中華民國第 61 屆中小學科學展覽會

作品說明書

科　　別：電腦與資訊學科

組　　別：高中組

作品名稱： Level Up! 利用NLP技術提升英文寫作品質

關鍵詞：自然語言處理、深度學習、語言學習 (最多三個)

編號：

摘要

本研究開發了融合 BERT 與 word2vec 技術之輔助模組，能有效推薦不改變文意之替代詞，並將以字頻評估單字程度，同時保持文章的高度流暢並充分提升文章等級。

首先，以基於 Enchant 的 PyEnchant 及 spaCy 對文本實施拼字修正，降低錯字佔比，並有效的增加可用的訓練資料。我們利用以 New York Times 的文本訓練之 word2vec 模型計算替代詞候選與目標字的相似度，並配合 BERT 模型，實現對句中單字提供較優的替代詞之寫作升級模組。我們發現以目標字出發的 word2vec 結合以句子出發的 BERT，有效使詞的替代保留句子原意並維持文章流暢性。最後，以替代詞在不同等級文本出現的頻率，我們能判斷出單字的優劣並推薦給使用者。

本研究能使英語學習者更直覺的將他們所學過較艱深之單詞靈活應用在寫作，使英語寫作難度降低。未來在將功能擴展至片語、文法替換與升級後能更有效提升英文寫作教學效率與成果。

壹、 研究動機

一直以來，寫作在語文學習中佔一席之地，不擅長英文寫作的我們發現自己寫作品質低落但又不知道應該如何提升寫作品質。而隨著資訊時代的到來，學習語言的方式也日漸多元，我們於是運用 NLP 中較為成熟的 word2vec、新提出的 BERT 等技術，搭配 TOEFL 寫作資料庫以及紐約時報文章等，研究如何製作一個系統輔助使用者提升寫作品質。

貳、 研究目的

實驗一: 以紐約時報及托福寫作範文分析指定文本的單字出現頻率與作文等級的關聯

實驗二: 實作 word2vec 模型以計算詞向量

實驗三: 整合 word2vec 及 BERT 開發詞彙推薦模組

參、 研究設備及器材

一、軟體環境

Python3.7、Jupyter Notebook、TensorFlow、PyEnchant

二、訓練文本來源

TOEFL Writing Samples、New York Times

三、預訓練模型來源

Hugging Face 的 BERT 模型("bert-base-uncased")、spaCy ("en_core_web_sm")

肆、 研究過程與方法

一、研究流程圖

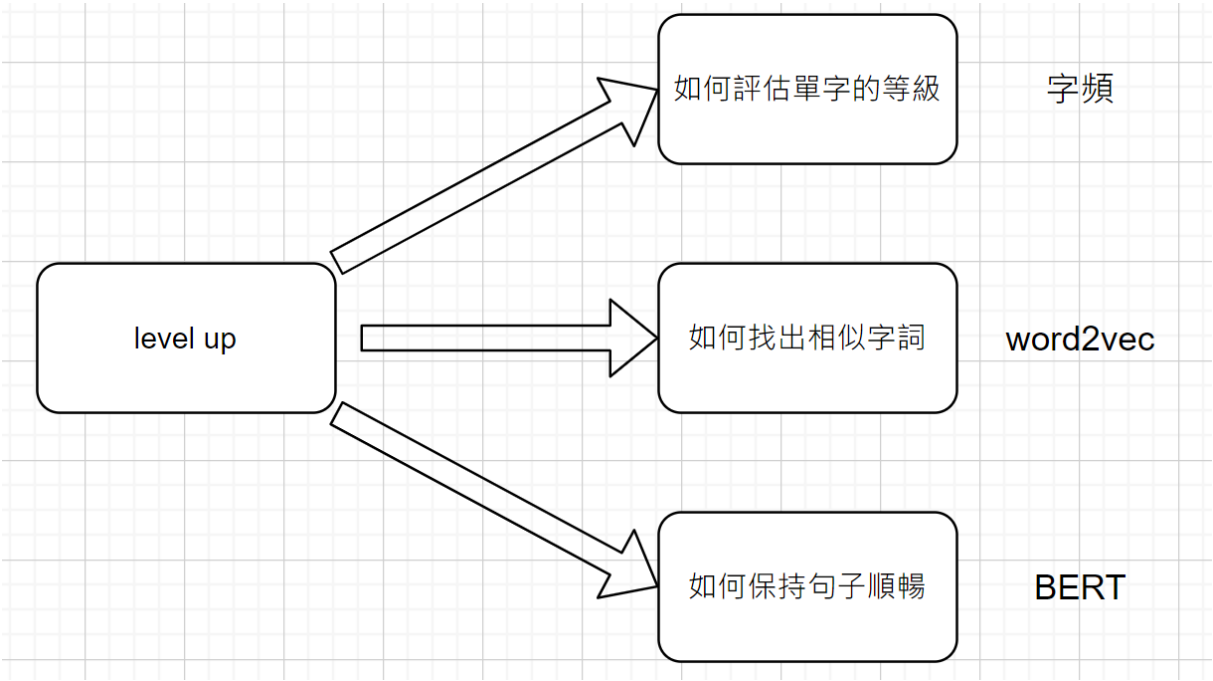

圖 4-1 研究流程圖

二、文獻探討

(一) 自然語言處理(Natural Language Processing)

1. 何謂自然語言

自然語言指的是不同於如程式語言或世界語等「人工」發明的語言，為自然地隨著文化發展而演化的語言，如漢語、日文、韓文及英文等均為自然語言。

2. 自然語言處理

自然語言處理(Natural Language Processing, 簡稱 NLP)是人工智慧和語言學的分支學科，研究處理與運用自然語言。其中自然語言處理的處理包括認知、理解與生成等子領域。

隨著技術的進展，自然語言處理早已脫離以「判斷輸入做出反應」之機制，而是利用許多不同的計算模型以統計機率方式使電腦找出語言特性，亦即規律性。

而隨著機器學習的進步，近年來自然語言處理逐漸變得熱門，尤其在硬體等級有了顯著提升的今日，自然語言處理又主要聚焦在非監督式學習與半監督式學習相關研究上。

(二)　Word Embedding

Word Embedding 將單詞的意思表示成在多為空間中的一個向量，解決了單詞只能以編號分別的情況。

1. 起源

在 NLP 發展初期，單詞是以 one-hot 向量表示，也就是說每個單詞有獨一無二的表示法，然而，常用的英文單詞數目高達 20000 個，這對電腦計算十分不友善。

2. word embedding

另一種方法是把單詞表示成一個向量。由下圖可以發現不但各軸都表示一個意思，字詞之間也有對應關係。

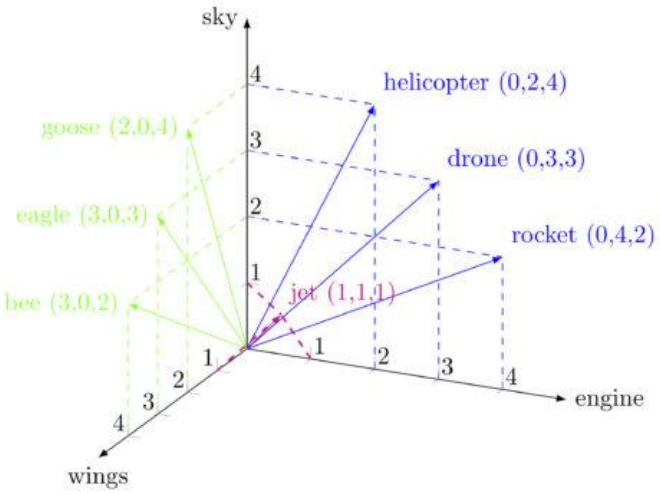

圖 4-2 word embedding 示意圖[5]

因為字詞是以意思作 embedding，相同意思的單詞應該有相似的向量。搜尋目標單詞在空間周邊的其他向量，以 cosine similarity 計算相似度。Cosine similarity 計算出 **a**,**b** 兩向量夾的 cosine 值，公式如下

$$\text{cos\_sim}(\mathbf{a},\mathbf{b})=\frac{\boldsymbol{a}\cdot\boldsymbol{b}}{|\boldsymbol{a}|\,|\boldsymbol{b}|}$$

然而，word embedding 的 word vector 難以計算，因此沒有被大量採用

3. word2vec

為了改善此情況，google 電腦科學家 Tomas Mikolov 發明了 word2vec 演算法，以非監督學習訓練出一個小型的神經網路，能有效地計算出 word vector(詞向量)，來表示一個單詞的意思。

Tomas 提出的模型包含一個 skip-gram 模型，該模型以一個單詞開始，以一個小型神經網路的計算出 word vector，並利用此 vector 算出周邊單字。基於”有相似鄰居就有相似性格”的假設進行無監督學習，可以有效預測其周邊的單詞。以神經網路計算出來的向量可以有效代表該單詞的意思

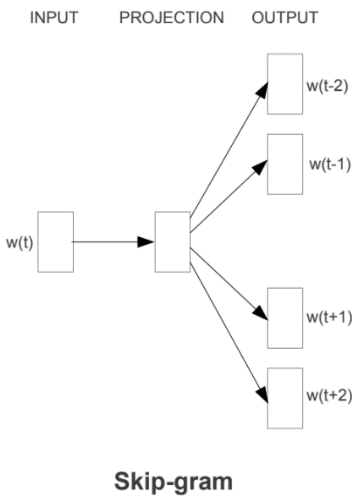

圖 4-3 skip-gram 示意圖[6]

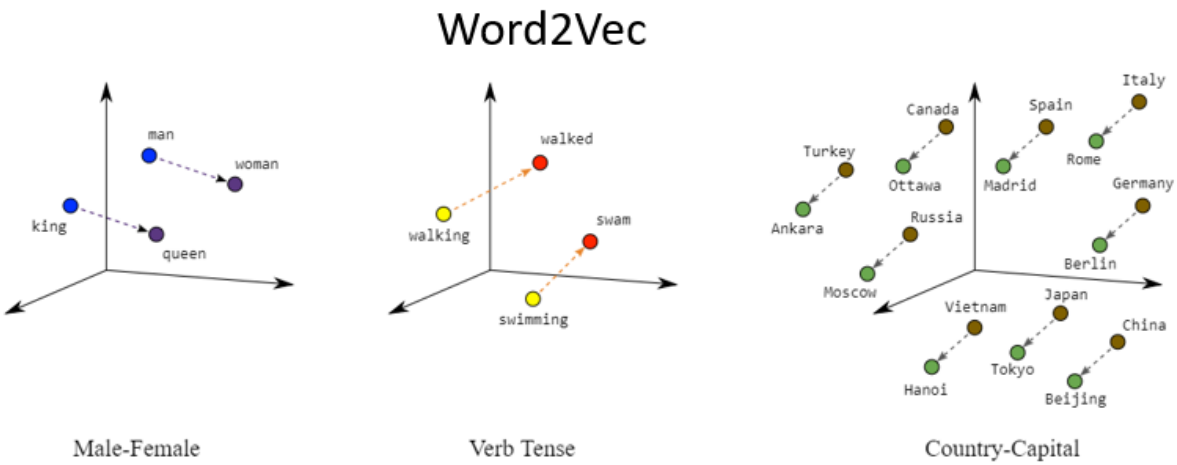

圖 4-4 word2vec 示意圖[7]

4. 缺點

word embedding 有兩個的缺點

- 一個單詞只有一種 word vector

  我們可以理解到，一個單詞並非只有一個意思。看看以下的範例

Constructors hired a crane to lift the heavy concrete block to the second floor.

Cranes migrate to the south to avoid the cold winter.

第一句的 *crane* 意思是吊車，而第二句的 *crane* 是鶴，如此一來，word vector 無法有效地表示所有意思

- 反義詞組

  看看一下的句子

The earl lives in a pretty mansion.

The earl lives in an ugly mansion.

反義詞組常被 word2vec 模型中一起出現。因為反義詞組常出現在相同的句子結構裡，因次在以 skip gram 模型訓練時，pretty 與 ugly 字擁有相似的 word vector，因此在作單詞替換時，常常會被替換成反義詞，而這是我們所不樂見的。

(三) BERT

2018 年，google 發表了 BERT (Bidirectional Encoder Representations from Transformers)模型，此模型解決了多數除了 language model 以外的問題

1. self-attention layer

BERT 利用了大量的 self-attention layer(自我注意力機制)。self-attention layer 能達成 RNN 的遞迴型作業，但是能平行計算該層將輸入變換成三個不同的 vector，q、k、v。將 q_1 內積 k_i 並乘上 v_i 就成為第一個輸出，其他以此類推。self-attention layer 結合了 seq2seq(或稱 encoder-decoder)的注意力機制，並與 bidirectional RNN 相似，在看完全全部資料下進行輸出

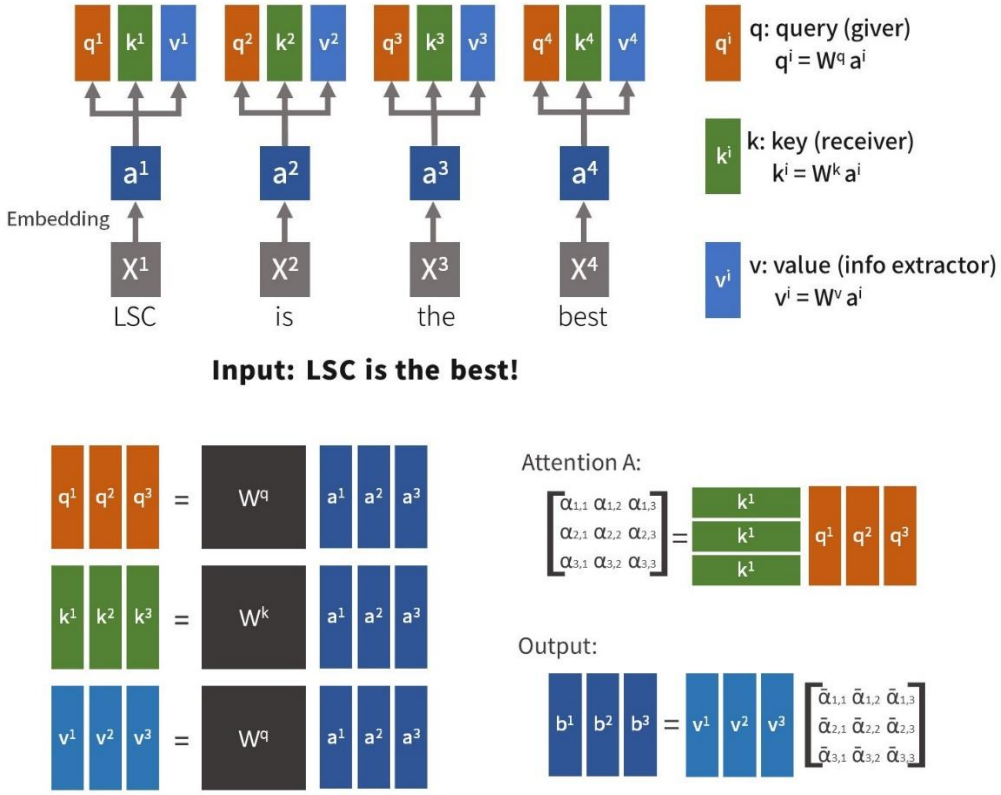

圖 4-5 self-attention 示意圖[8]

2. transformers

在 self-attention 發表的那篇論文裡，同時也提出了新的 seq2seq 架構稱為 transformers。transformers 大量使用了 self-attention layer，在下圖中的 multi-head attention 就是加強版的 self-attention layer。此設計與以前的 seq2seq model 雖然相似，但 self-attention layer 賦予的平行計算能力使得 transformers 能覺得更快，更好。

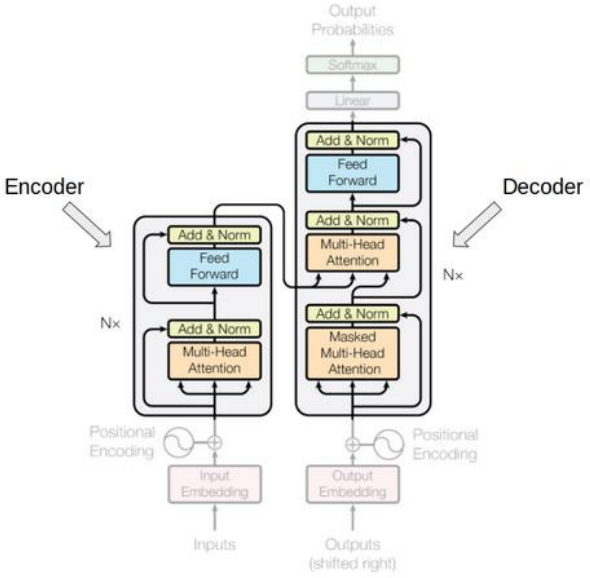

圖 4-6 transformers 示意圖[9]

3. BERT

BERT 擷取 transformers encoder 的部分，並將其以兩種任務訓練，句子關連與猜字。透過多達 24 層的 self-attention layer 與 3.4 億個參數，使他成為近年廣泛被使用的模型。

- 句子關聯性

  將兩個句子輸入 BERT 並將第一個輸出丟進一個小的 feed forward 網路，該神經網路會分辨這兩個句子是否是連貫的。因為 feed forward 網路的能力十分有限，因此 BERT 勢必要給這個網路完整代表這兩句子的向量。將 feed forward 網路移走，改接上更高階的 classifier 即可對句子語意的面相做評估

- Unmask

  另一個用來訓練 BERT 的任務是 unmask。Unmask，任務就是猜出句子挖空處要填什麼字，與克漏字十分相似。將被挖空那一格的輸出接上神經網路，並以 30000 個輸出表示網路猜出來的結果。

  本研究使用此任務模式，用以猜出移除目標單詞後，有什麼適合的選項可以填上去，避免寓意尷尬不順暢。

三、收集訓練模型所需之文本

(一) TOEFL Writing Samples

TOEFL Writing Samples 是托福收集 11 個非英語母語國家人士參加托福考試的寫作，製作成的適合自然語言處理用途的檔案。文本都根據給定的八個題目之一寫作出，文本寫作等級被區分成三級(High, Medium, Low)，總共有接近一萬篇文本。托福的資料庫已將文本 tokenized 並用一個 csv 檔存放各文本的等級，來源題目與作者國籍。

(二) New York Times

我們下載大量的紐約時報文章，並將其整理，只留下文章內容後製作為數個壓縮的文字檔

四、實驗一: 以紐約時報及托福寫作範文分析指定文本的單字出現頻率與作文等級的關聯

(一) TOEFL Writing Samples

我們將這些範文按照等級分別輸入程式，利用 spaCy 將句子拆解為詞彙並轉回原型(如：原文為 started，則只取 start)，並利用 PyEnchant 和 difflib 做拼字修正，統計該等級中所有單字的出現頻率，最後將結果以 json 檔輸出為 toefl_wordcount.json

```
import spacy
import enchant, difflib
from collections import defaultdict
import json

nlp = spacy.load("en_core_web_sm")
enc = enchant.Dict("en_US")
wc={'high':defaultdict(lambda: 0), 'medium':defaultdict(lambda: 0),
    'low':defaultdict(lambda: 0), 'all':defaultdict(lambda: 0)}

for level in ['high', 'medium', 'low']:
    wordcount=0
    print(f'counting level ... {level}')
    with open(f'database/TOEFL_{level}.txt') as text_file:
        for paragraph in text_file:
            wordcount+=1
            doc=nlp(paragraph)
            for token in doc:
                if enc.check(token.lemma_) == 1:
                    wc[level][token.lemma_]+=1
                    wc['all'][token.lemma_]+=1
                else:
                    DICT,mx = {},0
                    a = set(d.suggest(token.lemma_))
                    for b in a:
                        tmp = difflib.SequenceMatcher(None, token.lemma_, b).ratio()
                        DICT[tmp] = b
                        if tmp > max:
                            mx = tmp
                    wc[level][DICT[mx]]+=1
                    wc['all'][DICT[mx]]+=1
    print(f'level {level} completed')
    print(f'total wordcount: {wordcount}')

for level in wc:
    wc[level]=dict(wc[level])
with open('database/toefl_wordcount.json', 'w') as json_file:
    json.dump(wc, json_file)
```

圖 4-7wordcount 程式碼

我們分析 wordcount，定義一個參數$f_{r,l}$為在給定單字r在給定等級l裡的出現頻率，接著定義另一個參數$s_r = \frac{0.6\times f_{r,high}+0.4\times f_{r,medium}}{f_{r,low}}$，表示單字$r$的等級。

考慮到低等級的使用者知道的單字量較少，也只會使用較少不同的單字，我們可以假設較低等級的詞彙會具有較低的s_r。為了驗證較高等級使用者是否使用較多的高等單字，我們作了以下實驗: 我們取總出現次數最多的前二十個單字，並計算出它們的s_r值。為方便觀察，我們將每個單字的平均出現比例設定成 100%，再等比例調整$f_{r,l}$值。

(二) New York Times

我們將範文全部輸入程式，同樣以 spaCy 處理詞彙並計數，結果存為 nyt_wordcount.json，考慮到紐約時報文章來源，其文章等級已足夠高，加上我們只需要計算單字在範例文本(如 TOEFL Writing Samples, New York Times)中的出現頻

率，我們認為沒有必要再將其歸類為任何等級的文本。

五、實驗二: 實作 word2vec 模型以計算詞向量

(一) word2vec 模型

為了找出與要替換的單字的相似程度，我們必須找出所有單字的詞向量(word vector)。因此，本實驗中，我們以紐約時報訓練出了 word2vec 模型並評估了有效數據比例

我們利用 gensim 建模，由於 word2vec 是非監督式學習，我們希望訓練的資料庫越全面越佳，因此我們將紐約時報的文字檔輸入程式，參數設定如下表所示

model name	size	window	min_count	workers	epochs
word2vec_1	150	10	2	10	5

表 4-1 word2vec 超參數表

其中 size 指的是在此訓練中，詞彙會被呈現在多少個維度的向量空間內；在我們使用的 CBOW 下，window 參數決定 word2vec 猜測詞彙的窗口大小；min_count 則是決定詞彙會被用來訓練的最小出現次數；workers 則決定訓練的線程數量；最後的 epochs 決定訓練這個模型的迭代次數，最後模型存為 word2vec_1.model。

```
model = gensim.models.Word2Vec (documents, size=150, window=10, min_count=2, workers=10)
model.train(documents,total_examples=len(documents),epochs=5)

model.save('word2vec_1.model')
```

圖 4-8 word2vec 程式碼

由於出現次數少於 2 次的單字不會被記錄下來，而這種單字大多為拼字錯誤，因此我們要避免錯誤單字出現過多，以增加有效字彙量。實驗一中，我們以 PyEnchant 做拼字修正，為了驗證 PyEnchant 是否有效地完成拼字修正，我們計算紐約時報及修正前後托福範文，在 word2vec 計算時有效的單字比例。此比例越高，代表我們有用的資料量將越大

六、實驗三: 整合 word2vec 及 BERT 開發詞彙推薦模組

(一) BERT 模型的導入

由於我們希望能夠讓使用者在介面上進行詞彙升級，因此我們將會需要挑出可

以使詞彙等級提升的替代選項。我們主要用到兩個模型，分別為 BERT 模型，針對應被更換的詞列出適合在該處使用的詞彙表，以及 word2vec(word2vec_1)模型，從上述詞彙表中尋找意義相近的詞彙。利用此二模型，不僅兼具流暢度，也不更動句子原始的意思，這正是 level up 的目標

具體來說，我們先"遮蔽"(mask)應被替換的詞彙並用預訓練的 BERT 模型進行"填空"(unmask)，將 BERT 填入的選項與該選項的分數存進 options，如圖所示。

```
sentence="The 'beautiful' girl was dancing in the garden."
masked=sentence[:sentence.find('\'')]+'[MASK]'+sentence[sentence.find('\'', sentence.find('\'')+1)+1:]
word=sentence[sentence.find('\'')+1:sentence.find('\'',sentence.find('\'')+1)]
top_k=50

from transformers import pipeline
unmasker = pipeline('fill-mask',model='bert-base-uncased', top_k=top_k)
options_complete = unmasker(masked)
options = []
for doc in options_complete:
    options.append((doc['token_str'], doc['score']))
```

圖 4-9 BERT 程式碼

在將選項存入 options 之後，我們利用 word2vec(word2vec_1)尋找所有詞彙與應被替代的詞彙的相似程度(cosine similarity)，以其相似程度作為加權，乘上 BERT 算出的分數作為該詞彙的總得分，詳細程式碼如圖。也就是說，一個詞彙在替換某個給定句子中的特定單詞時的總得分為將其遮蔽後 BERT 填入該詞彙的分數乘上這個詞彙與原本的特定單詞之相似程度。如此一來，我們能確保推薦的單詞不會突兀的出現在句子中，也不會與原本的句子意思相去甚遠。

```
import gensim
model = gensim.models.Word2Vec.load('word2vec_1.model')
w1 = word
synonyms=model.wv.most_similar (positive=w1,topn=top_k)


synonyms=dict(synonyms)
print(options)
print(synonyms)
for i in range(top_k):
    print(options[i][0])
    try:
        options[i][1]*=synonyms[options[i][0]]
    except:
        options[i]=0
print(options)
```

圖 4-10 寫作升級程式碼

伍、 研究結果

一、實驗一: 以紐約時報及托福寫作範文分析指定文本的單字出現頻率與作文等級的關聯

為了驗證較高等級使用者是否使用較多的高等單字，我們取 TOEFL Writing Samples wordcount 中出現次數最高的前 20 個單字，並計算他們在各等級中出現次數佔該等級的比例與他們在所有單字中出現次數佔的比例。為方便觀察，我們固定任何單字在所有單字中的出現比例為 100%並按照其調整比例等比例調整其在各等級中的出現比例，並求出其s_r值，結果如表。

詞彙r	$f_{r,low}$	$f_{r,medium}$	$f_{r,high}$	$f_{r,all\ levels}$	s_r
people	113%	107%	89%	100%	0.85
car	108%	100%	99%	100%	0.92
life	96%	100%	100%	100%	1.04
time	109%	102%	95%	100%	0.90
thing	145%	116%	71%	100%	0.61
fact	88%	100%	102%	100%	1.15
idea	114%	106%	90%	100%	0.85
subject	114%	105%	92%	100%	0.85
product	115%	100%	97%	100%	0.85
student	110%	103%	95%	100%	0.89
example	108%	102%	95%	100%	0.91
way	104%	102%	96%	100%	0.95
knowledge	91%	97%	105%	100%	1.12
concept	91%	105%	95%	100%	1.09
community	117%	97%	101%	100%	0.85
person	108%	104%	93%	100%	0.90
advertisement	115%	104%	92%	100%	0.84
reason	129%	109%	83%	100%	0.72
lot	112%	116%	77%	100%	0.83
year	97%	98%	103%	100%	1.04

表 5-1 TOEFL writing samples 常見字之相對字頻

二、實驗二: 實作 word2vec 模型以計算詞向量

為了知道有效的資料量，我們以拼字修正前後的 TOEFL Writing Samples 作為訓練文本訓練 word2vec，並將其詞彙有效比例與幾乎無錯字的紐約時報比較，結果如表所示。

訓練文本	參數設定	有效詞彙數 (effective words)	全部詞彙數 (raw words)	詞彙有效率
無拼字修正的 TOEFL Writing Samples	參考表 4-1 模型 word2vec_1 之參數	34572264	50764818	約 68%
拼字修正的 TOEFL Writing Samples	參考表 4-1 模型 word2vec_1 之參數	38282329	50663713	約 76%
New York Times	參考表 4-1 模型 word2vec_1 之參數	201397765	253318565	約 80%

表 5-2 拼字改正之影響

三、實驗三: 整合 word2vec 及 BERT 開發詞彙推薦模組

我們利用 BERT 與 word2vec 模型製作出一個輔助模組，能對一個輸入的句子與句中一個給定單詞回饋一些建議修正的單詞。測試時我們輸入了 The 'beautiful' girl was dancing in the garden. 這樣的句子，其中應被替換的詞為'beautiful'。而模型建議將其更換的選項如表

推薦序位	options	score	推薦序位	options	score
1	'lovely'	0.78	11	'prim'	0.60
2	'gorgeous'	0.72	12	'magical'	0.60
3	'wonderful'	0.68	13	'evocative'	0.60
4	'seductive'	0.66	14	'glorious'	0.60
5	'charming'	0.66	15	'gloriously'	0.59
6	'endearing'	0.64	16	'delightful'	0.59
7	'magnificent'	0.64	17	'haunting'	0.59
8	'serene'	0.64	18	'sumptuous'	0.59
9	'lush'	0.62	19	'ghostly'	0.59
10	'handsome'	0.61	20	'nice'	0.58

表 5-3 寫作升級詞彙表

可以注意到其提供的詞彙等級大多直觀上比原本的 beautiful 高。然而我們也可以注

意到其亦有提供如 gloriously 等副詞型詞彙。

陸、 討論

一、實驗一: 以紐約時報及托福寫作範文分析指定文本的單字出現頻率與作文等級的關聯

觀察頻率最高的前二十個單字進行計算後的結果，可以注意到 thing 或 lot 這類的單字的確具有較低的s_r值，而 knowledge、fact 等直觀上較複雜的詞彙也具有較高的s_r值。根據歐洲共同語言參考標準(CEFR)所訂出的英文單字檔案(English Vocabulary Profile)，而在此系統中，等級由低到高分別被評為 A1, A2, B1, B2, C1, C2。我們發現 thing 與 lot 在其中皆為 A1 等級，而 knowledge 及 fact 則分別為 A2 及 B1。本實驗證實可以藉由計算出單字的在不同等級文本的字頻比例，來評估單字的優劣。

二、實驗二: 實作 word2vec 模型以計算詞向量

我們利用 spaCy、PyEnchant、difflib 對 TOEFL Writing Samples 做拼字修正，並以此為資料訓練 word2vec 模型，由實驗數據可發現，在做拼字修正後，有效的單字比例由 68%顯著的提升至 76%，增加超過 240 萬的單字可以訓練，這對 word2vec 模型的訓練是十分有益的。

三、實驗三: 整合 word2vec 及 BERT 開發詞彙推薦模組

我們利用 BERT 與 word2vec 所建立的模組具有完備的推薦詞彙功能，舉例來說，對於測試 The 'beautiful' girl was dancing in the garden.時，系統即推薦許多明確的較為華美且複雜的單詞。

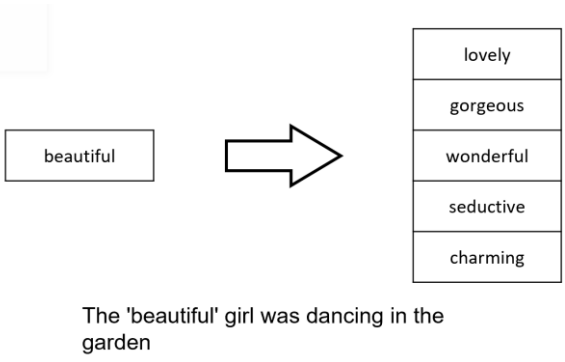

圖 6-1 寫作升級完成圖

柒、 結論

一、托福寫作測驗中較低等級的作文用詞等級也相對較低，因此可作為評估單字等級的方法

二、利用 PyEnchant 實施拼字修正，可提升有效數據的比例

三、基於 BERT 與 word2vec 技術，以紐約時報文本訓練的輔助模組可有效推薦詞彙

捌、 參考文獻

1. Cambridge, English Vocabulary Profile, https://www.englishprofile.org/wordlists/evp
2. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space, https://arxiv.org/abs/1301.3781
3. Leemeng, 淺談神經機器翻譯&用 Transformer 與 TensorFlow 2 英翻中 https://leemeng.tw/neural-machine-translation-with-transformer-and-tensorflow2.html
4. Ashish Vaswani, Attention is All You Need, https://arxiv.org/abs/1706.03762
5. https://corpling.hypotheses.org/495
6. https://tengyuanchang.medium.com/讓電腦聽懂人話-理解-nlp-重要技術-word2vec-的-skip-gram-模型-73d0239ad698
7. https://towardsdatascience.com/word-embeddings-for-nlp-5b72991e01d4
8. https://medium.com/lsc-psd/introduction-of-self-attention-layer-in-transformer-fc7bff63f3bc
9. https://eigenfoo.xyz/transformers-in-nlp/

玖、 未來展望

一、調整 word2vec 模型參數，並比較不同參數之模型建立的模組

二、推薦詞彙功能加強，支持片語的推薦以及針對目標物為片語時的推薦

三、使用不同的訓練文本訓練輔助寫作模組，將模組在地化或能支援不同等級的文章改進

四、推廣 wordcount 結論，利用使用者文章用詞判斷使用者等級並推薦適合該等級使用者的詞彙表

五、建立更好的使用者介面，使使用者能夠被更友善的對待