中華民國第 61 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

佳作

052510

以 Seq2seq 模型實現論文心智圖自動生成工作

學校名稱：高雄市立高雄高級工業職業學校

作者：	指導老師：
職三 林榮顯	鄭家和
職三 陳建宏	康姿瑩
職三 朱峻賢	

關鍵詞：Seq2seq、自然語言處理、心智圖

摘要

做研究時，經常需要閱讀多篇長篇論文，如何快速理解並歸納重點是影響效率收益的關鍵任務。針對此問題本研究運用深度學習配合其他程式算法輔助。將整個問題拆解成三個主要部分：模型處理、結構剖析、心智圖輸出，模型處理的部分我們訓練了三種不同的Seq2seq模型實現文本摘要，分別使用LSTM、BERT、ALBERT、比較並選擇綜合效益最好的模型；而負責拆解論文的結構剖析系統，則利用論文目錄分解整份論文，最後，心智圖輸出系統則整合剖析系統與模型結果，再調用Xmind API去生成架構清晰易讀易理解的論文心智圖。

壹、研究動機

對於一個想研究技術的人來說，在短時間內閱讀長篇文本（論文）並理解其內容顯然是相當困難的，長文本當中或許會有相當多的冗言贅字，影響讀者瞭解該篇文章的重點。於是，本身對於深度學習有研究的我們，在生活當中也同樣遇到文本過於冗長而不易閱讀的困擾。因此，我們想利用深度學習的領域結合課堂上所學的程式撰寫技術來解決此問題。

貳、研究目的

深度學習是一種常見的資料分析方法，讓電腦從大量資料中學習如何將輸入資料映射為一個理想的輸出結果，而將深度學習應用於NLP（自然語言處理）是近幾年常見的研究領域，其中抽象式摘要（Abstraction-based summarization）已經發展相當多年，雖然已有許多關於這個領域的論文及模型，但卻鮮少有專為中文所設計的模型實例，因此，我們想要製作一個處理中文長文本摘要生成的語言模型，並活用我們課堂上所學的C#與Python語言撰寫前端介面及後端文本處理程序，來達到簡化閱讀、梳理重點、統整架構之目的。

參、研究設備及器材

一、 硬體環境：

（一） 電腦一：

1. CPU：Intel(R) CoreTM i7- 9750CPU@2.60 GHz
2. RAM：32.0 GB
3. GPU：NVIDIA GeForce GTX 1660 Ti 6GB
4. 作業系統：Ubuntu 18.04

（二） 電腦二：

1. CPU：Intel(R) CoreTM i7- 6700CPU@3.20 GHz
2. RAM：16.0 GB
3. GPU：NVIDIA GeForce GTX 1050 2GB
4. 作業系統：Win10

二、 軟體環境：Python 3.7、TF 1.4、TF 2.2、Jupyter Notebook

肆、研究過程或方法

一、 研究流程圖（圖1）

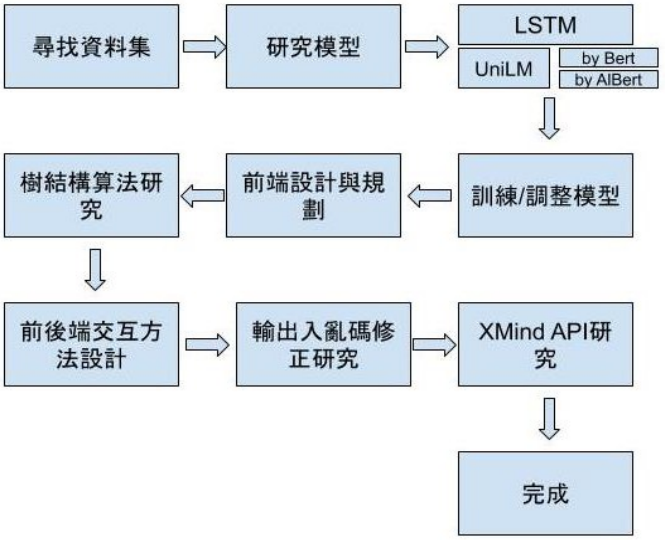


圖 1：研究流程圖

二、 模型製作

本論文主要使用三種演算法實現功能，分別為LSTM、Bert、Albert，其在語言模型中時常被運用到，因此，我們團隊決定從此三種模型去探討並實現文本摘要功能，以下為模型介紹與製作重點整理。

（一） 資料預處理

因模型所讀取的文本為txt檔，若是文章的來源為pdf時，轉成txt恐有一些不正確的符號出現於本文中，此現象會使Seq2seq的注意力機制錯誤的關注到這些雜訊，導致輸出結果產生相當大的誤差。為了能讓神經網路不受雜訊干擾，並且使電腦能理解其資料，以下為實現方法：

1. 將輸入文本的長度統一限制在 256 位字元，長度不足者以 0 值補齊，過長者分批輸入模型。
2. 去除標點符號。
3. 使用字典將輸入資料轉為獨熱編碼（One-Hot encoding）（圖2）。
4. Embedding（Word2Vec）[1]，其用意字詞用數學向量的方式來代表它們的語意。將字詞嵌入到一個語意張量空間，使每個獨熱編碼能映射到一個專有語意向量，並賦予其相對語義關係。

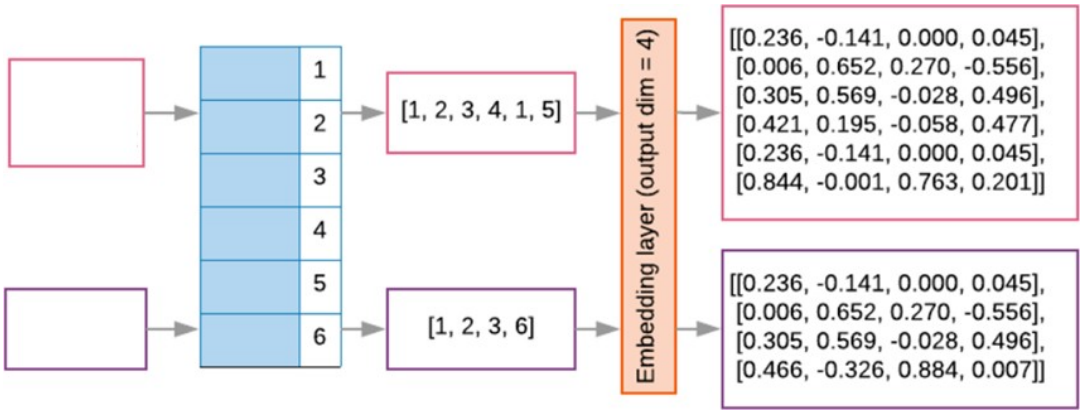
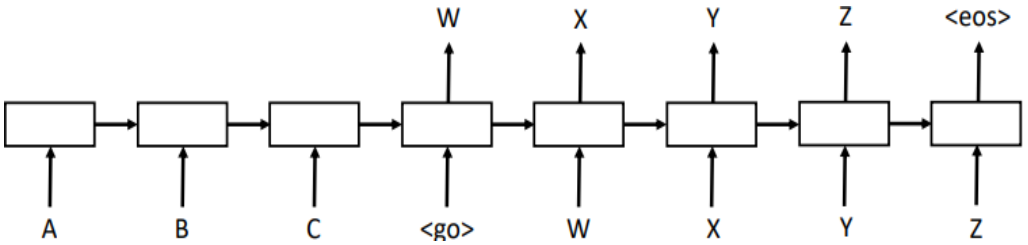


圖 2：One-Hot encoding 與 Embedding(https://developers.google.com/machine-learning/guides/text-classification/step-3)

（二） seq2seq[2]

1. seq2seq 架構介紹（圖3）



https://iter01.com/6587.html)

2. Encoder

其用意是獲取輸入序列的整體訊息並將其處理得到多個張量，本論文統一獲取256個張量。若要配合注意力機制，則要將所有得到的向量進行線性轉換後得到K（Key）與V（Value）。

3. Decoder

其用意為轉化Encoder輸出的張量，並且生成想要的結果，而本論文採用單向生成的方式處理。若要配合注意力機制，則要依據當前的輸入變換得到Q（Query）。

4. Attention機制

其用意為篩選出適合當下時間點的輸入資訊，方法為透過當前時間點的Q 和Kt之間進行點積（Dot-product），經過計算後再透過softmax函數將結果變換成0~1之間的值，然後再將其與Vt進行點乘後即可得到輸入序列的所有重要資訊。

以下為本論文所採用的注意力方程式：

$$softmax\left(\frac{qk^t}{\sqrt{d^k}}\right)v^t \qquad (1)$$

（三） RNN實現Encoder與Decoder[3]

1. RNN介紹

RNN（ Recurrent Neural Network， 遞回神經網路C，圖4 ） 是最先實現seq2seq 架構的神經網路，由FFN（Fully-connected Feed Forward Network，全連接前饋神經網路）演變而來。RNN擁有更強的序列概念，適合利用於處理具有時序特性的資料，如文字、天氣預測等等。而FNN的公式為ax+b=y，RNN的

公式為ax+yt-1+b=yt。

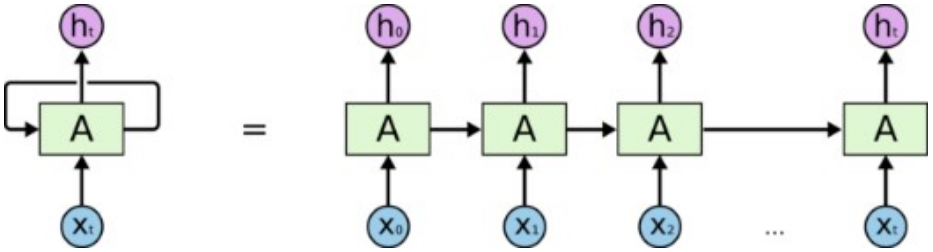


圖 4：RNN示意圖(https://colah.github.io/posts/2015-08-Understanding-LSTMs/)

2. 模型建構

本論文為使用的RNN架構為LSTM（Long short-term memory，長短期記憶）其效果相較於Simple RNN結果更好且不需大幅更改架構， 以下為其Encoder與Decoder重要構成介紹。

（1） Encoder

採用雙層雙向LSTM，雙向的使用能讓電腦所得到的語意更精確，並且在每個LSTM層前將張量進行正則化，以防梯度爆炸或消失。

（2） Decoder

使用雙層單向LSTM，與Encoder一樣都有做正則化，不過Decoder所用的為批次正則化，目的為加速收斂和穩定模型。

（3） Attention 參數取得

Q：從Decoder當前獲得的ht向量進行線性變換所產生的值。

K、V ：從Encoder整個序列所產生的每項ht分別進行線性變換所產生的值。

（四） BERT[4]

1. BERT簡介

BERT是由Google以無監督方法訓練出的LM（語言模型），主要由12層Transformer疊加而成，與論文Attention Is All You Need[4]一起提出，一般使用在 seq2seq的Encoder 層。

2. Transformer簡介（圖5）

Transformer是由論文Attention Is All You Need提出，論文作法為使用多頭自注意力機制來替代seq2seq中LSTM的模型，以解決LSTM難以發揮硬體的平行運算效能與無法建立有效性超長序列關係的兩大缺點， 自注意力機制的架構與數學概念其實和seq2seq的注意力機制相同，不同點為K、Q、V的取得，從Encoder、Decoder變為輸入自身，顧名思義自注意力機制K、Q 、V皆為輸入本身。以下對Transformer所使用的每樣元件做解析：

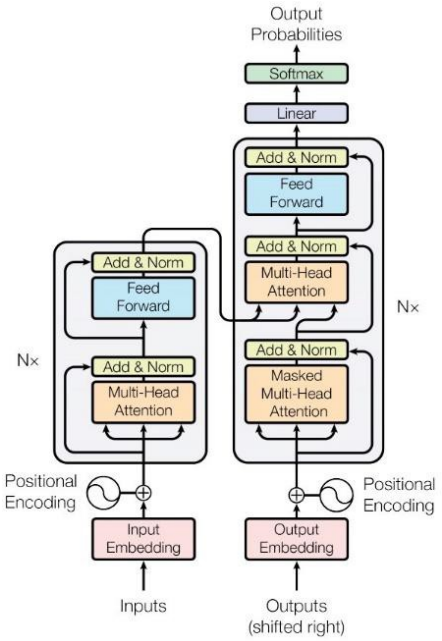


圖 5：Transformer架構圖(Ashish Vaswani， et al. “Attention Is All You Need”， 2017)

（1） Multi-Head Attention（多頭注意力機制）

此為 Transformer 的核心，多頭注意力機制的概念上就是將一組K、Q、V 利用張量拆分法切成多份 K、Q、V。然後再分別進行Attention取得注意力權重，其和Seq2seq中的注意力機制算法相同，最後將其結果與每項Vt做點乘運算，以獲得整個序列的最精練資訊後再將結果送入具有正反向傳播能力的線性系統（FFN）進行學習。

（2） Mask（遮罩）

使用遮罩可使 Attention（注意力機制運作）關注在有意義或者正確的資訊上。

Padding mask：此遮罩負責將序列中用來補齊序列長度的〔PAD〕Token 覆寫為 0，讓 Transformer 不會關注到這些位置。

Look ahead mask：因為序列的生成是有其順序性的，但 Decoder 進行生成時其注意力機制會同時獲取整個輸出序列的資料，為了確保 Decoder 在進行自注意力機制時使其有效範圍限制在當前與過去時序的資訊，不會作用於未來時序的序列區段之空白雜訊，所以使用該遮罩將未來時序的所有資訊覆寫為 0。

（3） Positional Embedding

由於自注意力機制進行時是同時完成整個序列的運算，而非 RNN 的依序進行，但因文字資料擁有時序關係，若依照自注意力機制的運算方法則會使結果產生誤差。所以，為了不影響到結果，我們將輸入張量增加一階存入位置編碼，利用位置邊碼公式讓模型學習到每個Token的位置關係，提供模型字詞的相對位置信息。

以下為位置編碼公式：

$$PE_{\{(pos\ ,\ 2i)\}} = \sin\left(\frac{pos}{10000^{\{\frac{2i}{d_{\{model\}}}\}}}\right) \qquad (2)$$

$$PE_{\{(pos\ ,\ 2i+1)\}} = cos\left(\frac{pos}{10000^{\{\frac{2i}{d_{\{model\}}}\}}}\right) \qquad (3)$$

（Ashish Vaswani， et al. “Attention Is All You Need”， 2017）

（4） Pre-train

Bert會先被訓練為一個對自然語言有一定「理解」的通用模型，之後會再依據需求進行微調以應對更精細的自然語言處理任務，其遇訓練方

法為克漏字填空（Masked Language Model， MLM）與句子相接（NextSentence Prediction， NSP）。

（五） ALBERT[5]

為 BERT 的進化版本，因 BERT 的參數量龐大（110M），運算成本高，ALBERT 改善這些缺點將參數量降到約16M，記憶體用量更小，具體方法如下：

1. Embedding 因式分解

BERT 模型中，V（Vector size）為輸入向量的維度，因為輸出資料的大小要與下次輸入的大小符合，因此 H（Hidden size）等於 E（Embedding size），當 H提升，那麼 Embedding layer 的參數量就會跟著（V * H）提升，但這些參數在訓練後期的更新頻率會大幅降低。 所以，ALBERT 將 V*H 分解成兩個矩陣，由於 H 與 E 在此沒有直接對應，因此打破了 E 等於 H 的限制。原本參數數量為 V*H 將會降低成 V*E + E*H，當 H 的數量很大的時候，這樣的作法能夠大幅降低參數數量。

以下為 BERT 與 ALBERT 參數量計算方程式：BERT：

$$ParameterNumBERT = EV = HV \quad (4)$$

ALBERT：$ParameterNumALBERT = (V+H) \times E$ (5)

2. 參數共享

原本 BERT 內 12 層的 transformer 參數不一致，ALBERT 將參數全部統一，大幅減少參數量，同時，實驗對比每層輸入出相似度，發現 BERT 的結果較為震盪，而 ALBERT 則很平緩，該結果被認為參數共享可以穩定模型。

3. Pre-train 方法改變

BERT 發布一段時間後，有相關研究指出[6]，NSP 訓練效益不高，於是 ALBERT 使用順序預測（sentence order prediction，SOP）取代NSP。

（六） UniLM[7]

UniLM是由微軟以無監督方法訓練出的語言模型，可實現自然語言處理任務的常見架構Seq2Seq和其組件（Encoder、Decoder），其內部主要架構和Google於

2017年提出的Attention Is All You Need論文[2]中的Bert語言模型相似，但其使用特殊的UniLM Mask與Bert-Pre-Train的Mask LM 方法，達成在Bert上實現單向生成，甚至多對多（Seq2seq）的模型架構，亦可用於Albert。。UniLM的架構可透過UniLM mask約束其工作狀態，使其指定為三種模式，無遮罩 （Encoder），全遮罩（Decoder），部分遮罩（Seq2seq），以下為三種模式說明。

1. Encoder（圖6）

與Bert正常情況下輸入方式相同，不使用任何遮罩控制輸入的順序範圍，一次讀取所有輸入序列，進行資料轉換。

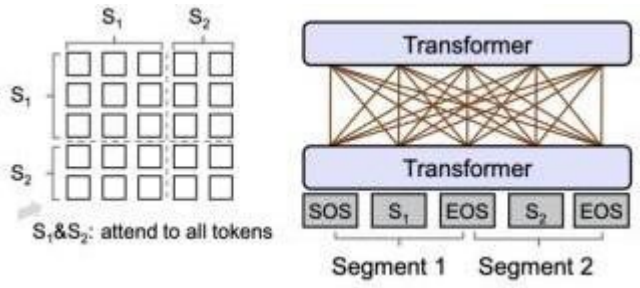


圖 6：UniLM之Encoder示意圖

2. Decoder（圖7）

與GPT系列模型輸出方式相同，使用順序遮罩控制模型的資料輸入流，而模型只可獲取當下與過去時間點所產生及擁有的資料，並使用Bert Pre-train所用到的Mask LM的填空生成方法，使其創造出和LSTM相同的序列讀取或生成功能。

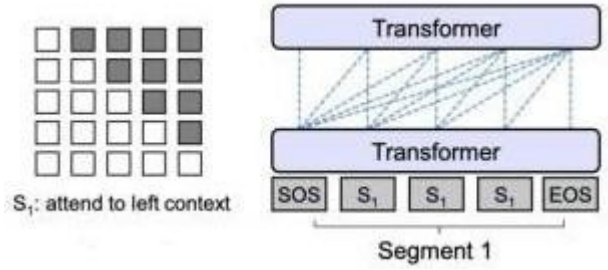


圖 7：UniLM之Decoder示意圖

其方法為限制整體序列的編碼區和解碼區，並在同個序列上規劃出一個沒有遮罩的區域，則該段區域即可作為Seq2seq Encoder使用，而剩餘的區域使用順序遮罩，即可作為Seq2seq Decoder使用，使其在生成時透過自注意力機制獲取編碼區與解碼區內未受順序遮罩影響的資料。

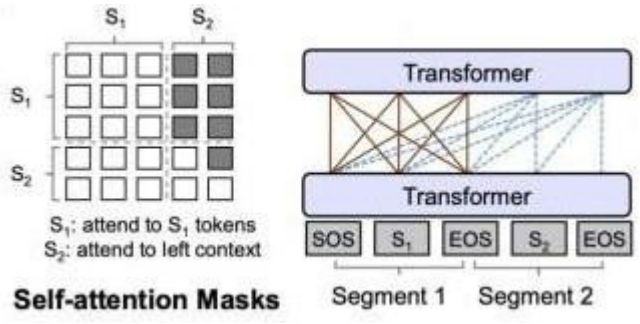


圖 8：Seq2seq組合示意圖

（ 七 ） RealFormer

RealFormer 為 Google 於 2021 年 提 出 的 transformer 優 化 方 案 (https://arxiv.org/abs/2012.11747) ，其概念簡潔，使用殘差連接傳遞每一層Transformer的Attention source，解決了層正規化帶來的梯度消失風險，因此我們決定嘗試加入我們的模型中，以達到更好的理論性能。

（八） 模型訓練

以下附上模型訓練的損失步進圖，如圖9、圖10、圖11，以及rouge&bleu評分步進圖，如圖12、圖13、圖14。

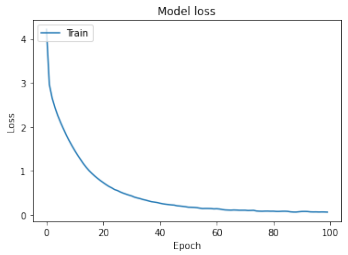


圖9：LSTM

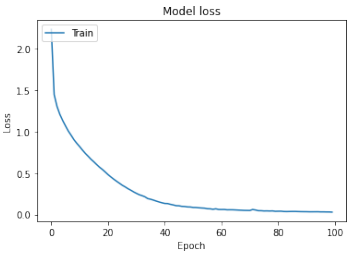


圖10：Bert

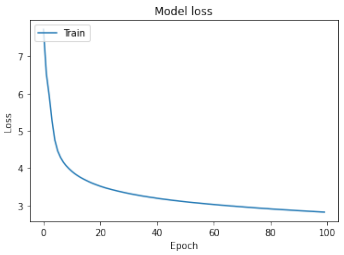


圖11：albert

從loss步進圖中我們可以確認所有的模型在訓練時穩定收斂。

rouge與bleu為自然語言處理任務中常用的兩個評分標準，一個透過召回率另一個則是依據統計學中常見的精確率，從rouge與bleu步進圖中我們可以確認模型效果，其中bert效果最好，albert次之，lstm則較差。

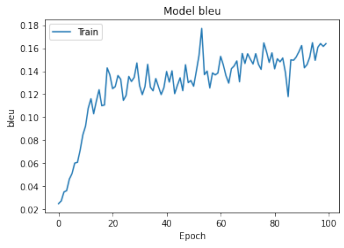


圖12：lstm

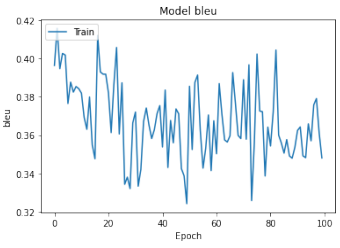


圖13：bert

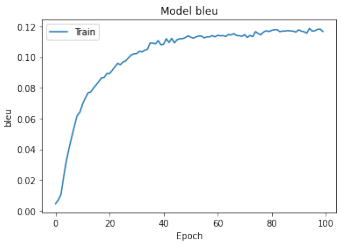


圖14：albert

三、 論文轉心智圖程式

（一） 論文轉樹狀資料結構的算法

為了擷取出論文的內文中每個段落與標題，我們設計出了這套算法來處理輸入資料（即論文），以每個段落的標題作為分界，整理出一個由標題與段落構成的字典（Dictionary）。實際做法如下：

1. 定位到 "目錄" 段落標題
2. 擷取目錄內容作為本文各段落標題的擷取依據
3. 定位到內文的首段上方
4. 利用先前的標題擷取為內文做分段及整理
5. 用段落標題的階層將整理後的資料處理成樹狀的資料形式

（二） C# 與 Python間的交談式處理

因為我們的模型必須在TensorFlow上架構並執行，所以必須使用Python作為後端（back-end）將整理好的樹狀資料依序輸入至模型並產生出結果，但由於C#與Python間並沒有現成有效的循序溝通手段，因此我們編制了指令以便兩者之間的溝通（指令如表一）。

C# → Python 的溝通指令	功能
_force	指示 Python 將記憶區的資料輸入至模型並傳輸生成結果（即強制生成）
_stop	結束與 Python 之間的交談處理
（資料）	將資料傳輸至 Python，

	根據傳輸資料長度及記憶區資料長度決定要暫存於記憶區還是傳輸生成結果
Python → C# 的溝通指令	功能
BeginOutput	提示C# 準備開始傳輸生成結果
EndOutput	提示 C# 生成結果傳輸完畢，準備將下一筆資料傳輸至 Python
Pass	在資料長度不足的情況下，提示 C# 將下一筆資料傳輸至 Python
NeedMore	提示C# 即將傳輸強制生成後的結果
Dropped	提示C# 強制生成後的結果為空
（資料）	將生成的資料傳輸至 C#

表 1：指令表（表一資料來源：研究者自行編製）

（三） XMind 的心智圖檔案內部結構

- 心智圖.xmind （ZIP格式封裝）
- content.xml （心智圖的內容，XML 格式）
- manifest.xml （心智圖檔案的結構，內容固定，XML 格式）
- meta.xml （心智圖檔案的版本，內容固定，XML 格式）

1. content.xml 內部結構

```
<?xml version="1.0"?>
<xmap-content xmlns="urn:xmind:xmap:xmlns:content:2.0" version="2.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xhtml="http://www.w3.org/1999/xhtml"
xmlns:svg="http://www.w3.org/2000/svg" xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <sheet id="心智圖ID">
    <title>心智圖標題</title>
    <topic id="中心點 ID" structure-class="心智圖形式">
```

```
        <title>中心點文字</title>
        <children>
          <topics>
            <topic id="分支點 ID">
              <title>節點的文字</title>
              <children>
                <topics>
                …（子階層的點）
                </topics>
              </children>
            </topic>
            ...（其他分支點）
          </topics>
        </children>
      </topic>
    </sheet>
</xmap-content>
```

2. 節點說明：

<?xml version="1.0"?>：定值，用來標記此文件為XML格式

<xmap-content ...>：定值，用來標記此文件為XMind 心智圖

<sheet id="心智圖ID">：宣告心智圖結構的開始，ID部分必須獨一無二

<title>心智圖標題</title>：心智圖的標題，會顯示在XMind分頁的標題欄上

<title>節點的文字</title>：心智圖節點的文字，會作為節點的標籤顯示在節點上

<topic id="中心點 ID" structure-class="心智圖形式">：心智圖的中心節點，ID部分必須獨一無二，"structure-class" 部分必須是XMind內定義的結構類別名稱

<topic id="分支點 ID">：心智圖的分支節點，ID部分必須獨一無二

<topics type="attached">："topic" 列表，用來標記複數個項目節點

<children>：心智圖節點的子項目列表，在這個項目下的"topic"會顯示在

父節點的子項目

</sheet>：宣告心智圖結構的結束

</xmap-content>：定值

範例：

```
<?xml version="1.0"?>
<xmap-content xmlns="urn:xmind:xmap:xmlns:content:2.0" version="2.0"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xhtml="http://www.w3.org/1999/xhtml"
xmlns:svg="http://www.w3.org/2000/svg" xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <sheet id="0">
    <title>測試心智圖</title>
    <topic id="thisisacenter" structure-class="org.xmind.ui.map">
      <title>這是個中心點</title>
      <children>
        <topics type="attached">
          <topic id="branchhihi">
            <title>這是第一分支</title>
            <children>
              <topics type="attached">
                <topic id="branchinside">
                  <title>這是分支中的分支</title>
                </topic>
              </topics>
            </children>
          </topic>
          <topic id="branchtwo">
            <title>這是第二分支</title>
          </topic>
        </topics>
      </children>
    </topic>
  </sheet>
```

```
</xmap-content>
```

3. 範例成果：（圖9）

圖 15：範例成果

（四） 模型生成結果的過濾

由於Bert系列的模型可能會因為自注意力機制的生成處理方法，輸出Token的機率不定，而導致模型產生震盪使輸出重複字詞以致於可讀性變差。鑑於此緣由，我們開發了一套過濾流程來清除因重複字詞而產生的不易閱讀的問題。過濾流程如下：

1. 從字串第 n 個字元（n 初始為 5）開始，擷取長度為 n 的字串做為比對字串。
2. 將字串中除原比對字串外所有與比對字串相同的字串消除。
3. 重複步驟 1、2 直到比對至字串尾端。
4. 如果 n 大於 1 則將 n 減一後重複步驟 1、2、3。
5. 偵測字尾是否有疊字問題，如果有則消除重複字尾直到無重複為止。
6. 如果字尾為連接字元（"的"、"之"、"及"、"於"、"與"），則消除此字元並重複步驟 5、6。

優點：能有效消除因震盪產生的重複字詞問題。

缺點：可能會意外消除一些重要的字詞，反而使可讀性變差。

（五） 論文心智圖的生成

經過將論文轉化成樹狀資料及將資料投入模型計算的步驟後，為了方便他人

觀看結果，我們需要將樹狀結構轉化成心智圖。轉化成心智圖的流程如下：

1. 擷取樹狀結構的單一節點。
2. 檢查上一個節點與此節點的節點深度如果大於或小於上一個節點，就建立與節點深度同等深度的分支節點，如果等於上一個節點，則建立與前一節點同級的心智圖分支節點。
3. 檢查此節點是否為末端節點如果是，就填入模型生成的段落摘要，如果不是，則填入該段落的標題。
4. 重複步驟1、2、3直到樹狀結構擷取結束。

伍、研究結果

將三種模型的優缺點經過分析與篩選後，我們採用了在效率與精確性都具有優勢的語言模型（Albert）做為心智圖程式的後端，以下為將長文本“偵測與辨識後方來車的嵌入式深度學習系統”丟入程式後所產生的心智圖（圖10），圖11為圖10的部分詳細圖。

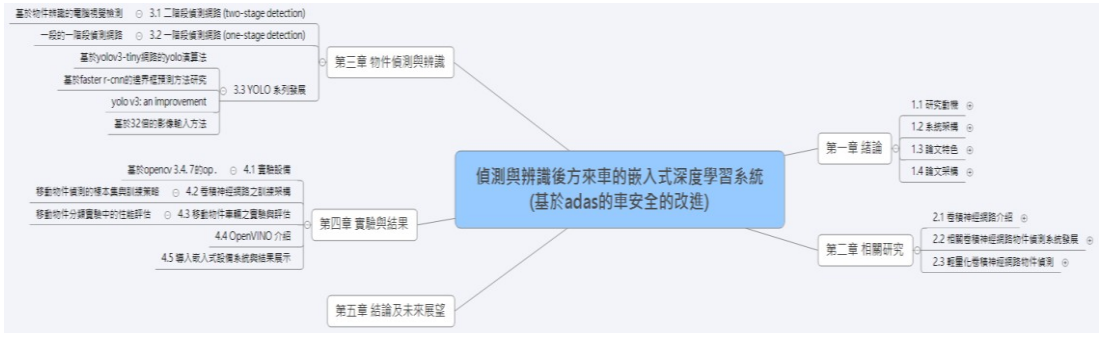


圖 16：範例長文本摘要心智圖

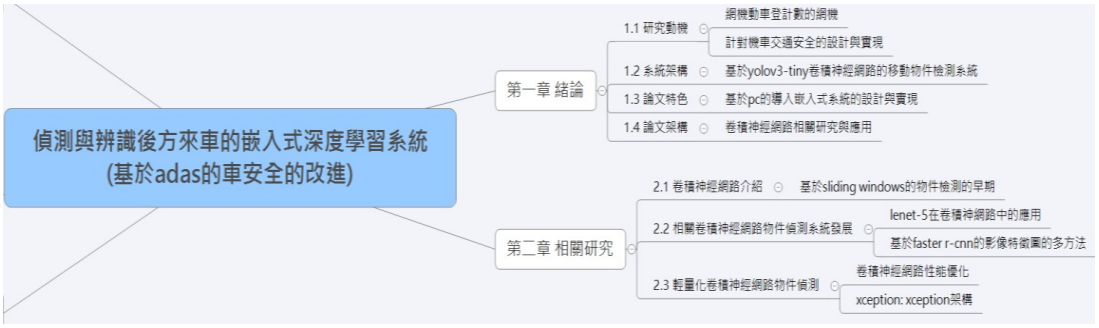


圖 17：圖十的部分區域清晰圖

由圖16與圖17所呈現的畫面來看，我們發現其輸出結果不但有符合文章的摘要

內容，且確實有實現心智圖的樣貌與目的。除此之外，從圖16與圖17的變化來看，即可發現我們的心智圖還能將摘要分支手動整理，讓使用者能夠依照自己的需求使心智圖變成想要的樣貌，達到提高專注力的效果。

陸、討論

一、 三種演算法的比較

經由這次的專題實作，我們參考了許多的相關論文，發現了三種演算法為LSTM、Bert、Albert ，雖然都能做到文本摘要的功能，但以效率與精確度來看卻是Albert的表現最為突出，以下為探討其原因。

（一） LSTM的優缺點

優點：在CPU環境的情況下可以快速地進行運算並得出結果，相較於使用自注意力機制類型的算法，輸出結果不會產生振盪問題。

缺點：因為LSTM容許的輸入文字量不多，若是輸出內容量多的文本將會造成輸出內容變質，使運算時間也會變長，且LSTM為較早期所產生的演算法，與現在的演算法相比一定會有差距。因此，我們沒有採用LSTM達成成果。

（二） Bert的優缺點

優點：Bert利用了Transformer架構去做處理，使輸入文字的上限量比LSTM相對來的高，且其所運用的自注意力機制算法，在具有平行運算能力的硬體上如GPU與TPU，其效率比LSTM好上許多。

缺點：由於Bert是預訓練模型，其參數量過於龐大（110M），雖然運算速度提升，卻對硬體設備需求來說相對提高了，我們現有的硬體設備無法有效的發揮Bert在平行運算上的優勢。其輸出結果會出現無意義的振盪問題，因為我們是由模型自行決定生成截止點，而不是像LSTM有著完全匹配時序資料的結構。

（三） Albert的優缺點

優點：其不但擁有Bert的所有優點，並且改善了Bert參數龐大的問題，使其在硬體上的需求不像Bert要求的那麼嚴格。精確度也不失Bert的水準，因此，我們最後選用了Albert來做我們心智圖程式的後端。

缺點：雖與Bert一樣有著振盪問題，但以結果論來看，其振盪的效應與Bert

相比有減輕的趨勢。

（四） 輸出結果比較

為了證實Albert的輸出結果較為優秀，我們開發了一個算法分析程式，其功能丟入一個長文本並顯示三種演算法的生成的時間與內容，以下為一個範例文本，將其放入程式後所產生的輸出結果為圖18。

範例文本：

對話機器人的智能應答，除可提供快速的客戶服務，亦可以幫助企業節省大量人力，所以提供這樣的服務也代表企業的一種競爭優勢。但其效能調校工作常需耗費時間及人力成本進行維護。我們希望可以提供一種生成式對話機器人，透過深度學習大量資料建立一種自然生成對話的模型。為了提高對話機器人的回覆準確率，我們在機器訓練過程中加入了詞性維度。藉由詞性，讓機器學習了解一個句子的結構及文法，在組成答句時，能夠更貼近人類的語言。 根據研究，生成式對話機器人多為序列對序列的深度學習模型。我們基於門控遞迴單元編碼器與解碼器組成序列對序列框架，再加入詞性，設計出四個新的詞性序列對序列。根據模型訓練後的評估結果，其中三種設計的模型都有高於基準序列對序列框架的效能表現，其中又以雙層詞性序列對序列模型的效能最為優越。 雙層詞性序列對序列的模型，經實驗多重驗證後，應可實作於業界的對話機器人的訓練上。提升的效能，除了可降低維護人力成本外；精準的回覆客戶問題，亦可增加客戶滿意度。

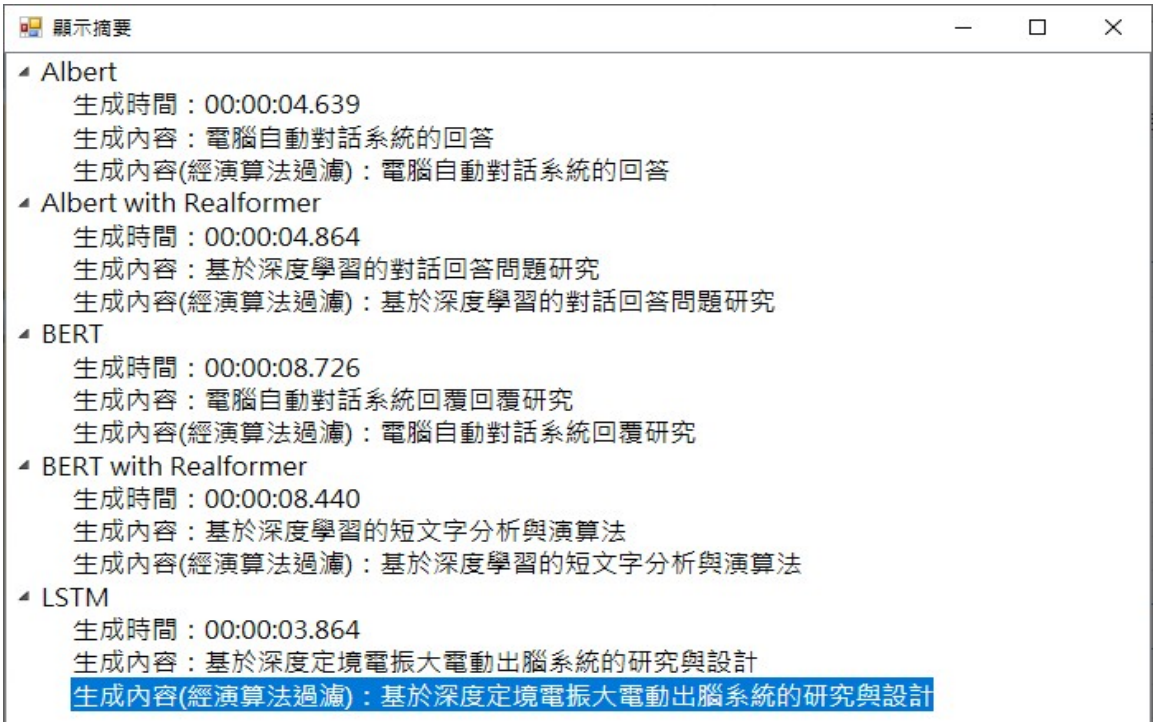


圖18：模型比較圖

從圖18中的成果中可以看到，雖然LSTM的生成時間快速，但其輸出結果資料參考性可說是非常低，而Bert與Albert的生成摘要雖然看起來都較符合文章的敘述內容，但Bert因為參數量的因素使其花了最久的時間。因此，結果最好的明顯為Albert。

二、 Bert運行

前面提到Bert對設備的要求相較於其他兩個語言模型更加嚴格，由於我們的硬體不足與經費層面上的困難，使「摘要生成器」中的Bert模型在訓練時出現了些困難，，導致Bert模型的訓練與調整的進度推進緩慢，但後來我們嘗試了重載訓練方法，雖然小幅犧牲了顯示卡效能，但我們成功將Bert權重完全載入了。

三、 心智圖結構問題

在大量測試時「摘要生成器」的樹狀結構分析器，我們發現因為各長文本的寫法或格式都會有些許的不同，有些會讓篩選節點的正則表示式無法正確作用使程式執行後會出現錯誤，導致生成出的心智圖結構出現問題，若要解決此問題就必須做出能統一文章樣式的輔助程式，希望未來能做出並將此問題解決。

四、 資料集

由於在研究資料及時，中文領域的摘要資料集不多，所以我們使用了CSL資料

集，但由於其蒐集的語料被限制在電腦科學的領域，導致模型的泛化性不足，若是將其他領域的資料丟入程式成效將會降低，希望在未來可以自行蒐集多領域的論文摘要語料。

柒、結論

雖然我們的程式現在分析不屬於電腦科學領域的文本無法達到最好的效果，不過還是能讓讀者多一種參考資料，透過圖表分析來理解文章內容在現代社會中是很有效的瀏覽方法，且若用在電腦科學的領域範圍內，所達到的效果又能更加直接幫助到他人，讓人瀏覽相關長文本時可以精確地找出重點，尤其是常聽學長姐分享他們的大學生活所遇到的問題，其中一項就有參考論文時所遇到的理解困難，這時這個心智圖程式就非常有幫助了。希望未來能夠讓其功能更完善，更有用，幫助到更多人解決閱讀問題。

捌、參考資料及其他

一、 引注論文

[1] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. （2013）. Distributed Representations of Words and Phrases and their Compositionality. ArXiv:1310.4546 [Cs, Stat]. http://arxiv.org/abs/1310.4546

[2] Sutskever, I., Vinyals, O., & Le, Q. V. （2014）. Sequence to Sequence Learning with Neural Networks. ArXiv:1409.3215 [Cs]. http://arxiv.org/abs/1409.3215

[3] Cheng, J., Dong, L., & Lapata, M. （2016）. Long Short-Term Memory-Networks for Machine Reading. ArXiv:1601.06733 [Cs]. http://arxiv.org/abs/1601.06733

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. （2017）. Attention Is All You Need. ArXiv:1706.03762 [Cs].

[5] http://arxiv.org/abs/1706.03762

[6] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. （2020）. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. ArXiv:1909.11942 [Cs].

[7] http://arxiv.org/abs/1909.11942

[8] You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J.,

Keutzer, K., & Hsieh, C.-J. （2020）. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. ArXiv:1904.00962 [Cs, Stat]. http://arxiv.org/abs/1904.00962

[9] Dong, L., Y, Nan., W, Wenhui., W, Furu., L, Xiaodong., W, Yu., G, Jianfeng., Z, Ming., & H, Hsiao-Wuen （2019）. Unified Language Model Pre-training for Natural Language Understanding and Generation ArXiv: 1905.03197 [Cs, CL]. http://arxiv.org/abs/1905.03197

二、 相關參考資料

[1] Hung-yi Lee. （n.d.-a）. Retrieved December 18, 2020, from http://speech.ee.ntu.edu.tw/~tlkagk/courses_DLHLP20.html

[2] Hung-yi Lee. （n.d.-b）. Retrieved December 18, 2020, from http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

[3] LeeMeng—淺談神經機器翻譯 & 用 Transformer 與 TensorFlow 2 英翻中. （n.d.）. Retrieved December 18, 2020, from https://leemeng.tw/neural-machine-translation-with-transformer-and-tensorflow2.html#Transformer%EF%BC%9ASeq2Seq-%E6%A8%A1%E5%9E%8B-+-%E8%87%AA%E6%B3%A8%E6%84%8F%E5%8A%9B%E6%A9%9F%E5%88%B6

[4] Lin, C.-Y. （2004）. ROUGE: A Package for Automatic Evaluation of Summaries. Text Summarization Branches Out, 74 – 81. https://www.aclweb.org/anthology/W04-1013

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. （2013）. Distributed Representations of Words and Phrases and their Compositionality. ArXiv:1310.4546 [Cs, Stat]. http://arxiv.org/abs/1310.4546

[6] Neural machine translation with attention | TensorFlow Core. （n.d.）. Retrieved December 18, 2020, from https://www.tensorflow.org/tutorials/text/nmt_with_attention

[7] TensorFlow API Versions | TensorFlow Core v2.4.0. （n.d.）. Retrieved December 18, 2020, from https://www.tensorflow.org/versions?hl=zh-tw

[8] Transformer model for language understanding | TensorFlow Core. （n.d.）. Retrieved December 18, 2020, from

https://www.tensorflow.org/tutorials/text/transformer

[9] Fine-tuning a BERT model | TensorFlow Core. （n.d.）. Retrieved December 18, 2020, from https://www.tensorflow.org/official_models/fine_tuning_bert?hl=zh-tw

[10] You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., & Hsieh, C.-J. （2020）. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. ArXiv:1904.00962 [Cs, Stat]. http://arxiv.org/abs/1904.00962

[11] 課程介紹—[自學課程]成為 Python AI 深度學習達人的第一堂課 | 政治大學磨課 師 課 程 .（n.d.）. Retrieved December 18, 2020, from http://moocs.nccu.edu.tw/course/172/intro

[12] [論文整理] ALBERT. 更輕型的BERT | by CHEN TSU PEI | NLP-trend-and-review | Medium. （n.d.）. Retrieved December 18, 2020, from https://medium.com/nlp-tsupei/%E8%AB%96%E6%96%87%E6%95%B4%E7%90%86-albert-

【評語】052510

本研究運用深度學習配合其他程式算法輔助，將整個問題拆解成三個主要部分： 模型處理、結構剖析、心智圖輸出。這是個有趣的應用且也具有一定的價值。建議思考若一篇論文沒有目錄時，又該如何區分出章節來進行個別進行摘要。另外，針對作品所提出方法所輸出心智圖的品質，由於評估相當主觀且不容易進行客觀評分，在未來可以多探討如何進行更為客觀的評分。

作品簡報

以Seq2seq模型實現論文心智圖自動生成工作

高級中等學校組
電腦與資訊學科

研究動機與目的

- NLP（自然語言處理）在近年來是相當熱門的一個領域，許多事情透過該領域使人類的科技越來越發達，帶給社會大眾更多的便利性。而我們也對這方面有著相當大的興趣，因此想結合生活中所遇到的問題去發展並解決問題。
- 本研究期望能將長篇論文以一個簡潔、有邏輯且不失精準的樣態呈現出來，達到簡化閱讀、梳理重點、統整架構之目的。
- 將長篇論文分段丟入語言模型輸出成多段文字摘要，並透過自製演算法將摘要整理，最後透過心智圖呈現，有如樹根深入每個段落，更細緻的表達論文內容。
- 現今常見方法為透過一系統將文本轉換為一段摘要，而這方法明顯去除了許多細節，若是使用我們的方法，則可以在保有細節的情況下達成快速分析的目的。

論文摘要資料集

- 使用CSL中文科學文獻數據集。
- 切分為訓練集、驗證集、測試集，分別有10^4、400、400組資料。

表1：資料集內容

一段論文部份短文	與之對應的人工摘要
針對車聯網(IoV)環境下訊息傳輸效率低下、網路資源開銷較大等諸多問題,提出一種適用於城市交通場景下基於車輛節點認…… 平均字數：236.74	車聯網環境下基於節點認知互動的路由演算法 平均字數：19.03

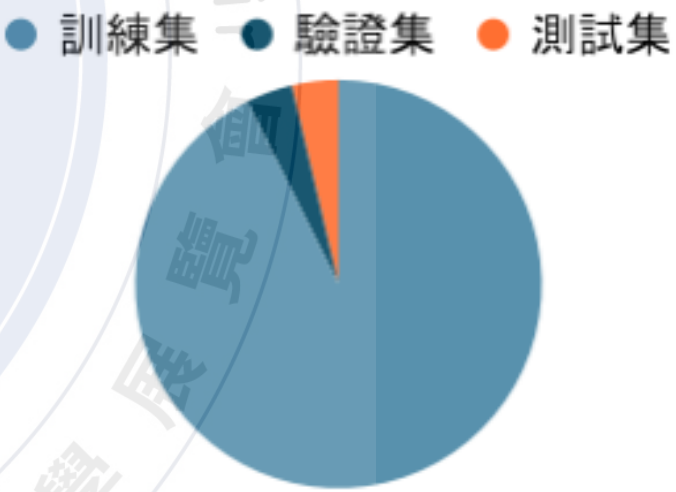


圖1：資料集使用分佈

模型使用

使用Google ALBERT語言模型為基礎並使用微軟UniLM微調使其具備NLG能力。

以下為模型簡易架構圖：

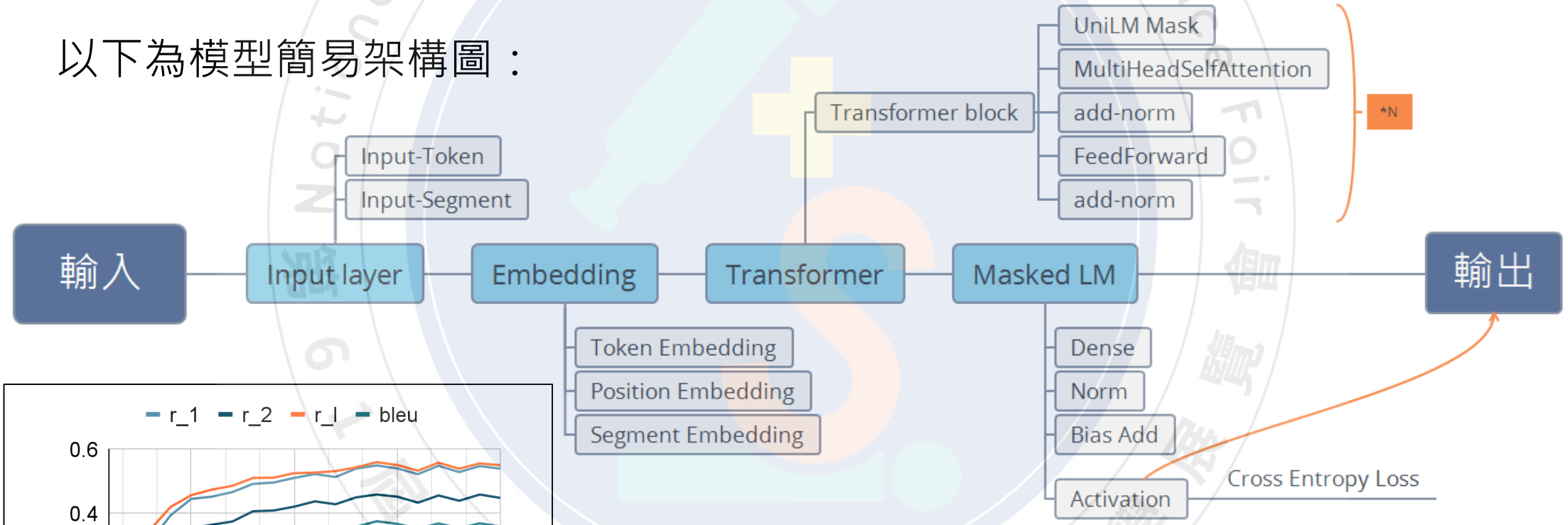


圖2：模型架構簡圖

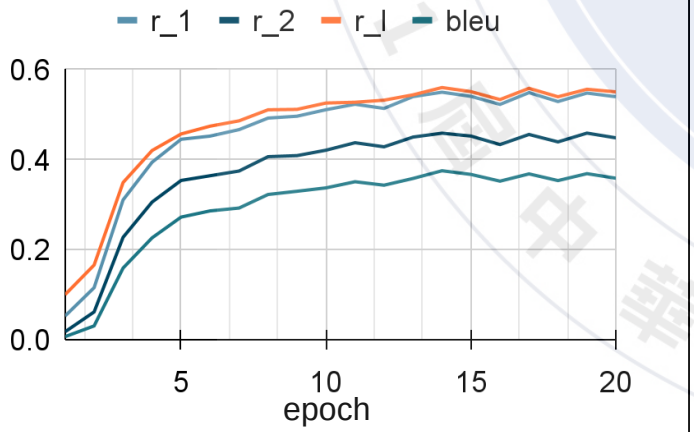


圖3：由此評分步進圖可知訓練有效

論文結構剖析方法

使用每篇論文皆有的目錄，為整篇論文產生樹狀架構以保證心智圖邏輯性。

方法如下：

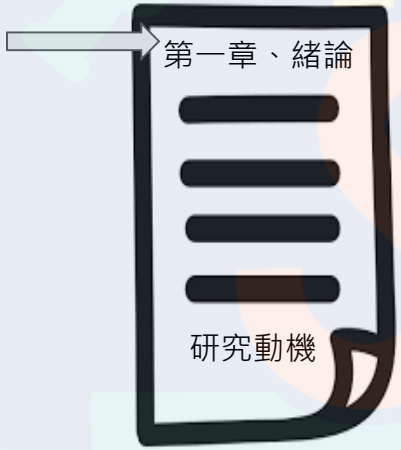
step.1

定位到 "目錄" 段落標題

step.2

擷取目錄內容作為本文各段落標題的擷取依據

step.3

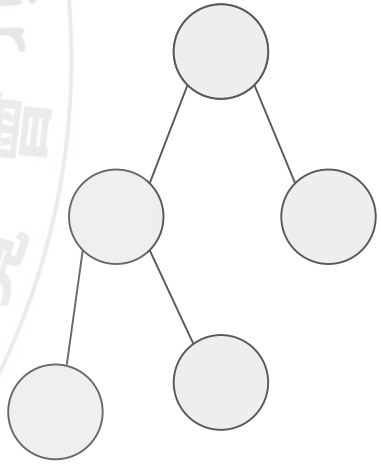


定位到內文的首段上方

step.4

利用先前的標題擷取為內文做分段及整理

step.5



用段落標題的階層將整理後的資料處理成樹狀的資料形式

圖4：結構剖析方法概念圖

C#/Python 交互指令

為本團隊自定義之前後端溝通系統，用於使結構生成系統(前端)與摘要生成系統(後端)能互相配合，溝通指令如下表：

表2：C# to Python之指令

_force	指示 Python 將記憶區的資料輸入至模型並傳輸生成結果(即強制生成)
_stop	結束與 Python 之間的交談處理

表3：Python to C#之指令

BeginOutput	提示 C# 準備開始傳輸生成結果
EndOutput	提示 C# 生成結果傳輸完畢,準備將下一筆資料傳輸至 Python
Pass	在資料長度不足的情況下,提示 C# 將下一筆資料傳輸至 Python
NeedMore	提示 C# 即將傳輸強制生成後的結果
Dropped	提示 C# 強制生成後的結果為空

震盪去除算法

由於Albert的機制缺陷，生成出的摘要可能出類似人類結巴的錯誤我們稱之為震盪，為解決此問題，我們提出了此算法。經由我們多次測試，n初始值取5會達到最好效果。

表4：去除前後對照表

去除前	去除後
電腦自動對話系統回覆回覆研究	電腦自動對話系統回覆研究
基於ai的繁體手寫文字集的手寫文字集	基於ai的繁體手寫文字集
基於社會網路的基於社會網路的分析與研究分析與研究	基於社會網路的分析與研究

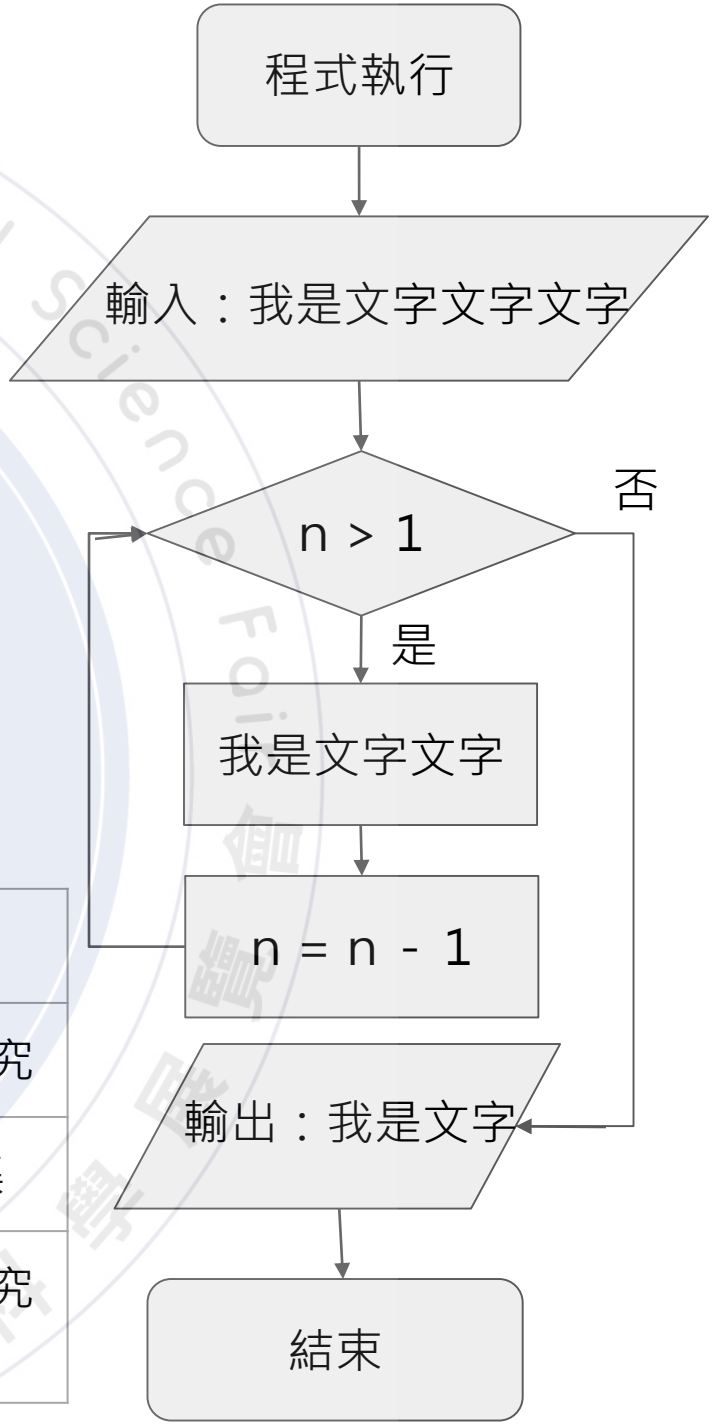
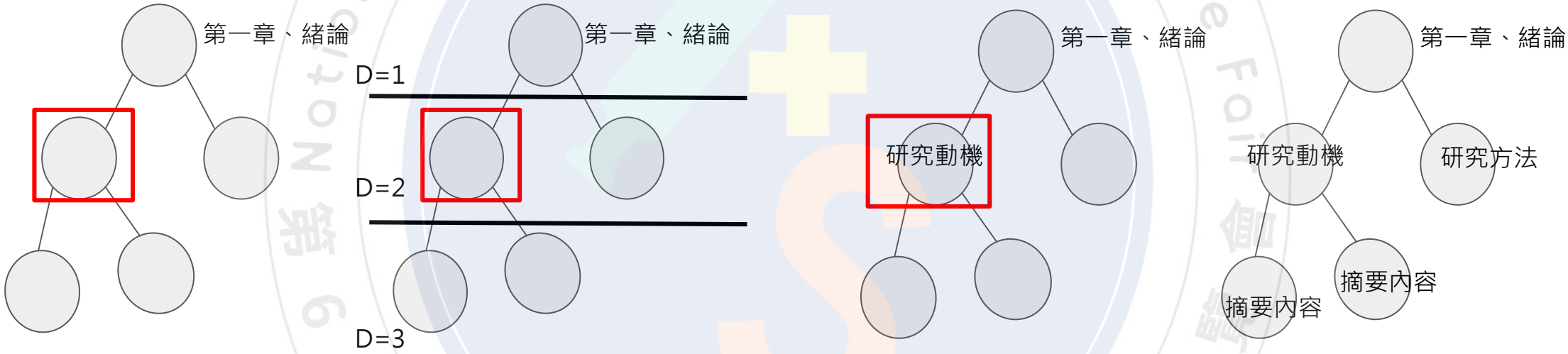


圖5：震盪去除算法流程圖

Xmind心智圖生成

獲取樹狀結構與段落摘要後，還須將其組織，以下是我們提出的組合流程：



擷取樹狀結構的單一節點

檢查上一個節點與此節點的節點深度如果大於或小於上一個節點，就建立與節點深度同等的分支節點，如果等於上一個節點，則建立與前一節點同級的心智圖分支節點

檢查此節點是否為末端節點如果是，就填入模型生成的段落摘要，如果不是，則填入該段落的標題

重複步驟1、2、3直到樹狀結構擷取結束

圖6：心智圖生成方法概念圖

系統操作展示

圖7：本研究之心智圖系統操作展示

研究結果

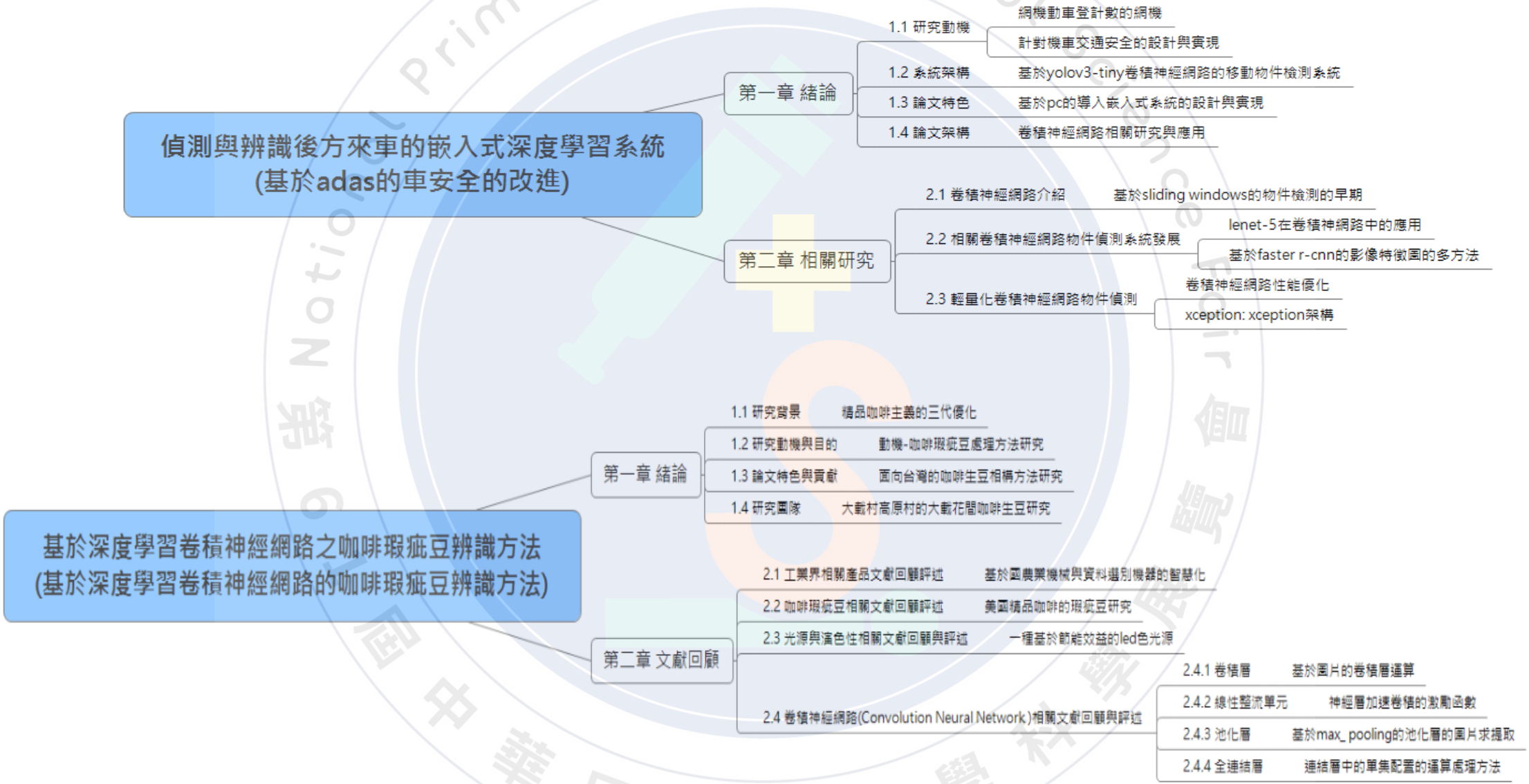


圖8：本研究心智圖自動生成結果二則範例

結論

在研究的過程中，我們訓練多種模型，包括 LSTM、Bert、Albert ，成效為Albert的成績最好，且為了構成心智圖，我們建構並實現分段流程與震盪去除流程，最終實現以下成果：

- 本研究提出一系統以解決將論文轉為心智圖之任務
- 使用ALBERT with UniLM比一般LSTM Seq2Seq效果更好。
- 提出了分段流程與震盪去除流程，以建構心智圖節點與修正內容。
- 若有更多領域之論文的資料集，模型會達到更好的效果。

參考文獻

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

[2] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

[3] Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., ... & Hon, H. W. (2019). Unified language model pre-training for natural language understanding and generation. *arXiv preprint arXiv:1905.03197*.

[4] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[5] Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

[6] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104-3112).