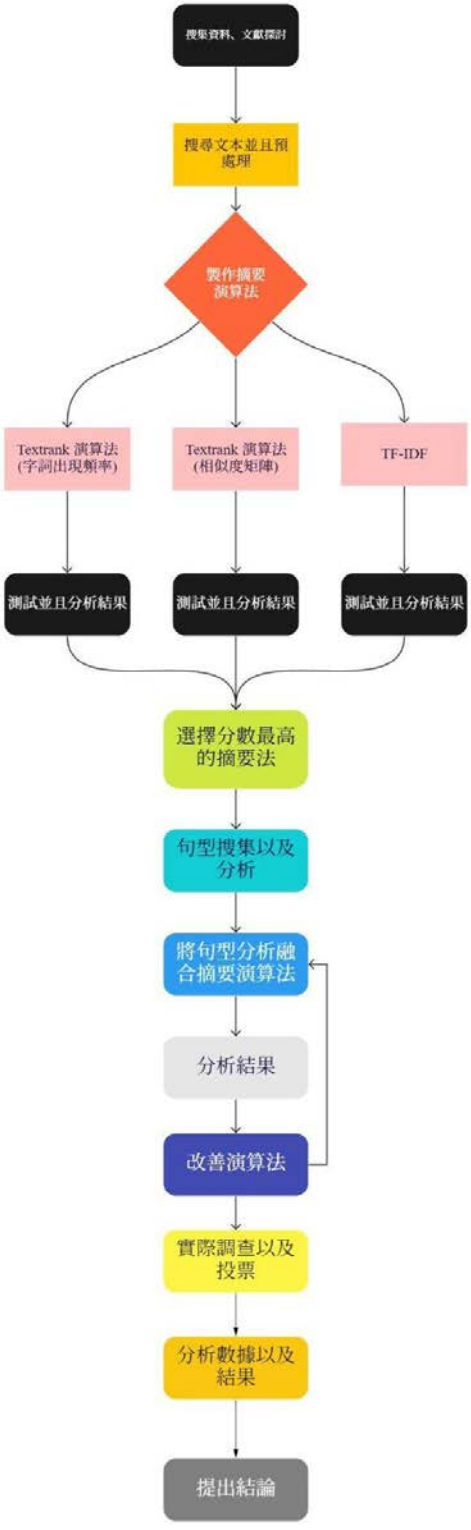
(二)、軟體設備

1. 程式語言：Python、HTML、CSS、JavaScript
2. 編程環境：Pycharm、Jupyter Notebook、Google Colab、Visual Studio Code

3. 程式/資料存放：GitHub

4. 統計數據：Word、Excel

二、研究架構



圖一：研究架構流程表

三、研究過程

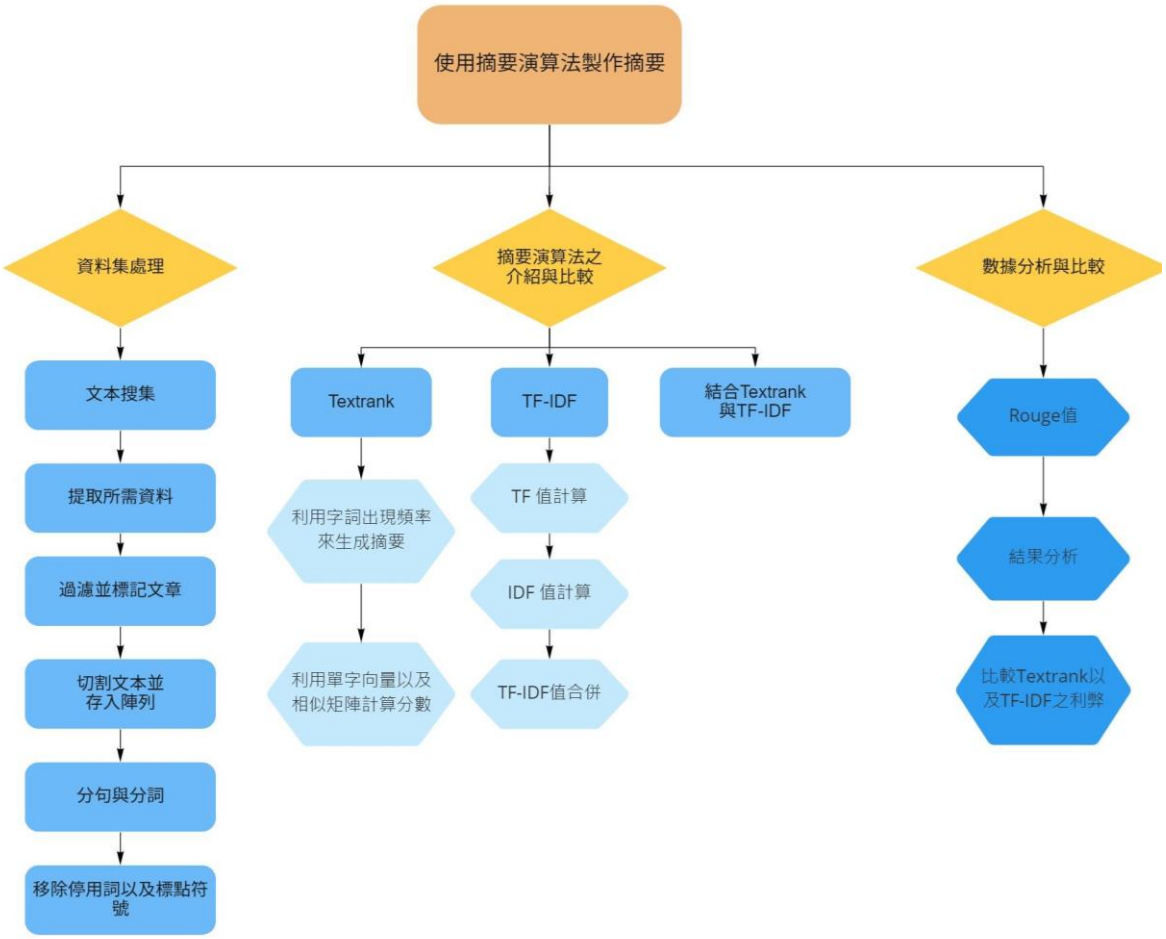
(一)、自然語言處理

自然語言處理(Natural Language Processing，簡稱 NLP) 是一門橫跨語言學和人工智慧，利用數學以及演算法讓機器理解並處理人類自然語言的學問。在日常生活中我們也能看到 NLP 廣泛的應用，包含了語言翻譯、Google 的搜尋演算法、智慧語音系統等等。

自動擷取摘要(Automatic Text Summarization)主要可以分為兩種，第一種為萃取法(Extractive Method)，此演算法是用挑選的方式，從字句中選出最重要，即出現頻率最高的字句，組合成摘要。而第二種演算法則為抽象法(Abstractive Method)，這種演算法是先將整篇文章理解、處理後，再產生一段全新的摘要。而此篇報告將會注重於藉由文章語句的分析改善萃取式摘要法。

現今社會的萃取式摘要法主要是基於一個單字在文本中出現的頻率來決定其重要性，我們發現這種方法雖然能夠讀取到大部分的摘要，但常常會因為某些重要觀念沒有不斷被重複提起而被省略，例如總結或是有被改述過的句子，這些句子都有一定的重要性，但卻不會被納入在摘要當中，這樣會造成最終生產出的摘要不準確或者是不完整，除此之外，我們發現被省略的句子都會包含特定的關鍵字或者是語句結構。因此，我們要以文章結構之分析結果為操縱變因，調整包含重要關鍵字之句子的配分比例，進而改善萃取式摘要演算法。

(二)、使用摘要演算法製作摘要



圖二：摘要演算法製作流程圖

此研究的第一階段為結合不同的摘要法開發出最接近人為摘要的萃取式摘要法。

1. 資料集處理

(1) 文本搜集

此研究所使用的資料集為由 Maszhongming 在 Github 釋出的 CNN/DailyMail dataset 文本，其中含有大約 30 萬篇由 CNN 和 Daily Mail 釋出的新聞，作者已將此文本處理過，其中含有兩種版本的資料(BERT/RoBERTa)，而此研究使用的資料為 test_CNNDM_bert 的檔案。

(2) 提取所需資料

資料集中包含了九個不同的 keys，而我們提取出了在研究中會需要使用的兩個 keys（‘text’和‘summary’）之資料。text 為會用來輸入摘要演算法之文本，而 summary 為人為生產的對應摘要。

(3) 過濾並標記文章

在我們製作的摘要法當中，因摘要法有長度的限制，因此我們過濾出可以成功輸入摘要法的文章，總共 100 篇。過濾完成後，我們將這些文章以及對應的摘要標記，並且輸出儲存於.txt 檔供之後使用。

(4) 切割文本並存入陣列

在輸入資料集進入摘要演算法後，需要先進行文本的切割，將輸入的.txt 檔案按照上一步驟做的標記辨認出文章以及摘要，在辨認的過程中，逐一將它們分別存入兩個不同的陣列，如此一來，要使用資料時便可在兩陣列相同的位置提取出文本還有對應的摘要。

(5) 分句與分詞

將要使用的文章透過 NLTK 函式庫中的 sent_tokenize()函式將文章分成句子，再用 word_tokenize()函式將句子分成字詞，並且存入陣列。會需要進行此二步驟是因為摘要演算法所接受的最小單位為字詞，因為此演算法需要利用每個字詞在文本中出現的頻率來計算分數，最終挑選出分數最高的句子來組成摘要。

文本	I am Lisa. I like to eat yellow bananas. I also like yellow cars.
分句結果	['I am Lisa.', 'I like to eat yellow bananas.', 'I also like yellow cars.']
分詞結果	[['I', 'am', 'Lisa', '.'], ['I', 'like', 'to', 'eat', 'yellow', 'bananas',

	'.'], ['I', 'also', 'like', 'yellow', 'cars', '.']]

(6) 移除停用詞以及標點符號

停用詞為自然語言處理過程中出現頻率極高，卻沒有實質統計上的意義的字詞，目前 NLTK 函式庫包含了 179 個常見的停用詞，例如：or（或）、is（是）等字詞。在將文本放入摘要演算法前，通常會先將文本中的停用詞過濾處理，以免混淆最後分數的輸出以及摘要的生成。

標點符號是透過 string.punctuation 取得的 40 個常見的標點符號，此步驟也是為了之後生產摘要時，不要混淆並且影響其結果。

分詞結果	[['I', 'am', 'Lisa', '.'], ['I', 'like', 'to', 'eat', 'yellow', 'bananas', '.'], ['I', 'also', 'like', 'yellow', 'cars', '.']]
欲移除之停用詞	[['I', 'am', 'Lisa', '.'], ['I', 'like', 'to', 'eat', 'yellow', 'bananas', '.'], ['I', 'also', 'like', 'yellow', 'cars', '.']]
欲移除之標點符號	[['Lisa', '.'], ['like', 'eat', 'yellow', 'bananas', '.'], ['like', 'yellow', 'cars', '.']]
移除停用詞以及標點符號	[['Lisa'], ['like', 'eat', 'yellow', 'bananas'], ['like', 'yellow', 'cars']]

2. 摘要演算法之介紹與比較

我們總共製作了兩種摘要演算法，分別是 Textrank 以及 TF-IDF，而 Textrank 摘要演算法我們總共製作了兩個版本，兩者的差異在於計算分數時所使用的方法。

(1) Textrank 摘要演算法

Textrank 演算法簡單來說就是一種利用相似矩陣得出來的分數來選擇最終摘要的文句之演算法，但是其中使用到的概念較為複雜。我們使用了兩種不同方法來取得摘要，分別為出現頻率以及相似矩陣兩種方式，在利用相似矩陣製作摘要演算法之前，我們先嘗試了較為簡單、直觀的方法，也就是利用計算文字在文本出現的頻率之高低來判斷分數，接下來我們才更進一步利用相似矩陣製作摘要，以下為我們在介紹演算法時會用到的處理過後之文本。

[['Lisa'],
['like', 'eat', 'yellow', 'bananas'],
['like', 'yellow', 'cars']]

甲、利用字詞出現頻率來生成摘要

利用某一字詞出現的頻率來計算最終的分數是最為簡單同時又精準的方式，我們將剛才處理過的文本放入迴圈計算每個單字的出現率，並且存入字典，接下來，我們將字典中所有的值都除以最大出現頻率，將數值控制於 0 到 1 之間，方便計算與分析。

以下為最終生產出的字典：

keys	value
'bananas'	1/2 = 0.5
'cars'	1/2 = 0.5
'eat'	1/2 = 0.5

'like'	2/2 = 1.0
'Lisa'	1/2 = 0.5
'yellow'	2/2 = 1.0

以'bananas'以及'yellow'為例子，在文本中，因'bananas'只出現一次、'yellow'總共出現兩次，將頻率計算完畢後，最大出現頻率為 2，所以'bananas'的分數為 1/2 = 0.5，而'yellow'的分數為 2/2 = 1。

[['Lisa'],
['like', 'eat', 'yellow', 'bananas'],
['like', 'yellow', 'cars']]

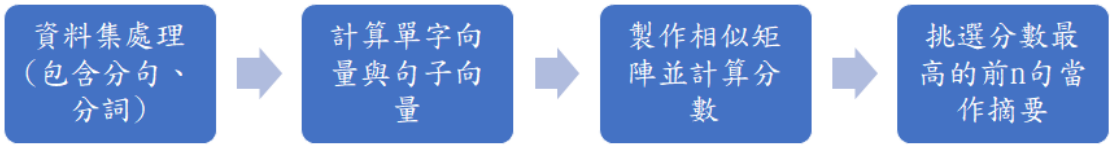
接下來，以句子為單位，將句子中有出現在頻率陣列中的字對應的頻率值相加，存入字典，得到最終的分數。

keys	value
'I am Lisa.'	**0.5**
'I like to eat yellow bananas.'	1.0 (like) + 0.5 (eat) + 1.0 (yellow) + 0.5 (bananas) = **3.0**
'I also like yellow cars.'	1.0 (like) + 1.0 (yellow) + 0.5 (cars) = **2.5**

在計算完句子分數後，我們便可以透過分數的高低決定最後生產出的摘要，在範例文本中，最後的摘要則為第二句('I like to eat yellow bananas.')，因為此句的分數是所有句子中的最高分。

乙、利用單字向量以及相似矩陣計算分數

此方法為現在最為常見的 Textrank 摘要演算法，步驟為下：



圖三：Textrank 摘要演算法流程

將字詞轉成詞向量的目的為把人類看得懂的文字轉換成電腦可以理解的向量。在製作向量的時候，以兩個句子為一單位，首先設定一個向量，而此向量的維度等於此二句中獨特且具有意義之詞的數量，接著在計算這兩句分別的向量，計算的方式就是將所有組成此句的單字向量合併並且構成最終的句子向量，我們以下文的範例文本進行解釋。

I am Lisa and I like cars. I like to eat yellow bananas. I also like yellow cars.

若以最後兩句為例子，則會生成維度為 5 的向量。

句子一	句子二	獨特且具有意義的詞
I like to eat yellow bananas.	I also like yellow cars.	'like', 'eat', 'yellow', 'bananas','cars' → 維度=5

而其中每一個字詞都代表不同的向量。

字詞	向量

like	[1,0,0,0,0]
eat	[0,1,0,0,0]
yellow	[0,0,1,0,0]
bananas	[0,0,0,1,0]
cars	[0,0,0,0,1]

最終的句子向量則是由單字向量組合起來。

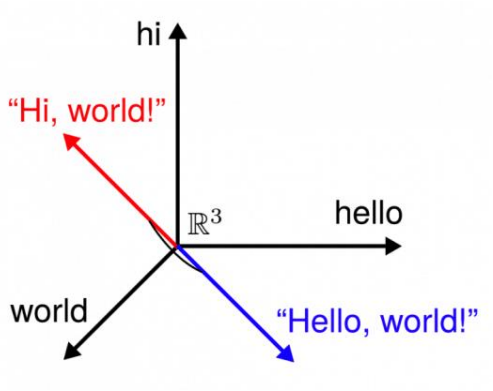
句子	句子向量
I like to eat yellow bananas.	[1,1,1,1,0]
I also like yellow cars.	[1,1,1,0,1]

接著藉由剛剛所得到的句子向量計算 cosine 相似度。

cosine 相似度 = cosine(向量一和向量二之間的夾角)

以剛剛的兩個句子為例，cosine 相似度為 0.59 ≈ 0.6

又例如在這張圖片當中，將句子向量用圖像的方式呈現在三維空間中，可以透過取兩向量('Hi, world!'和'Hello, world!')夾角的 cosine 值來得出其相似度。



圖四：相似度矩陣[八]

在計算完所有的相似度後，將它們放入陣列中，就可以得出最終的相似矩陣。

	I am Lisa.	I like to eat yellow bananas.	I also like yellow cars.
I am Lisa.	0.0	0.2582	0.5164
I like to eat yellow bananas.	0.2582	0.0	0.6
I also like yellow cars.	0.5164	0.6	0.0

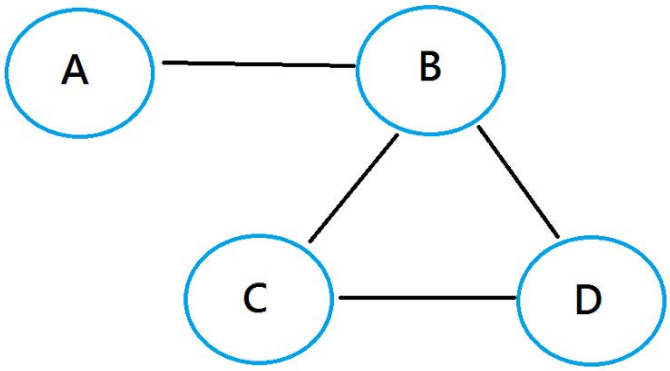
從相似度矩陣我們可以觀察出第三句和其他句子的相似度都較高，而第一句話則和其他句子的相關性最低。

為了方便解釋，我們現在使用一個較為簡單的相似度矩陣來帶入公式並且計算結果，在此相似矩陣中，我們假設相似程度只有相似(1)以及不相似(0)兩種狀況。

	A	B	C	D
A	0	1	0	0

B	1	0	1	1
C	0	1	0	1
D	0	1	1	0

由此表可以看出 A 句子和其他句子相似度較低，而 B 句則和其餘句子的相似度較高，藉由這樣的表格，我們可以畫出以下的關係圖。



圖五：相似度矩陣以及其關係圖[十一]

接下來我們便要把相似矩陣中的值帶入 Textrank 的公式，Textrank 的公式如下:

$$h(V_i)=(1-d)+d\bullet\sum_{V_j}\frac{w_{ji}}{\sum_{V_k\in Out(V_j)}w_{jk}}h(V_j)$$

圖六：Textrank 之公式[十一]

- 其中 Vi 為節點，也就是圖中每一節點
- h(Vi)為某特定節點的 TextRank 分數
- d 為阻尼係數，此係數為定值且介於 0 到 1 之間，常被設定成 0.85

- Wji 為節點之間相連的權重

我們使用此公式將各句迭代，藉此收斂到特定的數值，在此公式中，我們將阻尼係數設為 0.85。

句子	初始值	第一次迭代	第二次	第三次	第十次
A	1	0.433	0.674	0.504	0.586
B	1	1.850	1.248	1.606	1.462
C	1	0.858	1.039	0.945	0.985
D	1	0.858	1.039	0.945	0.985

經過十次迭代後，我們可以得知 B 句為最高分(1.462)，也就是說 B 句會被納入摘要當中。

(2) TF-IDF

TF-IDF (Term Frequency - Inverse Document Parsing) 為文章摘要中常見的加權技術，它利用詞頻 (Term Frequency) 和逆向文本頻率 (Inverse Document Parsing) 評估一個單字對於整篇文章的重要性，並給予分數。詞頻是指單一詞語在文章中出現的頻率，當一個詞語在文章中出現越多次時，代表它的重要性也越高，而逆向文本頻率則是一個詞語的普遍重要性，也就是它對於這篇文章的獨特性，若詞語在越少的文檔中出現，則其對文檔的區分能力就愈強，而當一個詞語在大部份文章中出現頻率都相當高時，這個詞語很可能是普遍使用的連接詞或是代名詞等，就算詞語出現頻率高依然不重要，因此在考慮詞頻的同時我們也要將逆向文本頻率納入考量來避免誤判頻率高但不獨特的詞語為重點。

TF-IDF 分數的計算方法為將詞頻的分數乘以逆向文本頻率的分數。

TF 分數主要是在計算詞語在文章中出現的頻率，計算公式中分母為文章的總字數，而分子為該詞語出現的次數。

$$tf_i = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

圖七：TF 分數公式[十]

IDF 主要是計算詞語對此文章的稀有度，計算公式中分母為包含詞語的文件數目，分子為文章總數，而最後將結果取以 10 為底的對數讓數值比較平滑。

$$idf_i = \log \frac{|D|}{|\{j: t_i \in d_j\}|}$$

圖八：IDF 分數公式[十]

如果一個詞出現在給定語料庫的每個文檔中，則該詞的 idf = 0，這也就代表這些詞語並不獨特，並無法代表文章重點。而與文檔相關的詞應該頻繁但獨特，不應出現在每篇文章中(即停用詞)。

(3) 結合 Textrank 以及 TF-IDF 產生新的摘要演算法

在製作完以上兩種演算法後，我們將 Textrank 演算法以及 TF-IDF 演算法合併，將 Textrank 演算法中計算單字向量的程式碼取代成 TF-IDF 計算單字頻率，最後比較結果。

我們在製作並且輸出摘要時，我們訂定摘要的長度為最接近人為提供之摘要，如此一來便能將長度以及句子的數量設為控制變因，並且凸顯出摘要演算法的優缺點以及成效。

3. 數據分析與比較

(1) Rouge 值

將文章摘要產生出來後，我們使用 Rouge 來計算演算法生成出的摘要和文本中人為生產的摘要之相似度。

Rouge 值中有三種主要的計算方式，分別為 Rouge-1, Rouge-2, 以及 Rouge-L，這三種類別的差別為計算單位的長度，Rouge-1 以一元語法(Unigram)為單位，Rouge-2 以二元語法(Bigram)為單位，而 Rouge-L 則是計算人為生產之摘要和演算法生產之摘要的最長共同子序列(Longest Common Subsequence)，而每一類別又再分成三種(Recall, Precision, F1-score)，這三種分數越高，代表摘要演算法的功效越好。

Recall (r) = 兩摘要重複的字之數量/人為摘要之總字數

Precision (p) = 兩摘要重複的字之數量/摘要演算法之摘要的總字數

F1-score (f) = 2*[(precision*recall)/(precision + recall)]

(2) 結果分析

以下表格為在製作完所有演算摘要法後，我們將第一步驟處理好的所有文本輸入演算法，並且使用 Rouge 值測試其結果取得之平均值。

	Gensim summarization	Textrank (字詞出現頻率)	Textrank (相似矩陣)	TF-IDF summarization	Textrank + TF-IDF
rouge-1_r	0.599	0.3599	0.1778	0.1953	0.3241
rouge-1_p	0.2697	0.3203	0.4204	0.196	0.3036
rouge-1_f	0.355	0.3256	0.2	0.1797	0.2993
rouge-2_r	0.2806	0.1288	0.052	0.0332	0.1014
rouge-2_p	0.0987	0.1141	0.0606	0.0305	0.0918
rouge-2_f	0.1372	0.1153	0.0492	0.0281	0.0914
rouge-L_f	0.3203	0.2754	0.1671	0.1674	0.2575

接下來的分析將會以 Rouge-1_f 的欄位為主，原因為此欄位結合了 precision 以及 recall 二值，最能夠精準並且全方面的表現出摘要演算法的成效。

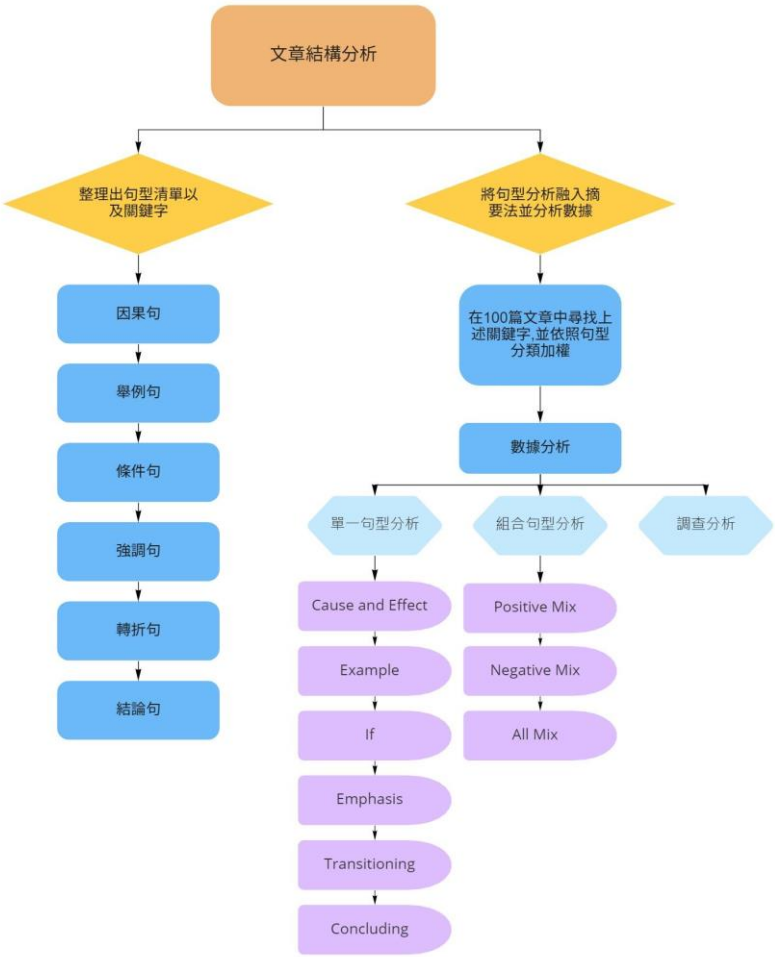
首先，Gensim summarization 的分數遠大於其餘四種摘要演算法。Gensim summarization 為現今最為常見且最精準的萃取式演算法，它是一種基於 Textrank，並且經過長期測試、修正後得出的演算法，其中使用了詞遷入(word2vec)的技巧，使它最終產生出的摘要準確度極高。

再者，利用字詞出現頻率所編寫之 Textrank 演算法的分數大於利用相似矩陣所算出的分數，經過閱讀文本並且分析兩種摘要演算法所生成的摘要，我們得出了一些結論，我們認為這樣的結果和新聞文章本身的特點很有關聯。我們所使用的文本是從 CNN/DailyMail 發布的新聞所提取，而此類新聞的特點是它們通常都圍繞著某一特定主題講述，而此特定主題或詞(人物、事件等)便會非常頻繁地出現在文章中，不僅如此，為了讓閱讀者容易理解，新聞並不會使用過於花俏或特別修飾過的字詞、手法，因此，只要文章是著重於同一重點，此特定單字便會很常在文本中重複出現，這樣的特點使得運用頻率來計算分數的摘要法獲得了較高的分數，而運用相似矩陣的摘要演算法並非如此。

接著，我們觀察到 TF-IDF 摘要演算法的分數更是意外的低。產生此結果的原因有很多種，不過經過觀察其輸出資料，我們推測出了兩個可能的情形。第一，這些新聞文本的用詞、結構、主題等，因為文本當中的新聞大多為當時美國、英國熱門的話題，重複性不免上升，這樣的狀況讓重要的關鍵字很難被提取，因為 TF-IDF 注重的不只是字詞出現頻率的高低，更重要的是字詞的獨特性。第二，就算 TF-IDF 成功找到不同文章主要著重的主題，此演算法也無法成功地將最終極少的摘要句提取出來，因 TF-IDF 演算法會一次將 100 篇文章判讀並且分析字詞出現頻率以及獨特性，範圍廣大，即便找出每篇文章最重要的關鍵字，也會因為其重要的細節相較於其他常見的詞彙沒有那麼常出現，或沒有那麼獨特，所以沒有被成功辨認並且納入最終的摘要。

最後，當我們將 Textrank 以及 TF-IDF 合併製成摘要演算法，分數仍然無法超過我們利用字詞頻率所製作的 Textrank 摘要演算法，這更可以凸顯出此新聞文本的特性以及為何 TF-IDF 不適合用來當作提取此類文本摘要的演算法之原因。

(三)、文章結構分析



圖九：文章結構分析流程圖

此研究的第二階段為融合上一步驟生產出的摘要演算法和此步驟得出的文章結構分析，並且判斷加入哪一些句型關係可以讓摘要法更接近人為生產的摘要。

1. 整理出句型清單以及關鍵字

我們透過閱讀分析 CNN、BBC 等多大新聞平台的句型結構以及國高中英文課本教材內容整理出了一系列常見的關鍵字以及它所屬於的句型種類，而根據不同句型的出現頻率以及其重要性，此研究將會注重於 6 種基本句型 – 因果句、舉例句、條件句、結論句、強調句、轉折句，剩下的句型有待未來開發研究。

舉例	目的	比較	強調
According to As follows Example Instance In particular In substantiation Just as Particularly Specifically Such as To demonstrate To illustrate To substantiate	In order to To	Be to parallel in Comparatively Correspond to Correspondingly Likewise Like Respectively Similar to Similarly Than To have… in common	Apparently Certainly Clearly Especially Indeed Interestingly Obviously Oddly Enough Significantly Still Surely Undoubtedly
	條件		
	If		

因果	轉折	順序	
Accordingly As a consequence As a result Be responsible for Because Cause Consequently Contribute to Due to For this reason Have an effect on Hence	Although But By contrast Conversely Different from However In contrast Instead Rather than Nevertheless On the contrary On the other hand	After Afterward As soon as At last At the same time Before During Earlier Eventually Finally First For the time being	Later Meantime Meanwhile Next Now Nowadays Recently Second Since Soon Subsequently Temporarily

Lead to Now that Owing to Result in Since So Therefore Thus	Otherwise Unlike Unfortunately Whereas While Yet	From now on Immediately In the end In the first place Last Lastly	Then Third To begin with Ultimately When While

解釋	遞進	讓步	結論
As a matter of fact In fact In other words In similar terms Indeed Namely That is	Additionally Again Also And Besides Furthermore In addition Moreover Too What is more	Admittedly After all Although Despite Even if Even though In spite of Though	Briefly In a nutshell In a word In brief In conclusion In short On the whole Summing up To conclude To put it briefly To sum up To summarize

2. 將句型分析融入摘要法並分析數據

(1) 在 100 篇文章中尋找上述關鍵字，並依照句型分類加權，加權分數多寡由多次實驗找出效果最佳之加權值作為依據。

選入摘要所需之最低平均分數	每句平均分數

7.6644	2.7709

我們根據選入正式摘要所需之平均最低分數、文本中每一個句子經過演算法所得出的平均分數、以及含特殊句型之句子的平均分數來推斷加權分數的大小。

除此之外，我們提供了每種特殊句型在加權之前的最高分數，以證明雖然平均分數經過加權後仍無法達到成為正式摘要的標準分數，必定會有特殊句型在經過處理後被選入正式摘要。

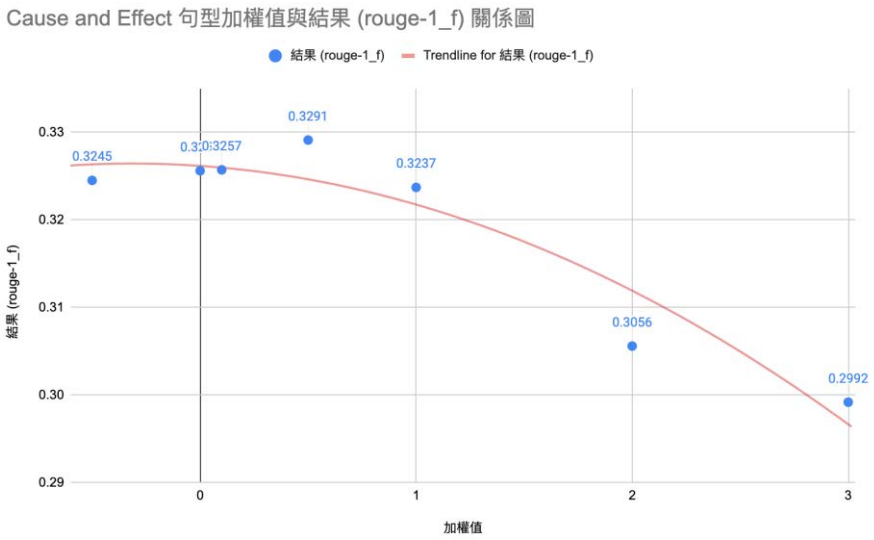
特殊句型	含此句型之句子的平均分數(未加權)	含此句型之句子的最高分數(未加權且小於選入摘要所需之最低平均分數)
Cause and effect	3.6789	7.5714
	選入摘要所需之最低平均分數與含此句型之句子的最高分數之差為 0.093，因此我們使用的第一個加權分數為 +0.1，進而確保此動作會對最終的分數造成影響。	
Example	3.9993	7.2857
	選入摘要所需之最低平均分數與含此句型之句子的最高分數之差為 0.3787，因此我們使用的第一個加權分數為 +0.4，進而確保此動作會對最終的分數造成影響。	
If	3.2184	7.625
	選入摘要所需之最低平均分數與含此句型之句子的最高分數之差為 0.0394，因此我們使用的第一個加權分數為 +0.04，進而確保此動作會對最終的分數造成影響。	
Concluding	2.9568	4.4

	選入摘要所需之最低平均分數與含此句型之句子的最高分數之差為 3.6244，因此我們使用的第一個加權分數為+4，進而確保此動作會對最終的分數造成影響。 因為用 100 筆資料測試時，我們發現有 Concluding 句型的句子數量不多，因此為了求得更精確的數據，我們使用了 500 筆資料輸入測試。	
Emphasis	3.0849	7.5455
	選入摘要所需之最低平均分數與含此句型之句子的最高分數之差為 0.1189，因此我們使用的第一個加權分數為 +0.2，進而確保此動作會對最終的分數造成影響。	
Transition	3.0452	7.625
	選入摘要所需之最低平均分數與含此句型之句子的最高分數之差為 0.0394，因此我們使用的第一個加權分數為 +0.1，進而確保此動作會對最終的分數造成影響。	

(2) 數據分析

甲、單一句型分析

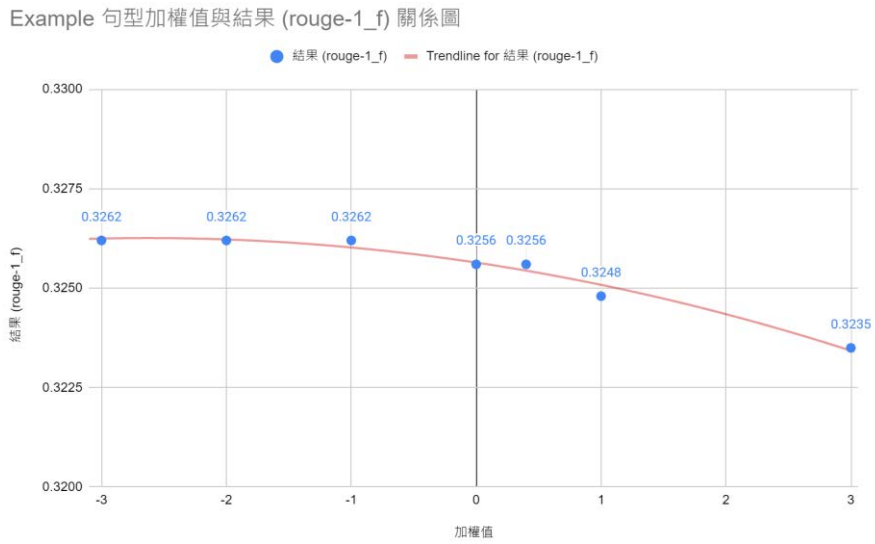
我們的測試方式為先測試上述表格推測的第一個加權分數與另一個負數分數，接著依照測試結果的回饋來決定第二個加權分數的加權值與測試方向（若負數結果較佳就將加權值減少，反之，若正數結果較佳，則加權值增加。），經過測試數筆資料，我們最終得出趨勢圖。



圖十：Cause and Effect 句型加權值之結果

加權值	結果 (rouge-1_f)
-0.5	0.3245
0	0.3256
0.1	0.3257
0.5	0.3291
1	0.3237
2	0.3056
3	0.2992

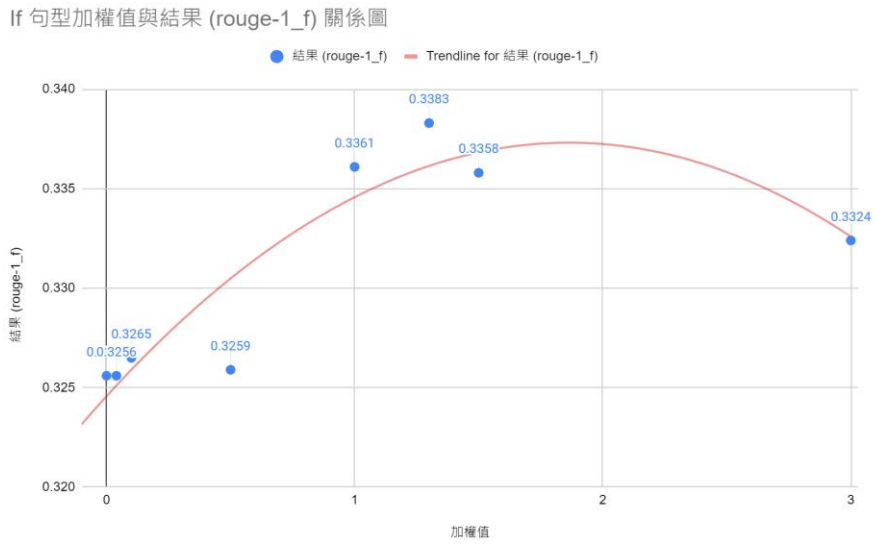
在 Cause and Effect 句型中，我們測出**最佳的加權值是+0.5**，而在加權更多或是當加權值為負數時，結果都會使 rouge-1_f 值降低，我們認為是因為因果關係在新聞中有一定的重要性，因為新聞常常需要使用因果來解釋事情的來龍去脈，因此有因果關係的句型相對其他來說較為重要，因此加分較為恰當，然而當分數加太多時會導致摘要中因果關係句的比例太高，因此摘入相對不重要的因果句，此方法可以讓本來就將重要的因果句子加分，並更可能進入被納入摘要的句子之中。



圖十一：Example 句型加權值之結果

加權值	結果 (rouge-1_f)
-3	0.3262
-2	0.3262
-1	0.3262
0	0.3256
0.4	0.3256
1	0.3048
3	0.3235

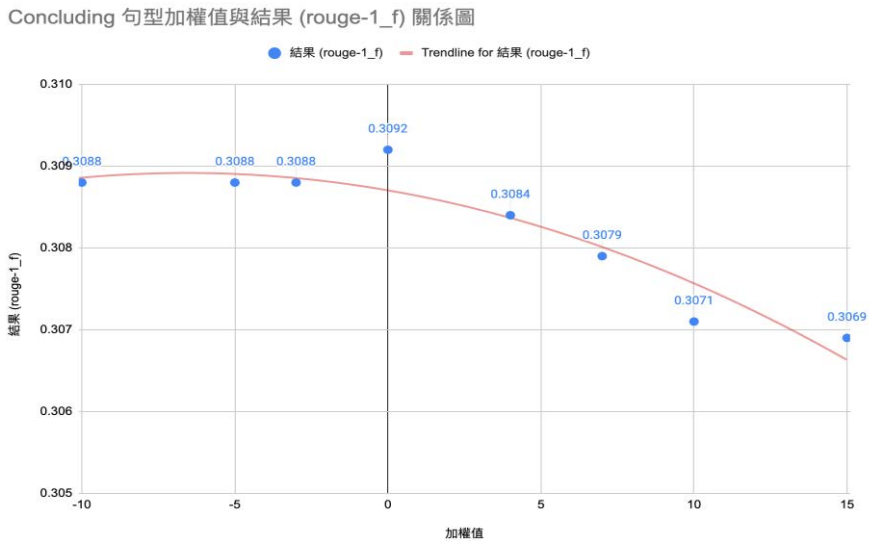
在 Example 句型中，我們測出**最佳的加權值為小於零**，而在加權加權值為正時，rouge-1_f 值有明顯降低，經過分析文本與摘要，我們了解到此現象是因為舉例關係在新聞中扮演較為細節與陪襯的角色，並不是文章最大的重點，可以在摘要中忽略，因此在句型中出現舉例時，我們對分數進行扣分來避免這些相對不重要的句型被納入摘要中，進而增進摘要的準確度、閃避這些太小的細節部分。



圖十二：If 句型加權值之結果

加權值	結果 (rouge-1_f)
0	0.3256
0.04	0.3256
0.1	0.3265
0.5	0.3259
1	0.3361
1.3	0.3383
1.5	0.3358
3	0.3324

在 If 句型中，我們測出**最佳的加權值為 1.3**，並且讓結果有明顯的進步，而在加權更多或更少時，rouge-1_f 值都有明顯下降，我們推測其原因為新聞當中包含推測未來走向、預估、以及預測未來等資料，這些資料極為重要，但是在我們的觀察中，他們在加權之前的分數較低，因此，在加權值較重時成效更為明顯。

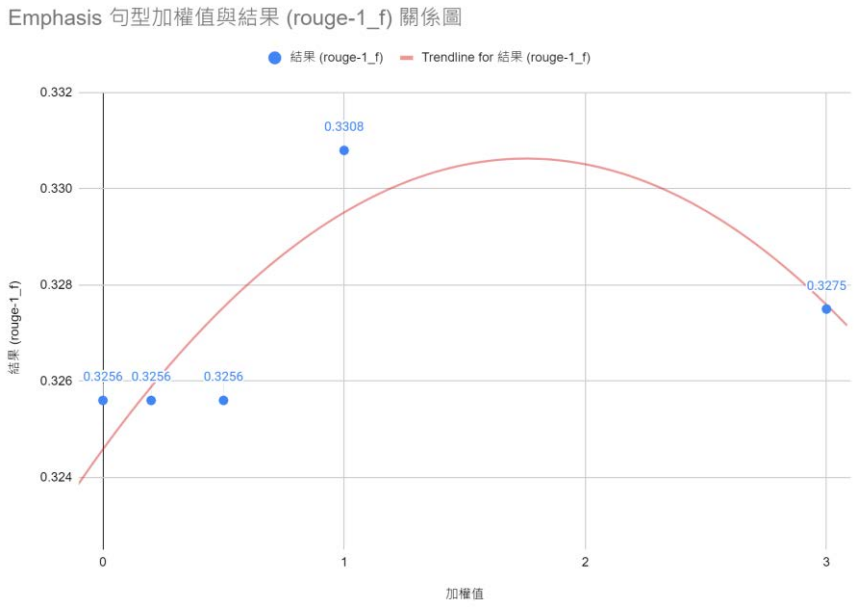


圖十三：Concluding 句型加權值之結果

加權值	結果 (rouge-1_f)
-10	0.3088
-5	0.3088
-3	0.3088
0	0.3092
4	0.3084
7	0.3079
10	0.3071
15	0.3069

在 Concluding 句型中，我們測試出的**最佳的加權值為 0 分**，而因為在測試 100 筆資料時，含有 Concluding 句型的句子數量很少，**所以我們將測試資料增加至 500 筆**，此句型的數量也大幅增加。經過觀察，文章中的 Concluding 句型共有兩種，第一種為統整文章重點的 Concluding 句型，第二種則是為整理文章中提到的一些調查、數據的句子，這些在新聞當中通常都是為了支持主旨而提出的，因此均為細節部分，若列入摘要則會使得摘要更不準確。在我們未加權時，第一種的 Concluding 句型已經被融入摘要當中，而第二種沒有被納入摘要，當我們將加權值提高時，會將第二種講解細節的句子被放入摘要當中，進而造成最終 Rouge 值降低。反之，在我們將加權值減少時，會導

致原本的第一種重要的 Concluding 句型從原本的摘要中被移除，使得最後摘要的準確度降低。

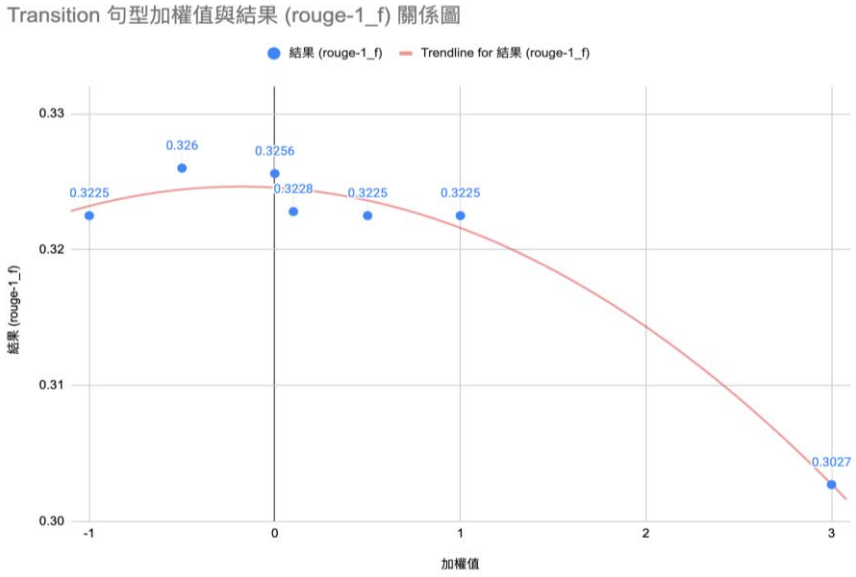


圖十四：Emphasis 句型加權值之結果

加權值	結果（rouge-1_f）
0	0.3256
0.2	0.3256
0.5	0.3256
1	0.3308
3	0.3275

在 emphasis 句型中，我們測試的**最佳的加權值為 1**，其對 rouge-1_f 的影響相較其他句型較為顯著，然而加分過多時 rouge-1_f 值又下降了，這是因 emphasis 在新聞中是強調重點的語句，十分重要，因此此句型出現在摘要中的頻率也特別高，所以給予相當高的加權可以幫助找出最精確的摘要，而加分過多時會倒置所有含有 emphasis 句型的句子都被納入摘要，然而並不是所有強調句都重要到需要納入摘要，有時候也可能是重複強調，因

此我們給予一個相對高的加權值能讓模型最有效益。



圖十五：Transition 句型加權值之結果

加權值	結果 (rouge-1_f)
-1	0.3225
-0.5	0.3262
0	0.3256
0.1	0.3262
0.5	0.3256

在 transitioning 句型中，我們測試的**最佳的加權值為-0.5**，然而其對 roug-1_f 的影響也不是特別顯著，這是因為 transitiong 在新聞中十分常見，在敘述事件時常常都會用到銜接的語詞，銜接詞語對內容無幫助也跟內容無關，所以銜接詞語無關此句是否重要與是否應該納入摘要中，因此 transition 句型對判斷摘要

1	0.3256	沒有很重要，些微扣分或保持原樣是最恰當的模型。
3	0.3048	

以下的表格為最終的整理表格，為了呈現公平的比較結果，我們還是以 Concluding 句型用 100 篇的資料測試的結果來呈現。

	Cause and Effect	Example	If	Concluding	Emphasis	Transition
最佳加權分數	+0.5	-1	+1.3	0	+1	-0.5
rouge_f 值	0.3291	0.3262	0.3383	0.3256	0.3308	0.3262

乙、組合句型分析

在單獨句型的摘要測試完畢後，我們將不同句型組合在一起，運用每種句型最適當的加權分數(也就是可以獲得最高 Rouge 值的加權分數)，得出最終結果。以下的第一個表格為不同句型所屬於的摘要類型，第二個表格為我們選擇的三種組合方式以及未加權的摘要法之數據比較，其中正加權摘要法包含的句型為 Cause and effect，Emphasis，以及 If，因為他們的最佳加權分數均為正，而負加權摘要法含有的句型為 Example 以及 Transition，因為他們的最佳加權分數均為負，最終的綜合組合為六種句型全部綜合在一起。

	Cause and Effect	Example	If	Concluding	Emphasis	Transition
分類	正加權組合	負加權組合	正加權組合	無 (不加權為最佳狀態)	正加權組合	負加權組合

組合分類	未加權	正加權組合	負加權組合	綜合組合
rouge_f 值	0.3256	0.3379	0.3293	0.3361

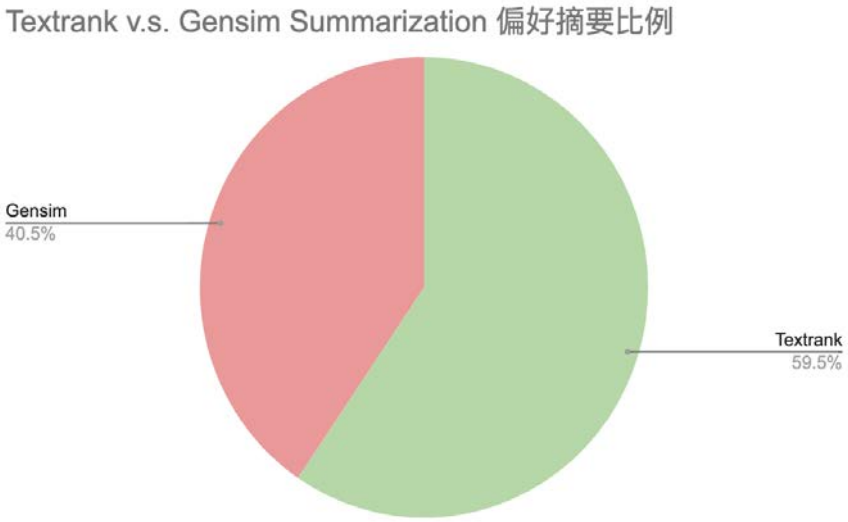
經過結果比較，全部加權後的分數都較未加權者高，成功的達到目的也證明我們的假設，而其中表現最佳的正加權組合相較未加權版本提高了 4%的準確度。接下來我們更進一步分析三種組合的數據，首先，正加權組合高於負加權組合以及綜合組合的原因我們推論有數種，首先，正加權組合高於負加權組合是因為正加權組合所影響的變因有 3 個，而負加權則只影響 2 個變因，除此之外，負加權組合中句子的最佳加權分數和 Rouge 值都低於正加權句子，所以雖然負加權組合的結果有高於未加權的摘要法，但是它對於產生精確摘要相較於正加權組合的影響較小。再來，正加權組合高於綜合加權組合的原因我們推測是因為當一個句子同時包含正加權句型以及負加權句型時，負加權句型會使得正加權句型對於句子的影響下降，導致句子最終並未被納入摘要中，此推測的假設為正加權句型的加分在一個句子當中比負加權句型的扣分來的更為重要，而這個假設也被我們用數據證明存在，因為在上一部份測試資料時，最佳加權值所得出的 Rouge 值增加得越多，代表此句型在文章當中的重要性更高，而正加權句型的三個 Rouge 值增加的量均比負加權句型的兩個 Rouge 值增加的量還要多，代表正加權句型在文章中比負加權句型更具影響力。最後，因為負加權組合的變因數量和影響程度均比綜合組合還要低，所以負加權組合比綜合組合的分數略低一些。

(3) 調查分析

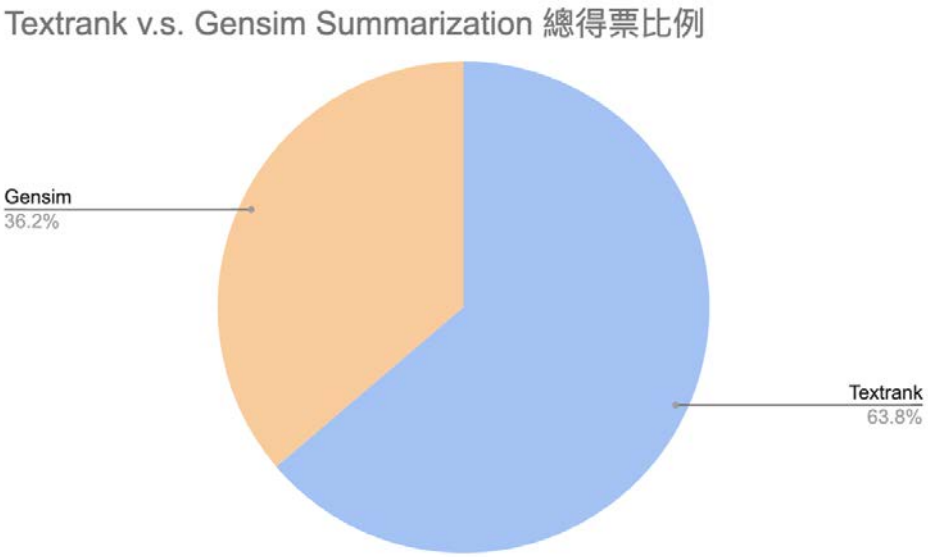
為了達成我們最終目標將此次改進產出的摘要演算法成果寫成網站提供給大眾使用，我們認為調查人們對於我們現階段所生產出的摘要之看法非常重要，而機器（rouge 值）所認為精準的摘要並不等於人們最喜歡的摘要模式。因此，我們請了七位不同年紀層、對新聞摘要有不同需求的受訪者閱讀 100 篇我們摘要演算法(使用 Rouge 值最佳的正加權組合)產出的摘要並且將它和現在社會中最常見、Rouge 值也最高的 Gensim summarization 所產生出的 100 篇摘要進行比較以及挑選，在不知道摘要是哪個模型產生出來的情況下選出一個他們覺得較為符合需求以及最能涵蓋整篇文章的主旨之摘要，以下我們將進行此次調查所得出的數據之分析。

在 100 篇文章中，扣除 Textrank 和 Gensim Summarization 產出完全相同摘要的文章後，共有 79 篇，而其中，受訪者偏好 Textrank 摘要的有 47 篇，偏好 Gensim

Summarization 摘的有 32 篇，高出 15 篇(19%)。而在投票數比例上(計算方式為將 Gensim Summarization 和 Textrank 的得票總數分別除以 7，然後再將 79 篇文章的兩種比例分別的總和除以 79，以此方式得出最終的比例)，63.8%的時候受訪者較偏好 Textrank 所產出的摘要，其餘 36.2%的時候受訪者較偏好 Gensim Summarization，結果顯示 Textrank 在經我們修改後人們偏好度較 Gensim Summarization 高出 27.6%。我們以以下兩張圓餅圖（圖十六、圖十七）來呈現最終結果。



圖十六：Textrank v.s. Gensim Summarization 偏好摘要比例



圖十七：Textrank v.s. Gensim Summarization 總得票比例

經過仔細分析兩種摘要法產生的摘要之差異及人類閱讀文章時的習慣，我們認為 Textrank 演算法相較 Gensim Summarization 更受大家喜愛是因為它更能模擬人類閱讀時的思考方式。在我們閱讀文章時，為了更加有條理的理解文章，腦中會自動分析文章的句型結構，而在閱讀有特殊句型節構的字句像是因果句、條件句等等時，我們不僅能更有條理的理解文章大意，並在腦中清晰的羅列出文章提及事件的前後順序、因果關係，這些句子在直述句中也會在我們腦中留下更深的印象，讓我們摘要時更容易使用他們。若兩句話之間有特殊關係卻沒有用連接詞連接在一起，閱讀長篇文章的我們一樣會很難分辨出納些句子的重要性，這樣的情況更加地顯示出了替不同句型結構加入不同權重的重要性。除此之外，根據過去閱讀不同種類文本的經驗，我們能夠直觀的判定出文章中哪些句型較為可能為文章的重點，例如，我們使用的新聞文本這一類的文章通常都是在講述某事件的原委，而一個事件是由原因以及結果所組成，因此就算一開始閱讀文本時並沒有特別注意到這些含有關鍵字的特殊句型，在選取摘要時，我們也會偏向選擇更有邏輯性的文句，以下我們使用一個例子來解釋上述分析。

Textrank	Gensim Summarization
A Turkish court imposed the blocks because images of the deadly siege were being shared on social media and deeply upset the wife and children of Mehmet Selim Kiraz, the hostage who was killed.	A Turkish court has blocked access to Twitter and youtube after they refused a request to remove pictures of prosecutor Mehmet Selim Kiraz held during an armed siege last week grief: the family of Mehmet Selim Kiraz grieve over his coffin during his funeral at eyup sultan mosque in Istanbul, turkey.

在此以上摘要當中，雖然兩種摘要法想表達的意思十分相近，但 7 位受訪者依舊都把票投給了 Textrank 摘要法，這是因為 Textrank 摘要法透過關鍵字 because 呈現出了新聞中因果關係的文章邏輯，使得讀者更好瞭解文章脈絡。在以上摘要中，兩者皆是在講述土耳其政府禁止社群媒體得使用因為武裝份子拒絕將他們羈押一名檢察官的照片從社群網站上刪除（藍字體部份），在 Genism Summarization 中，當陳述完這個事件後，摘要銜接了另一個較不相關的資訊：在檢察官舉行於伊斯坦堡一處教堂的葬禮上，他的家人很悲傷的站在棺材旁。而在 Textrank 當中，因為我們有分析文章句型結構並判斷出因果關係句進行加權，因此產生出的摘要顯現出文章的因果關係，在陳述完同一個事件之後，Textrank 摘要演算法講述了土耳其政府採取此措施的原

因：網路上流傳的檢察官照片造成他的小孩與妻子的悲憤因為該名檢察官最終作為人質遭到殺害。這樣的分析對讀者來說更接近文章想要表達的重點，因此更受大眾喜愛。

參、研究結果與討論

此科展研究主題為摘要演算法和文章句子結構的關聯性，此研究與其他研究摘要法的科展主題不同因為我們不僅比較了不同摘要法的利與弊，並且製作出了一套我們自己的摘要演算法，我們還分析不同的句型結構，最後將這個新的變因融於演算法之中並觀察其對生產出來之摘要的影響，最後，我們也統整了人們對於我們製作出來的摘要演算法的看法與偏好程度。

過程主要分為兩大部分，第一部分為研究比較不同摘要法的效能並分析結果，第二部分則是藉由分析文章中不同的語句結構根據不同的結構加權，測試有哪些語句可以成功地讓摘要法產生出更為精準的摘要，最後將成品藉由調查的方式判斷大眾喜好的摘要法與其原因。

在第一步驟中，我們首先嘗試製作 Textrank 以及 TF-IDF 摘要法，並分析其數據以及輸出的摘要成果，最後我們將兩者結合，利用 TF-IDF 計算出詞頻，在使用 Textrank 的手法將最高分的句子挑選出來。在三種摘要法都製作完成後，我們花了許多時間比較並且分析不同摘要法的利與弊和探討我們所使用的文本和結果的關聯性。在我們仔細的分析文本的性質後，我們了解到每篇新聞的格式均非常相近，不論是結構還是用詞都十分直白、不會採用難以理解的手法。根據這樣的推斷，我們推測 TF-IDF 分數較低是因為此演算法無法成功辨別到每篇新聞的關鍵字，而且它挑選出來的句子和主題沒有相關性，我們認為這樣的現象和每篇新聞的相似結構以及用詞很有關聯。除此之外，最終尋找字詞頻率的 Textrank 演算法能夠脫穎而出的主要原因也是因為新聞為了讓讀者容易理解新聞內容，在用詞上會避免拐彎抹角的手法，讓整篇文章的重點很容易利用特殊字詞之頻率的方法凸顯出來，並且被選為最終輸出的摘要。

在第二步驟中，我們將語句分析結合第一部分的模型，希望藉此增進摘要準確率，在詳讀各大主流新聞如 CNN、BBC、New York Times、Fox News 等並結合英語課本與文法書籍中所提到常見的句型結構後，我們整理歸納出不同句型的關鍵字，在搜索到相符關鍵字時將句子歸類到它所屬的句型中，並進行加權分數的動作以實驗何種句型能對摘要模型起到提高準確率

之作用。其中，假設語句、強調語句、與因果關係語句為新聞中重點最常出現的地方，有時卻沒出現在摘要中，因此我們利用加權分數將這些句子納入摘要，增加準確性，而相反的，舉例語句中的資訊在新聞中較為細小而不重要、出現在摘要中的比例極低，因此在加權分數為負數時能較好避免摘要選入不重要的細節，接而提升摘要模型的準確率，另外，總結語句不加權為最佳狀態，因為這樣可以使總結文章的語句被納入摘要而總結舉例中的資料等細小資訊的語句也會因為分數不夠高而不被納入摘要，除此之外，銜接語句因為與句子內容無關且無關乎句子重要性因此不論加分或減分都沒有對摘要準確度造成顯著的影響。

在最後一階段中，我們實際進行調查以證明我們所製作出的 Textrank 摘要演算法能夠產生出更接近人為思考方式的摘要，透過同學們的幫忙，我們成功藉由比較 Textrank 演算法以及 Gensim 演算法所生成出的摘要得出了最終的數據，我們發現我們所製作出的摘要演算法更受到人們喜愛，經過分析，原因為此摘要演算法更接近人類的思維，產生出一個較有邏輯以及能呈現文章脈絡的摘要，而這種摘要也和人們在閱讀文章時的思考過程很類似，因此即便 Textrank 模型在經過我們改良後提高的分數未超過 Gensim 模型，還是較受大眾喜愛的原因。

肆、結論與應用

本研究發現摘要演算法的成果和句型結構的不同有著關聯性，而我們只要能夠將不同句型成功的辨別出來，並且為它們加上適合它們的權重，我們便可以使得最終生產出來的摘要更為精準，除此之外，本研究也發現人們更偏好使用語句分析法所生產出的摘要，這樣的實驗主題不但新穎，也成功地產生出一套不但準確率高，也符合人們思維的摘要演算法。

最終得出的摘要法不但能供學生閱讀長篇文章時使用，在緊迫的學業壓力下能夠有效率地完成複習與閱讀，還能夠讓人們在現今資訊發達的時代有效率的篩選出對他們最有幫助的內容，而不是在資訊的茫茫大海中找不到自己所需要的資訊。

我們未來希望將我們改進的摘要法模型製作成網站推廣給更廣大的使用群，藉由簡單的複製貼上文章，使用者就能得到該篇新聞的主旨摘要，提供給他們更便利的新聞瀏覽體驗，並且避免使用者們浪費寶貴的時間在有著誘人標題卻冗長又缺乏重點的文章。

伍、參考文獻

一、台灣人工智慧學校. 斷開中文的鎖鍊！自然語言處理 (NLP)是什麼？ https://aiacademy.tw/what-is-nlp-natural-language-processing/

二、Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. https://aclanthology.org/W04-1013.pdf

三、Ganesan, Kavita. What is ROUGE and how it works for evaluation of summaries? https://kavita-ganesan.com/what-is-rouge-and-how-it-works-for-evaluation-of-summaries/#.YWrrdxpBxJ8

四、Hands-on implementation of TF-IDF from scratch in Python. https://www.google.com/url?q=https://analyticsindiamag.com/hands-on-implementation-of-tf-idf-from-scratch-in-python/&sa=D&source=docs&ust=1634658529924000&usg=AOvVaw1YwWmk8zyojb_BmVhb8jNC

五、Joshi, Prateek. An Introduction to Text Summarization using the TextRank Algorithm (with Python implementation) https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/

六、Maszhongming. MatchSum. https://github.com/maszhongming/MatchSum

七、Panchal, Akash. NLP — Text Summarization using NLTK: TF-IDF Algorithm. https://towardsdatascience.com/text-summarization-using-tf-idf-e64a0644ace3

八、Prabhakaran, Selva. Cosine Similarity - Understanding the math and how it works (with python codes) https://www.machinelearningplus.com/nlp/cosine-similarity/

九、Saxena, Sawan. Text Summarization in Python using Extractive method (including end-to-end implementation) https://medium.com/analytics-vidhya/text-summarization-in-python-using-extractive-method-including-end-to-end-implementation-2688b3fd1c8c

十、Xiuneng. TF-IDF 模型詳解. https://www.cnblogs.com/huliangwen/p/7420387.html

十一、Yueh-Lin Tsai. TextRank 演算法介紹 https://tan800630.medium.com/textrank-%E6%BC%94%E7%AE%97%E6%B3%95%E4%BB%8B%E7%B4%B9-e73b44679bce

【評語】190025

參賽者探討了不同摘要演算法的優劣並探討使用句型結構技術來增進摘要演算法正確率的有效性，實驗的目的在於探索和調整不同的加權值來達到最好的效果。雖然文本摘要已經有一些有名的現有模型可以直接套用，參賽者選擇提出語句分析來優化摘要選擇，評審們一致讚賞，並期待參賽者可以持續發展此研究，應用在實際的場域中（例如新聞平台。並持續探索與調整不同的加權值來達到最好的效果，並探索在不同類型與風格的文章中探索其適用性。