國立政治大學資訊科學系

碩士學位論文

高壓縮比超長文本抽象式摘要生成

High Density Abstraction for Very Long Text Summarization

指導教授：黃瀚萱　博士

研究生：蕭郁君　撰

中 華 民 國 111 年 1 月

DOI:10.6814/NCCU202200337

中文摘要

本研究探討的是文本摘要的新任務：生成具有高壓縮比的長文本抽象化摘要。自動摘要是自然語言處理領域中廣泛研究的主題，目前以 Transformer 架構爲基礎的神經網路模型在新聞的抽象式摘要上，展現了一定的成效。本研究則針對更具挑戰性的輸入類型，書籍，作爲摘要的對象進行探討。與長度僅數百字的新聞相比，書籍長達上萬字或更多，而對超長輸出入進行建模，是當前神經網路模型的一大挑戰。除此之外，書籍摘要需要將大量文字，改用少許概括性的文字重新表述。然而目前不論萃取式或抽象式摘要，主要的原理均是對輸入中的文句進行選擇與排序，故不易將大量詳細瑣碎的文句濃縮成概括性的宏觀概念。因此，書籍摘要的高壓縮率，構成現有摘要生成技術的另一挑戰。

爲解決上述的兩個挑戰，我們提出了一個基於 Transformer 神經網路的多層處理架構，適用於非常長的文本摘要，而且可以在監督式與非監督式兩種模式下運作。爲了訓練我們的模型，我們提出了僞標記的策略，在不需額外人工標記的情況下訓練生成模型，進一步提出了一種自監督學習任務，利用多任務學習的方式，促進抽象摘要模型選用廣泛的宏觀表達方式，將具體詳細的措辭重新表述。實驗結果顯示，與現有方法相比，本篇論文提出的方法可以生成更好的摘要。

關鍵字: 自然語言生成、抽象式摘要、長文本摘要

DOI:10.6814/NCCU202200337

Abstract

Text summarization is a topic widely-studied in the area of natural language processing. Most works of summarization focus on news or document summarization, where the length of input text is usually limited to hundreds of words. This work shows an attempt to deal with a much more challenging case, book summarization. Compared with news article, the length of a book usually exceeds ten thousands or even more, making a barrier to current neural network models, which have a shorter input limitation. The high compression ratio of book summarization forms another challenge for most current extractive and abstractive summarization models, which generate the summary by selecting and reordering sentences or words in the input, failing to condense details into broad, macro concepts. To address these two issues, we present a novel hierarchical model for very long text summarization in two ways, unsupervised and supervised. We train the Transformer-based generation model with pseudo-labeled data in the hierarchical manner for handling very long input text. A self-supervised learning task is further proposed for improving the ability of the abstractive summarization model for rephrasing specific, detailed wording with broad, macro expressions. Experimental results show our approach can generate better summaries compared with existing methods.

Keywords: Natural language processing, Abstractive summarization, Long text summarization

DOI:10.6814/NCCU202200337

目錄

DOI:10.6814/NCCU202200337

DOI:10.6814/NCCU202200337

表目錄

DOI:10.6814/NCCU202200337

圖目錄

DOI:10.6814/NCCU202200337

第一章　緒論

第一節　研究背景

文本摘要任務 (Text Summarization) [17] 是指給予一段文字或一篇文件，利用文字生成的方式，將文章中的要點濃縮成一串簡潔扼要的文字，在自然語言處理領域中經典的課題之一。其中，經由模型/演算法自動產生的摘要稱之爲候選摘要 (candidate summary)，而由人工手寫出的摘要稱爲參考摘要 (reference summary)。

文件摘要主要分爲兩個方法: 節錄式摘要 (Extractive Summarization) [16] 和抽象式摘要 (Abstractive Summarization) [1]，前者是在原始文件內容中，選取一部份重要的詞語、句子或是段落作爲該文件之摘要，而後者則是利用語意理解技術，獲取原始文件的意圖後，根據理解內容及特徵，生成能闡釋整篇文檔重點的一段文字。以文件數量區分的話，也有單文件摘要和多文件摘要的區別。多文件摘要的主軸比起單文件摘要，還多出了以下幾點：構建出具有最大覆蓋率，能覆蓋所有文件的文意、減少因爲字數增加而形成的冗餘詞彙和增強句子之間最大內聚性。本篇的性質實爲單文件摘要，雖然長文件其實可以視爲多個文件的組成，然而與多文件摘要不同的是-我們需要考量到文件的組成順序以及如何在長文中過濾資訊，只挑選出最重要的段落。

現實生活中，人們對文長較長的文件所需的閱讀時數較高，也會相對需要文件摘要輔助，來達到快速掌握長文件或多個文件的主題及內容主軸。因此對於摘要任務而言，受限於模型的輸入長度，如何處理超長文檔，是近年來許多人研究的議題。

由於節錄式摘要所得之摘要會是文檔本身的部分文字，因此若希望能在簡短的文字中表達極其長的文件，會取決於文件本身的文句。首先，文件中的句子是否長度不超過門檻且內容具備重點的句子。再者，倘若選取多個句子作爲摘要，選取到的句子與句子間可能也會有語意不連貫的問題，而導致摘要品質低落。基於前文所提及節錄式摘要的限制，本篇採用了抽象式摘要作爲摘要模型的基礎，並且輔以辭彙改寫，概括文件的重要意涵，生成抽象化的摘要語句。

DOI:10.6814/NCCU202200337

以監督式學習 (supervised learning) 爲例，訓練資料除了需要原始文件本身，還需要人工撰寫的摘要來當作訓練和自動評估摘要品質參考的對象，屆時會需要大量時間及人力投入人工摘要及摘要品質評分才能使模型獲得更好的效能，因此本篇一開始採用自監督學習 (Self Supervised Learning) 的方式，不考慮使用眞值標籤 (ground truth) 的大規模訓練資料集，而是自動化的假生成文件和假摘要來作爲我們訓練摘要任務時的資料。而後再針對非監督和監督式訓練設計不同設置來處理長文本摘要任務。

第二節　研究動機

過往在文本摘要方面的工作主要集中在新聞摘要、社群媒體貼文等文本上，近年來則拓展至對話和其他類型的文本。有別於以往的研究，本研究希望處理的是長文檔的摘要。長文檔中，最普遍且具有結構化形式的就是書籍。書籍中也有分很多種類，撇除一些特殊主題及書畫類書籍，從寫作模式上，大致可以分爲小說類與非小說類兩種。在本研究中，我們將重點置於「書籍」，並且以小說這類的書籍作爲實驗素材。

不同於之前研究的文本類型，一本書的長度多數超過一萬字，超過十萬字的書更不罕見。因此在處理書籍—此類長文本時將會有以下幾個問題：

1. **非常長的輸入導致當前神經網絡模型的技術問題**

   最近的技術模型，大都基於 Transformer 的模型在新聞摘要和其他生成任務中取得了良好的性能，但在合理的計算成本下，最大輸入長度被限制在 512 到 4,096 個詞令 (token) 的範圍內。支持更長的輸入意味者模型參數量大幅增長，不僅需要更多的計算成本，而且提高訓練難度，不易獲致理想的效能，進而影響文本摘要的品質 [2]

2. **高壓縮率**

   如何在有限且短少的數字內中精確的描述整本書的重點，需要的是極高的壓縮率。根據 CNN/Daily Mail 數據集，新聞摘要的壓縮率約爲 14.89%，意即輸入新聞長度爲輸出摘要的長度平均的 14.89%。相比之下，書籍摘要的壓縮率要高得多，以章節資料集 NoverlChapters 爲例，壓縮率有 18.31%。意即我們需要仰賴更高密度的抽象化能力，即摘要模型將幾個長而詳細的句子壓縮成更短、更簡潔的表達的能力。然而現有之摘要模型，無論是抽取式還是抽象式，都是從輸入中抽取重要的句子（抽取式）或單詞（抽象式）並重新排序或局部代換，缺乏運用巨觀的概念去概括細瑣文句的能力。

DOI:10.6814/NCCU202200337

3. **缺乏訓練數據**

大部分的抽象式摘要模型都採用監督式學習，也就是在現有人工撰寫摘要後去訓練模型，但對於書籍摘要，收集帶有摘要註釋的書籍文本實是相對困難。以現有之書籍摘要資料集 NovelChapters 爲例，僅有 90 本書。而新聞摘要的資料集動輒數十萬筆，相形之下量級有極大的差異。再者，書本皆具有版權問題，若建立共享基準數據集供監督式學習使用且予以公開，未先整合各方書籍版權，恐將面臨相關法律問題，單就資料集的建置成本將不切實際。

本研究希望能利用非監督或者監督式的模型訓練方式，彌補訓練資料的不足，並且能從長文本裡眾多的文字中，擷取不同段落中各自的意涵，生成出通順且連貫，能包含上下文文意的摘要。

第三節　研究目的

本研究提出了一種書籍摘要的替代方法。考慮到高密度抽象，我們提出了一個用於多級框架的摘要模型，採用了非監督及監督的訓練方式，介紹了一種構建高抽象訓練實例的新方法。本研究的貢獻有以下兩個方面：

- 我們提出了一個新穎的框架，分別支援無監督與監督的方式構建用於非常長的文本摘要的分層模型。
- 我們提出了一項自我監督的任務，以提高模型將特定、詳細的措辭壓縮爲廣泛的宏觀表達的能力。

DOI:10.6814/NCCU202200337

第二章　文獻探討

第一節　萃取式摘要

萃取式摘要即從原文中選取一部份重要的詞彙片段或者是句子甚至段落作爲該文件之摘要。通常監督式訓練採用端到端的訓練方式，將摘要任務視爲分類問題，利用大量的標記資料，來訓練句子表示; 非監督式訓練則傾向利用圖來架構句子間的關係。

[14] 分析了文件嵌入 (sentence embedding) 對於摘要任務的重要性，利用 BERT 模型對於輸入中不同的句子依據奇偶數做了不同的分段嵌入 (segment embedding)，以及每個句子的開頭都使用 [CLS]，讓 BERT 不僅僅是關注在詞嵌入上，能藉由上面這些修改，更關注在文件嵌入上。實驗也證明該方法在新聞類資料集上有不錯的效能。

在摘要任務裡，撰寫者會將文章轉譯後寫成摘要，一個摘要有可能是多個句子的總結，因此我們無法很直觀的知道摘要和原文是如何對應的。因此也可以利用配對方法，讓輸入和摘要對齊配對，進而找到需要被抽取的句子。例如 [30]，將摘要任務視爲語義文本匹配問題，考慮句子之間的關係。先以句子級的匹配程度挑選摘要，將該摘要組合後作爲候選摘要，之後在藉由與原文檔在語義上的匹配度的考量和候選摘要之間的差異性，挑選出最後的摘要。

非監督萃取式摘要任務如 [4] 將簡單圖 (simple graph) 的排名算法與兩級層次模型結合，通過構建用於文件嵌入的有向層次圖以及不對稱邊權重，將句子和分段的重要性納入圖中。最後直接提取具有前 k 高重要性分數的句子來作爲摘要。

然而多數的萃取式方法也依舊仰賴大量的標記資料來訓練，而萃取式摘要雖說具有一定的摘要品質，但仍然缺乏精煉摘要的能力。

第二節　抽象式摘要

抽象式摘要指的是在了解文意後，通常會在一定長度的限制下，產生一段能涵蓋原文重點的文字。摘要裡的文字可以不在原文中出現。

DOI:10.6814/NCCU202200337

早期抽象式摘要模型多爲 RNN [23] 系列的模型架構，例如 Pointer Generator Network [24]，之後則以 Transformer 系列的模型爲主。

Transformers 模型 [25] 是編碼器 (encoder) 和解碼器 (decoder) 相連组成，而編碼器由 6 个相同的層 (layer) 组成，每個層裡面都設計有多頭自注意力機制，幫助模型學習輸入序列中上下文的特徵，之後會再過個全連階層，其中上面兩個後面都加了殘差連接和正規化機制。解碼器和編碼器相同，一樣使用了 6 个相同的層 (layer) 组成，唯一不同的地方在於它多引進了編碼器解碼器的注意力機制 (cross-attention)，輸入爲編碼器的輸出以及第 i-1 個辭令對應的解碼器的輸出，輸出爲第 i 個辭令對應到的的詞令概率分布。在訓練時，會利用 teacher forcing 的機制來做訓練，也就是說解碼時利用上一步的參考摘要的詞令來預測，藉由遮罩的方式讓模型看不到未來的詞令，然後一次全部解碼；而預測時，因爲沒有參考摘要，所以需要一個個步驟生成每個詞令，產生最後的摘要。

此類模型對輸入中的詞令進行抽取、替換和重新排序，生成總結摘要，如: T5 [18], BART [11], PEGASUS [29] 等等，其中，T5 模型基本上的架構如同上述所提，唯一和一般的 Transformers 不同的點是在位置編碼時，Transformers 使用的是絕對位置，而 T5 則用的是相對位置，在計算注意力權重的時候加入位置資訊，而且每一層都加一次，讓模型對位置更加敏感。作法如下：即每個位置對應一個數值而不是向量，將在自注意力機制裡的 key 和 query 相對位置的數值加在歸一化前，每個 head 都有自己的位置編碼，而所有的層共享一組位置。T5 同時利用了大量的預訓練數據做以下 2 種訓練: 監督式訓練和自監督訓練，利用區間 (span) 替換成特殊辭令 ([MASK]) 去進行自監督訓練; BART 模型則改成雙向的編碼器加上自回歸解碼器架構，比起 T5 則引入了更多去除雜訊的方法幫助訓練，例如將句子隨機排列，文件轉換（隨機挑選詞令，轉換輸入文件，以該詞令作爲該輸入文件的開頭)。上述這些模型在多數的摘要任務上都得到了很好的表現。

T5 模型提供統一的框架用於大部分的自然語言處理任務上，例如：GLUE(General Language Understanding Evaluation)(文本分類任務常用的 9 個數據集)、摘要生成、翻譯任務等等，將這些任務都套用到輸入和輸出皆爲文本的格式，不同的任務在輸入端會有不同的字首 (prefix)，讓模型辨識它現在處理的是哪種任務。由於 T5 爲預訓練模型，因此可以在後續任務中在目標任務/領域上進行微調，學習新的任務或資料。然而雖然 T5 的不限制輸入長度，但仍然會受限於記憶體的大小，因而有了長文本摘要這個主題。

DOI:10.6814/NCCU202200337

第三節　長文本摘要

由於轉換器 (Transformer) 系列的模型無法直接處理超長文本，因此長文本摘要 (long input summarization) 任務大致會用以下 2 種做法來間接達成：第一種方法是簡化注意力機制 (attention) 的參數量，例如 [2]、[28] 和 [8]，或者在模型中增加一些設計，使得原本對於模型來說過長的輸入能被關注到，如 [22]，讓原本的模型可以能夠容納更長的輸入。不過，此類模型的輸入仍然會有所限制，無法直接處理超長文本; 又或是因爲模型參數量龐大，需要不小的記憶體。而這些研究大部份處理的是 Arxivc 和 Pubmed 資料集，雖然壓縮率和書籍章節差不多，但數據量已具有一定規模，而且原文件與摘要重複的詞彙片段也較多。

第二種方法則是利用階層式架構來處理長文本，通常在這類的做法下，長文本可以分成不同層級，例如文件層級和段落層級，將各自層級做完摘要後合併形成最終的摘要。其中 [9] 提出了 Booksum 資料集，並且將輸入文本視爲多個層級：書/章節/段落，以監督式訓練的方式，將所有在摘要任務上具有優異表現的 Transformers 模型，在小說類書籍上生成抽象式摘要，並且展示了在這類資料集上的效能。該篇論文並未更改模型架構，因此可以將其視爲該資料集 BookSum 的基線之一。

[10] 提供了 NovelChaper 資料集，並利用萃取式的方式，將小說章節利用 stable matching 演算法，將章節內容和參考摘要間分別以不同粒度級別的句子組成，一對一對齊，形成萃取式摘要。

[5] 建立了多個轉換器窗口，讓輸入可以被拆分成多個分段，並且以 BIGRU 層來共享不同窗口間的資訊，本研究與其不同的地方是，我們希望在編碼時就可以考量全局資訊，並且以較少的參數量來達到資訊的傳播。

在目前的研究中，大多數的論文都傾向在監督式訓練的條件下對長文件做萃取式或是抽象式摘要任務，只有少數的研究如 [4] 採用非監督式訓練方法。因此在本研究中我們希望探討如何以非監督抽象式摘要的方式來訓練模型，之後再以它作之後模型的基礎。

第四節　句子融合與句子簡化

句子融合 (sentence fusion) [15] 是指將多個互相獨立的句子連貫合併成單一個句子。在句子融合的任務中，通常會希望能減少重複性。而句子簡化 (sentence compression) [27]，則是將句子的長度縮短。換句話說，兩個任務皆在句子級別執行摘要，通常兩者

DOI:10.6814/NCCU202200337

較常使用在多文件的摘要任務。

句子壓縮的方法可以分爲基於刪除和基於抽象的方法。基於刪除方法下，會從從原句中移除部分詞得到壓縮句，至於如何判斷哪些詞彙是可以移除的，會依靠詞性、詞關係、句子架構 (syntactic parser tree) 從中修剪不太重要的詞或子樹。修剪的方式由演算法或學習模型決定。而基於抽象的方法下，則會利用標記過後的資料對 (pair) 在端到端模型上執行訓練，以大規模的資料對來訓練監督式句子壓縮模型。

對於長文本摘要任務而言，通常會將整段長文再拆分成數段文字，我們希望在本研究中能使用技術將上述這 2 個任務加進去模型訓練中，讓模型學習到如何將多個句子/段落的意思揉合進一段文字中。

DOI:10.6814/NCCU202200337

第三章　研究方法

第一節　概述

本文所提出之模型架構分爲兩個部分：訓練時期和推論時期，架構圖如3.1.1 ，圖中的虛線代表可選擇的流程，該模型主要任務是希望在給定輸入長文本 x 的條件下，生成抽象摘要 y。本方法於訓練時期和預測時期分別有不同的模型架構：訓練時期，首先利用多任務學習的方式，利用 2 個子任務: 自監督學習的摘要生成和知識融合任務。自監督學習中所使用的資料，利用的是維基百科的頁面中對於不同字詞的解釋說明文字，以及對於不同類別有對應的主要頁面和其子類別的階層式目錄，利用基因演算法，建立原始文件和其摘要的並行數據 (parallel data)。知識融合任務藉由信息瓶頸理論，學習重要的辭彙資訊，然後再用剛剛推理得知的重要的辭彙資訊生成摘要，期望藉由這一系列的動作讓模型學習到哪些資訊對辭彙之間的文字架構組成，使句子能很好的融合。藉由以上所提及的這 2 個任務，我們得到無監督訓練後的摘要生成模型 (a)。而在監督式訓練設置下，我們稱爲微調時期 (finetune stage)，利用循環式模型架構，利用目標域 (target domain) 下的書本文字做訓練，得到摘要生成模型 (b)。

預測時期，由於 Transformer based 模型無法在單一模型中輸入過長的文字輸入，則利用多層架構來處理，輸入文字在分成幾個段落後，會有以下 2 種做法：萃取式輸入或全文輸入。假設輸入文字是書本的話，則以兩級方式進行摘要，分別在章節和書籍級別上使用 M；如果是輸入是章節的話則只有一級，一個章節可以分成多個段落，摘要模型 M 生成每個段落的摘要再合併形成章節摘要。至於書本的話，整段文字可以切分爲多個章節，而每個章節可以再細分成多個段落；先對每個章節生成摘要，然後根據生成的章節摘要，挑選出最重要的前 k 個，最後，挑選到的章節摘要再餵給摘要模型 M，生成書本摘要。

總的來說，關於第一章所提到的 3 個問題，針對輸入過長這個問題，我們利用推論時期的多層模型和萃取式做法，來間接減少文字資訊以及縮短模型的文字輸入。至

DOI:10.6814/NCCU202200337

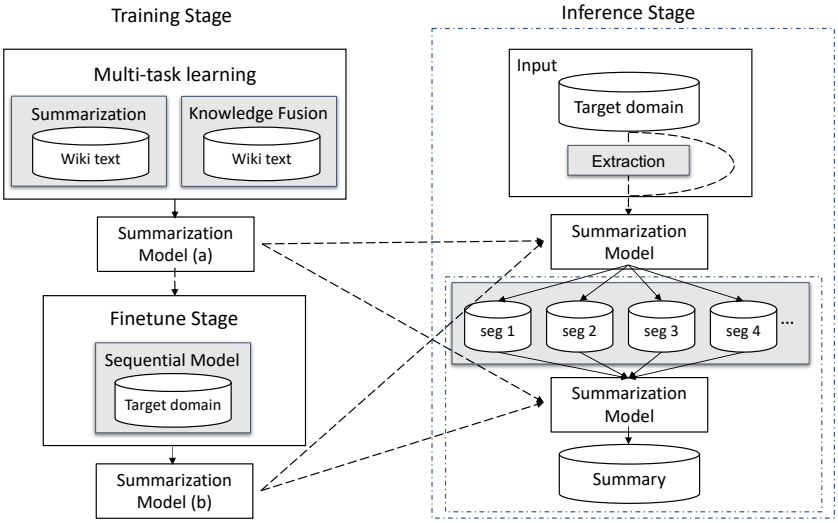


圖 3.1.1 摘要模型架構圖

於高壓縮率這個問題，我們藉由知識融合任務來幫助模型辨認文字中更重要的字詞。至於訓練數據的部分，我們使用維基百科內容仿照書本架構，建構出輸入摘要資料對來幫助模型訓練摘要任務，也就是說我們利用上述提及的不同方法來解決長文本 (書籍) 摘要任務中的困難點。

第二節　摘要任務

一、 概論

我們的目標是在不使用人工手寫摘要的情況下，訓練一個單一非監督式抽象摘要模型。我們希望能仿照書籍文字一樣，擁有多層級的架構，例如：整本書、章節、段落，越高的層級會對所涵蓋的文字內容總結成更高壓縮率、更抽象的摘要，亦即輸入中的不同層級會對應到不同的抽象程度，而我們需要構建出不同抽象級別條件的資料。

定義如下：

$$\hat{y}^d = M(x^d) = \arg\max_{y^d} P(y^d|x^d) \tag{2.1}$$

其中，d 是抽象程度，x^d 是在在抽象程度 d 時輸入的文本，$\hat{y}^d$ 是在抽象程度 d 時的生成摘要。而我們的目的是在給定不同抽象程度的輸入，生成該抽象程度時最高概率的摘要。理想的資料型態應該會是對於不同的抽象級別，會有 (x^d, y^d) 來訓練模型 M。下面內容則會詳述維基百科分類的架構，以及我們如何根據它的分類架構，構建維基

DOI:10.6814/NCCU202200337

百科主題知識圖譜，用於組織維基百科頁面之間的關係後，生成不同級別資料 (x^d, y^d) 構建爲標記實例的新方法。

二、 維基百科目錄架構介紹以及定義

在維基百科中，已經內建好完整的目錄結構，裏面包含每個字詞的介紹和所屬類別 (category)，以幫助使用者查找訊息。以下我們將介紹維基百科架構，以及如何透過它本身的架構對應到不同級別的抽象資料 (下一節)。

對於任意一個維基百科頁面 p 而言，根據維基百科對於版面佈局的定義[1] ，該頁面組成如下：導言章節是指在章節目錄之前的文欄段落，會簡短的介紹該條目的特殊性，章節目錄之後則稱之爲主體，提供該條目的完整信息，而該頁面 p 則會被分類到多個類別。爲了之後方便，我們將導言章節命名爲 $y(p)$，主體命名爲 $x(p)$。對於任意一個類別 v_i 而言，它可能會對應到多個子類別和多個所屬頁面以及多個主要頁面 (main article)，形成一個完整的樹結構；這裡的主要頁面會是該類別的主要詞條的頁面，對於該類別具有代表意義。以“機器學習“爲例，它會有 14 個子類別，例如：人工神經網路、深度學習等，以及 120 個被分類在該類別的相關頁面，其中也包含“機器學習“，也就是這個類別的主要頁面。

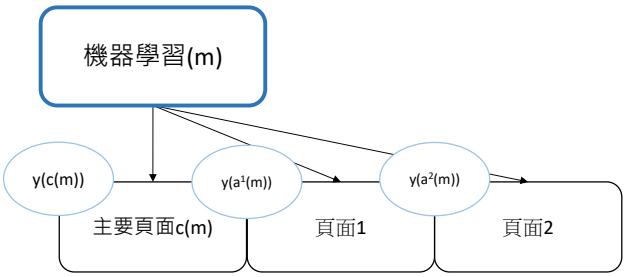
根據上面介紹的維基百科的分類架構，我們可以將其視爲一個有向的知識圖譜 $G = (V, E)$，其中 V 是所有被維基百科定義的類別，而 E 則定義爲類別間特殊的關係，爲了模擬書本中章節與全文的階層關係，因此將兄弟節點 (sibling node) 與子節點 (child node) 這兩種關係納入 E 中，並且分別命名爲 E_s、E_c。針對不同抽象級別 d，X^d 和 Y^d 分別代表當抽象級別爲 d 時的輸入和摘要集合。

假設 v_i 爲維基百科裡其中一個類別，它會對應到一或多個主要頁面[2]，其主要頁面中的文字爲 $c^m_{v_i} \in C(v_i)$，而其他所屬頁面的文字爲 $a^n_{v_i} \in A(v_i)$。令 v_j 爲維基百科裡的另一個類別，$v_i, v_j \in V$ 且 $v_i != v_j$，我們定義對於 v_i 這個類別，會有以下 2 種關係：集合 $Children(v_i) = \{v_j | v_j : (v_i, v_j) \in E_c\}$，也就是說 $v_j \in Children(v_i)$ 會是 v_i 的子類別。同樣的，如果 v_i 和 v_j 同屬於一個類別，則該集合 $Sibling(v_i) = \{v_j | v_j : (v_i, v_j) \in E_s\}$ 包含了所有 v_i 的兄弟節點。

[1] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Layout#Body_sections

[2] 這邊我們在爬取資料時，只考慮到維基百科中有定義它有主要頁面的類別，若沒定義，則不將其列入考慮 (d=1,2 的摘要集合)

DOI:10.6814/NCCU202200337



$y(a^i(m)) \in [x]^1_m$, i = {1,2} : 類別c所屬頁面的導言章節
$y(c^j(m)) \in Y^1$, $j = \{1\}$: 類別c所屬主要頁面的導言章節

Level 1

圖 3.2.2 抽象級別爲 0 時，訓練資料來源結構示意圖

三、 實例生成

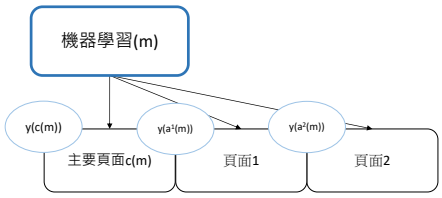
我們這節將描述如何根據上節的定義，從 X^d 和 Y^d 來生成假資料對 (x^d, y^d)，其中 $x^d \in X^d, y^d \in Y^d$，當抽象級別 d 越高時則該輸入文字對應到的摘要越抽象，下面將依照抽象級別 d 來一一作介紹。

當抽象級別 $d = 0$ 時，會對應到 X^0, Y^0 兩個集合，延續 [13] 的設定，以維基百科的頁面文字 p 作爲假資料來源。X^0 會由頁面的主體 $x(p)$ 所組成，Y^0 則會由頁面的導言章節 $y(p)$ 組成。也就是說我們會把 wiki 頁面的第一段視爲摘要，總結下面的主體文字，建構 $d = 0$ 抽象級別的僞標記訓練實例，如示意圖3.2.2。

而對於更高的抽象級別 d，我們的目標是爲 y^d 安排更長、更詳細、更具體的文檔來安排 x^d 的內容。爲了實現這個目標，我們利用維基百科對於每個類別所形成的階層結構，在給定類別 v_i，$C(v_i)$ 作爲介紹主題 v_i 的主要頁面，$d = 1$、$d = 2$ 的輸入分別會從 $[x]^1_{v_i} = \{y(C(v_j))|v_j \in Sibling(v_i)\}$ 和 $[x]^2_{v_i} = \{y(A(v_j))|v_j \in Children(v_i)\}$ 這兩個集合，通過檢索集合內相關維基百科頁面的內容 $[x]^d$，將 $[x]^d$ 內所有句子挑選以及重新排序，來自動組合對應級別的 x^d，而 y^d 則爲 $y(C(v_i))$。以機器學習這個類別爲例 (以 m 代替)，圖3.2.3可見該類別下所有頁面 (不含主要頁面) 的導言章節內容 $y(A(m)) = [x]^1_m$ 經過處理後得到 $x^1 \in X^1$；圖3.2.4可見該類別下所有子類別下的頁面 (不含主要頁面) 的導言章節內容 $y(C(n)) = [x]^2_m$, $\forall n \in Children(m)$ 經過處理後得到 x^2。

下面則會詳述我們如何將集合 $[x]^d$ 轉換爲一個文檔 x^d，該文檔應該是摘要模型的

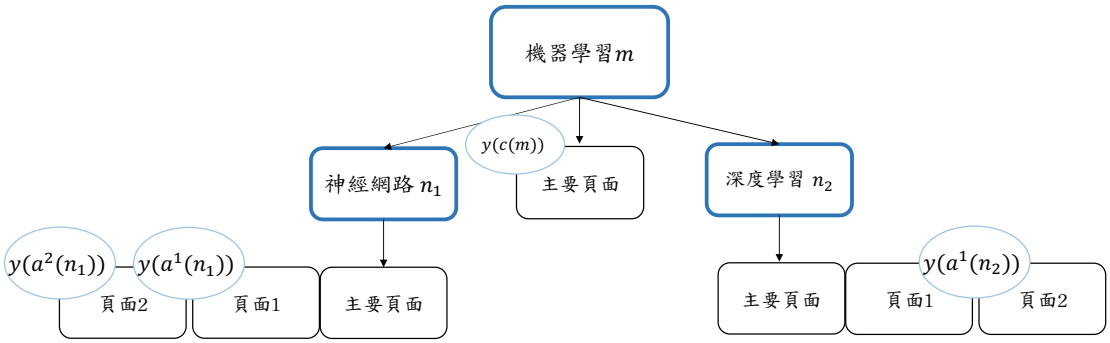
DOI:10.6814/NCCU202200337



$y(a^i(m)) \in [x]_m^1$, i = {1,2}：類別m所屬頁面的導言章節
$y(c^j(m)) \in Y^1, j = \{1\}$: 類別m所屬主要頁面的導言章節

Level 1

圖 3.2.3 抽象級別爲 1 時，訓練資料來源結構示意圖



$y\left(a^i(n_j)\right) \in [x]_m^2$：類別m的子類別下所屬頁面的導言章節
$y(c(m)) \in Y^2$：類別m所屬主要頁面的導言章節

Level 2

1

圖 3.2.4 抽象級別爲 2 時，訓練資料來源結構示意圖

DOI:10.6814/NCCU202200337

輸入以生成 y^d。x^d 的內容最理想的是滿足以下四個標準。

- x^d 的長度比 y^d 長但比 L 短。
- x^d 的內容比 y^d 更詳細更具體。
- x^d 的內容可以盡量覆蓋 y^d 的信息。
- x^d 的內容按照類似於 y^d 的順序組織。

我們將集合 $[x]_v^d$ 當作輸入文檔的候選集合 $S = \{s_1, s_2, ...\}$，並從 S 選擇一個子集 $\hat{S} \subseteq S$，通過演算法以一定的順序排列拼接形成 x^d。

對於這個優化問題，我們基於以前的用於尋找元素的最佳排列的遺傳算法的研究，例如 [19] 將遺傳算法應用於旅行商問題 (TSP)，提出了一種遺傳算法來搜索最優的 x^d，它是 S 的一個子集的排列的句子串聯。

在我們的例子中，我們的遺傳算法的適應度函數定義了優化的目標，包括以下兩個條件。

- x^d 的長度不超過 L。
- x^d 和 y^d 之間的 ROUGE 分數 [12] 越高越好。

其中度量 ROUGE-L 廣泛用於評估摘要模型。作爲 ROUGE 分數的一種變體，ROUGE-L 通過使用它們的最長公共子序列來測量兩個字符串之間的重疊。ROUGE-1 和 ROUGE-2 則使用 n-gram 測量重疊相比，而 ROUGE-L 對順序的一致性更敏感。我們使用了 ROUGE-L 和 ROUGE-1 的加權總和作爲評估，這樣，得到的 x^d 不僅是想要的長度，而且還優化了以類似的順序表達 y^d 的信息。

我們在四個 epoch 中以 300 個排列的池 (pool) 大小執行遺傳算法。使用有序列表的交叉 [21] 和通過兩個元素隨機交換的變異。最後，選擇具有最高 ROUGE-L 和 ROUGE-1 加權分數的排列以形成 x。3.2.1展示了主題“歸納推理”在兩個層面的訓練實例。

我們在不需要人工註釋的情況下構建了不同抽象級別的訓練實例。先前的研究從維基百科構建類似的長文本摘要數據集，如 [13] 和 [32]，它們都通過爬取相關文檔來收集源文本以構建多文檔摘要數據集. 與這兩項工作不同，我們的方法旨在爲具有高密度抽象的單個文檔摘要構建數據集。

使用我們的數據集，我們的摘要模型 M 能夠通過暴露於不同抽象級別來學習特定表達式和廣泛表達式之間的關係，獲得交叉熵損失 $\mathcal{L}_m$。

DOI:10.6814/NCCU202200337

Level	Input (x)	Summary (y)
$d = 0$	Types. Generalization. A generalization proceeds from a premise about a sample to a conclusion about the population. The observation obtained from this sample is projected onto the broader population. For example, say there are 20 balls—either black or white—in an urn . To estimate their respective numbers, you draw a sample of four balls and find that three are black and one is white. [⋯]	Inductive reasoning is a method of reasoning in which the premises are viewed as supplying some evidence, but not fullassurance, of the truth of the conclusion. Many dictionaries define inductive reasoning as the derivation of general principles from specific observations, although there are many inductive arguments that do not have that form. Inductive reasoning is distinct from deductive reasoning. If the premises are correct, the conclusion of a deductive argument is certain; in contrast, the truth of the conclusion of an inductive argument is probable, based upon the evidence given.
$d = 1$	A generalization is a form of abstraction whereby common properties of specific instances are formulated as general concepts or claims. Noam Chomsky and Hilary Putnam attended some of the lectures on which the book is based as undergraduatestudents at the University of Pennsylvania, leading to a lifelong debate between the two over the question of whether the problems presented in the book imply that there must be an innate ordering of hypotheses. [...]	
$d = 2$	This is a statement most people would consider incorrect, due to emergence, where the whole possesses properties not present in any of the parts. Some scholars classify cherry-picking as a fallacy of selective attention, the most common example of which is the confirmation bias. They may alternatively have to do with the presence of singularities. [...]	

表 3.2.1 類別“歸納推理“的僞標記訓練實例

順帶一提，我們的方法可以通過探索類別的孫子甚至曾孫子來擴展到更高的抽象級別訓練實例。

我們通過 mediaWiki 收集所有維基百科頁面和分類結構，[3] 而根主題則取自官方主題列表，[4]，我們爬取每個類別以及下 1 層的節點類別，以及各自類別下的維基百科頁面，並使用正則表達式進一步清理所有抓取的文本。在收集資料的過程中，我們排除了所有有關書籍的 Wikipedia 頁面，避免將參考摘要混入訓練數據。

[3] https://www.mediawiki.org/wiki/MediaWiki
[4] https://en.wikipedia.org/wiki/Category:Main_topic_classifications, https://en.wikipedia.org/wiki/Wikipedia:Contents/Categories

DOI:10.6814/NCCU202200337

第三節　知識融合任務

由於書籍摘要需要極高的壓縮率，我們希望鼓勵模型提綱挈領，擁有改寫的能力，可以將冗長且太詳盡的內容以簡短的概要來描述。

我們將之前的著作 [26] 的概念融合進我們的多任務模型，[26] 從隨機變量 X(亦即輸入) 中編碼最有信息量的細節 Z，通過信息瓶頸 (information bottleneck) 的理論，希望能更好地預測隨機變量 Y，公式如下：

$$I(Z, X) - \beta I(Y, Z) \tag{3.1}$$

其中，$I(\cdot)$ 衡量了 2 個隨機變量間的相互資訊 (mutual information)，β 爲常數係數。

假設給定輸入爲一個句子 $x = (x_1, x_2, ..., x_n)$ 共有 n 個字，而 y 會是該任務的標的，z 是 x 經過模型後得到的特徵，我們希望借由將最重要的資訊編碼進 z，讓模型學習到哪些文字應該被遮罩，進而學習該辭令，然後幫助模型更好的學習 y。爲了強迫模型學習輸入的特徵，故對所需要用到的特徵和輸入解耦 (m 和 x 獨立)，令 z 等於 m 和 x 的元素積 (element-wise product)，$m = (m_1, m_2, ..., m_n)$ 會是和 x 長度個的 Bernouli 分配，$m_i \sim Bernouli(\pi)$，也就是說對於輸入序列，每個詞條都會是 π 的機率被編碼進 z，而 π 控制有多少資訊量會留下來做訓練。該項任務的目標函數在算式 3.2。

$$\mathcal{L}_k = E_{m \sim p(m|x)}[-\log(q(y|m))] + \beta \sum_i \mathrm{KL}[p(m_i|x)||r(m_i)] \tag{3.2}$$

其中，r 是 m_i 近似先驗分配，KL 散度限制模型學習到的 $p(m_i|x)$ 應該和先驗分配之間的差異不能太大，而加號左邊中的 $q(y|m)$ 則會是 $p(y|m)$ 的近似值。由於 m_i 不能偏微，因此需要使用重參數化 (re-parameterization) gumble-softmax 的方法。我們的方法會利用一個編碼器模型和 (encoder model) 來編碼 p 和一個編碼器-解碼器模型（encoder-decoder model）架構來編碼 q，並固定上面 2 個模型的輸入的詞向量 (embedding)。我們使用句子融合的資料集 WikiSplit [3] 作爲我們訓練該項子任務的資料，期望模型學習到如何壓縮並且抽象化句子。

DOI:10.6814/NCCU202200337

第四節　生成模型

我們的生成模型基於 T5 [18]，性能最好的文本生成 Transformer 模型之一。而多任務學習中的摘要任務是使用 $d = 0$、$d = 1$ 和 $d = 2$ 級別的僞標記數據進行訓練，再加上輔助任務：知識融合這項任務，通過加權係數權重進一步與主模型合併：

$$\mathcal{L} = w_m \mathcal{L}_m + w_k \mathcal{L}_k \tag{4.1}$$

得到我們最終的損失函數和微調模型。

第五節　推論模型

一、 概論

由於在我們的架設中，輸入會是長文件，因此針對過長的輸入我們會有以下 2 種處理方式:

- 萃取式

  最直接的做法也就是抽取長文中最重要的段落作爲模型輸入，各自生成摘要完後，最後連接起來形成最終的摘要。利用公式，選取分數前 k 高的段落，這些段落可以視爲該章節中最重要且最需要被涵蓋到的部分。其中，e_{ij} 在這裡定義爲每個段落與其他同章節但不同段落的相似程度，這邊的相似程度以 cosine similarity 做計算。

$$\text{centrality}(s_i) = \lambda_1 \sum_{j<i} e_{ij} + \lambda_2 \sum_{j>i} e_{ij} \tag{5.1}$$

  然而此類作法: 不管是先挑選段落再生成摘要，又或是 [9]，從所有段落中各自生成摘要後再進行挑選與作爲最後的摘要，兩個做法都忽略了考慮各個段落間的關係。

- 循環式

  比起上面的做法，我們多考慮了前後段文意的相關性 (sequential)，利用 RNN 中循環 (recurrent) 的概念，讓模型能吃進多個輸入，模型架構圖如3.5.5。由於訓練時，此類模型因爲記憶體的限制，所以大致上只能放進長文本中的 k 個段落作爲

DOI:10.6814/NCCU202200337

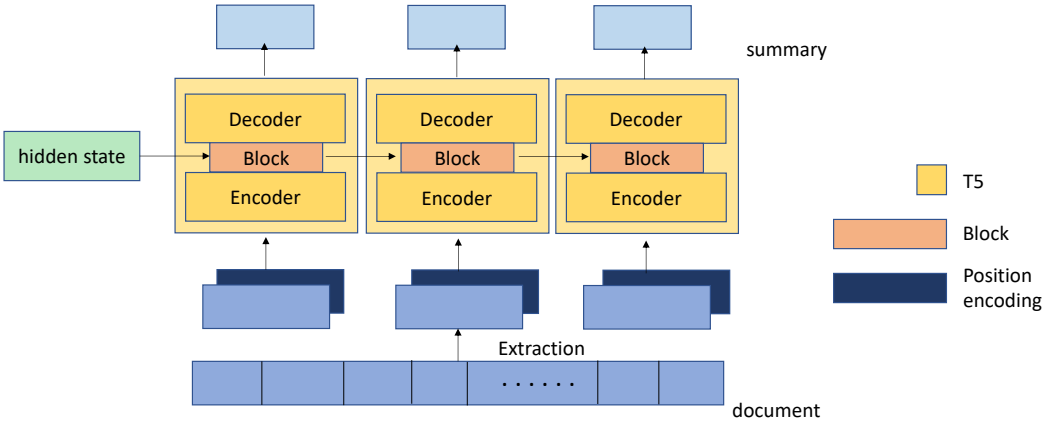


圖 3.5.5 此示意圖以 k = 3 爲例，利用萃取式和循環式架構生成摘要

輸入，而後依序輸入進我們已經預訓練好的模型，生成 k 個摘要再合併，至於完整的架構會在下一段詳述。

二、 循環式模型架構

(一) 模型架構

模型架構如圖3.5.5，輸入爲多個文本，模型則是利用已經微調訓練後的模型，共享參數，期望將狀態在自身網路中循環傳遞，學習長期時間關聯和單位之間潛在的依賴關係。

由於模型會有多個不同的輸入，因此我們在輸入向量加上了輸入位置訊息，也就是它在這篇文檔中的位置。同時，因爲輸入的個數也會不同，在訓練時可能會有些位置訓練不足，難以梯度反向傳播，因此我們最後使用位置編碼 (position-encoding) [25] 來做爲位置向量。

公式如下：

$$\begin{aligned} \mathrm{PE}_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ \mathrm{PE}_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{5.2}$$

由此每個位置有一個唯一的表示，並且不同位置之間的關係可以通過他們位置編碼間的仿射變換來獲得。

模型部分增加了新的模塊 (block)，將上一個狀態資訊傳遞到後面，讓隱藏狀態

DOI:10.6814/NCCU202200337

(hidden state) 能涵蓋全部的資訊，設計如表5.3，令長文 $= [p_1, p_2, ..., p_k]$，當處理到 p_i 時，會將模型的編碼器編碼過後的 p_i 的隱藏狀態 s_i 和 p_{i-1} 的隱藏狀態 (prev hidden state) s_{i-1} 連接 (concat) 起來後投影再層標準化 (layer normalization)，最後餵給 p_{i-1} 的模塊，隱藏狀態的起始值爲零矩陣。損失函數仍維持使用交叉熵函數，整個訓練過程會從 p_1 跑到 p_k 後再反向傳遞回梯度，如果文件段落長度小於 k，則補上空字串到 k 個段落爲止; 推論過程則是會依序在 p_i 生成各自的摘要後再合併成爲最終的摘要。

$$
\begin{aligned}
s_i &= \text{EncoderModel}(p_i) \\
s_i &= \text{nn.Linear}([s_i; s_{i-1}]) \\
s_i &= \text{LayerNormalization}(s_i) \\
\text{vocab dist.} &= \text{DecoderModel}(s_i)
\end{aligned}
\tag{5.3}
$$

下面會介紹當使用非監督式訓練和監督式訓練時不同的架構配置。

(二) 非監督式訓練

在非監督式訓練方法中，輸入的文字會是抽取過後的長文件段落，由於我們沒有眞值標籤，因此會拿段落自身的文字作爲訓練時的標籤。

(三) 監督式訓練

在監督式訓練方法中，則有以下 2 種配置：

- 萃取式輸入

  在此配置中，輸入的文字會是抽取過後的長文件段落，標籤則是參考摘要，由於單一的章節可能會有多種不同來源的摘要，這裏是隨機抽樣來作爲當下章節的標籤。

- 全文輸入

  萃取式輸入的最大缺陷在於當我們挑錯段落時，就不可能產生適當的摘要，因此我們設計此機制把全文作爲輸入，期望讓模型自己挑選正確的段落，而在此架構下，原本的模塊5.3會增加一個二分類問題，判斷是否在這個段落生成摘要，如圖3.5.6。也就是說在第 i 個段落時，s_i 會做兩次投影後再過 sigmoid 函數，得到機率，如果該機率大於門檻值 0.5，則生成摘要，反之則不，損失函數爲二值交叉熵函數。

  爲了得到二分類問題的標籤，我們需要做資料對齊：將輸入文字和參考摘要兩者的分段去做匹配，看看每句的摘要分段會來源於哪些章節段落。匹配演算法用的

DOI:10.6814/NCCU202200337

是混合整數規劃 (mixed integer programming)，這裡我們假設每個摘要通常會揉合多個文章段落的文意，而一個章節段落只會對應到一個摘要分段，m 爲章節段落的個數，n 爲參考摘要的分段個數，W_{ij} 對應的是第 i 個段落和第 j 個參考摘要分段的分數 - Rouge-1 和 Rouge-2 的加權和，產生限制式如下：

$$max \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} Nij \tag{5.4a}$$

$$s.t. \sum_{i=1}^{m} N_{ij} = 1 \quad \forall j \tag{5.4b}$$

$$\sum_{j=1}^{n} N_{ij} >= 1 \quad \forall i \tag{5.4c}$$

$$\sum_{j=1}^{n} N_{ij} <= min(abs(m-n)+1, n/2) \quad \forall i \tag{5.4d}$$

式5.4b 限制一個章節段落只會對應到一個摘要分段，式5.4c 和5.4d 限制一個摘要分段對應到章節段落的數量區間。

此時 N_{ij} 會是個 0-1 矩陣，也就是匹配的結果。根據這個結果，我們會再對回輸入的段落，由於一個摘要會對應到多個段落，因此我們以該摘要分段對應到最後一個段落當作分類的眞值，而其他沒被對應到的段落則設爲 0，不輸出摘要。

舉個例子，假設現在有 5 個章節，和 3 句摘要句子要對齊，已知配對結果爲:$(1,1),(3,2),(4,2),(5,3)$，在我們的設定下：二分類問題的標籤會等於 $[1,0,0,1,1]$。

訓練時期由於硬體的限制，於是我們依據文件的長度，在此區間隨機挑一數字到它後面的 k 個段落作爲輸入訓練，如果該段落的二分類眞值標籤爲 0 時，則忽略該段落的交叉熵損失；推論時期則輸入全文，將每個段落的機率排序，取前 k 高的摘要後合併，產生最後的摘要。

在上面兩個設置中，訓練時會混入標籤爲段落自身的文字的資料，讓模型生成的摘要和輸入相關，而不會讓模型因爲資料過少，過度訓練後集中注意在某幾個詞條上。

DOI:10.6814/NCCU202200337

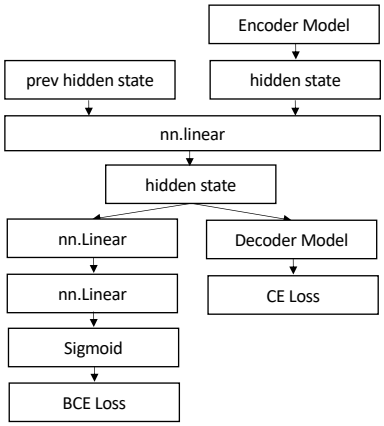


圖 3.5.6 模塊架構圖

DOI:10.6814/NCCU202200337

第四章　實驗

第一節　資料集

一、 介紹

長文件中輸入和摘要的長度差異極大，因此和一般的資料集相比，在固定候選摘要的長度下，除了去探討不同方法所產生的摘要品質好壞，更需要考量更多的方面，例如: 資料壓縮比-得知資料被壓縮的程度；覆蓋率-可以看出該篇參考摘要的文字有多少比例來源於原始文件，從這兩個指標我們可以約略知道該資料集的困難程度。

下面列出的指標算法參考 [6]：$A =< a_1, a_2, ..., a_n >$ 爲輸入文件，$S =< s_1, s_2, ..., s_m >$ 爲摘要，word tokenizer 使用的是 nltk 套件。該篇論文將 $F(A, S)$ 定義爲在 A 和 S 中共同出現的詞令序列，根據貪婪算法提取出重疊的詞彙片段，做法如下圖4.1.1：

$$\begin{aligned} \text{覆蓋率} &= \frac{1}{|S|} \sum_{f \in F(A,S)} |f| \\ \text{壓縮率} &= \frac{|A|}{|S|} \end{aligned} \tag{1.1}$$

覆蓋率定義爲摘要中提取文章內出現的片段的百分比; 壓縮率定義爲文章和摘要之間的字詞比。

DOI:10.6814/NCCU202200337

```
function F(A, S)
    F ← ∅, ⟨i, j⟩ ← ⟨1, 1⟩
    while i ≤ |S| do
        f ← ⟨ ⟩
        while j ≤ |A| do
            if s_i = a_j then
                ⟨i′, j′⟩ ← ⟨i, j⟩
                while s_{i′} = a_{j′} do
                    ⟨i′, j′⟩ ← ⟨i′ + 1, j′ + 1⟩
                if |f| < (i′ − i − 1) then
                    f ← ⟨s_i ··· s_{i′−1}⟩
                j ← j′
            else
                j ← j + 1
        ⟨i, j⟩ ← ⟨i + max{|f|, 1}, 1⟩
        F ← F ∪ {f}
    return F
```

圖 4.1.1　集合 $F(A,S)$ 僞代碼

DOI:10.6814/NCCU202200337

二、 資料集介紹

下面會介紹本篇所使用的資料/資料集：

- wiki-summ

  第三章中我們藉由維基百科的架構，建立了摘要資料對。我們從 486 個根類別中獲得了大約 70,000 對、$d = 1$ 時的 64,000 對和 $d = 2$ 時的 13,000 對。

- NovelChapters

  資料集來源於 [10](以下簡稱 NovelChapters)，由於有些網頁已經失效或者更動，外加論文作者並沒有附上完整的資料清理過程，因此資料筆數會和原論文提供的數據有滿大的出入，故下面的實驗結果並不會放上該論文的數據做比較。該數據集包含小說類的章節以及對應摘要所形成的資料組，其中小說出處來源於古騰堡計劃[1]，而小說的摘要則從一些網路上公開的書摘網站爬取而得，因此每一個章節可能會有多個不同來源的摘要。統計數據如下表4.1.1。

  NovelChapters 中資料集裡共有 3535 個章節，也就是 8990 對章節/摘要資料組，壓縮比高達 18.31，足以顯示這是個具有挑戰性的任務，因爲我們需要更準確地捕捉文本的關鍵，而覆蓋率有 0.79，比起 [6] 所列的資料集 CNN/DM 或者是 Newsroom 都更低，也就表示文件中的資訊含有更多冗餘的文字，而應該被捨棄。

- BookSum

  資料集來源於 [9](以下簡稱 BookSum)，該數據集將過往數據與書籍摘要相關的論文 (其中也包含了 NovelChapters)，統合彙整成現今最完整且最詳細的資料集，裡面包涵了書本/章節/段落層級以及前兩個層級對應到的摘要 (以下簡稱 BookSum(chapter)、BookSum(book))。統計數據如下表4.1.1

  BookSum(chapter) 章節層級與 NovelChapters 的覆蓋率雖然相差不多，而壓縮率卻更低了一些，因此我們預期 BookSum(chapter) 的效能會比 NovelChapters 來的好一些。而 BookSum(book) 書本層級由於文件字數過多，壓縮率過高，可以看的出這會是個極艱難的任務。

[1] https://www.gutenberg.org/

DOI:10.6814/NCCU202200337

Dataset	NovelChapters	BookSum(chapter)	BookSum(book)
資料筆數	8990	12293	436
壓縮率	18.31	15.97	126.22
覆蓋率	0.79	0.78	0.89
輸入文件單字數	4755.17	5101.88	112885.15
摘要單字數	405.05	505.42	1167.20

表 4.1.1 資料集相關數據

第二節 實驗評估指標

本研究使用自動評量指標來評估生成摘要，採用摘要研究常見之 ROUGE 和 METEOR 來衡量候選摘要與參考摘要的相似程度。

- Rouge

  摘要任務主要希望能多預測出參考摘要裡的字詞，提高召回率 (Recall)，而 Rouge [12] 衡量的是候選摘要與參考摘要兩者重疊的 n-gram (overlap) 以及變形。Rouge-N 的召回率會等於參考摘要與候選摘要重疊的 N-gram 個數除以參考摘要中 N-gram 的個數。摘要任務主要使用的指標會有以下 3 種：Rouge-1(R-1), Rouge-2(R-2), Rouge-L(R-L)。其中 Rouge-1, Rouge-2 兩者分別指的是 unigram, bigram 的重複比例，而 Rouge-L 則是將 n-gram 的概念替換爲 LCS(最長公共子序列)，召回率會等於最長公共子序列的長度除以參考摘要長度。實作用的是 PyROUGE 套件[2]計算 Rouge 分數。

- METEOR

  METEOR 同時考慮了的準確率和召回率兩個面向，兩者取調和平均而最終得出測度。METEOR 也包括其他指標沒有發現一些其他功能，如同義詞匹配 - 用 WordNet 等知識源擴充了一下同義詞集，同時考慮了單詞的詞形（詞幹相同的詞也認爲是部分匹配的，也應該給予一定的獎勵，比如說把 likes 翻譯成了 like）

  在評價句子流暢性的時候，用了 chunk 的概念期望候選摘要和參考摘要能夠對齊的一致，chunk 的數目越少意味著每個 chunk 的平均長度越長，也就是說兩者的語序越一致。

[2] https://github.com/andersjo/pyrouge/tree/master/tools/ROUGE-1.5.5

DOI:10.6814/NCCU202200337

下面的實驗都會以這 2 個指標來當作評量摘要的標準，兩者的分數越高，則顯示候選摘要具有一定的品質。

第三節　模型、超參數以及實驗配置

一、 生成模型

T5 模型使用 Hugging Face 框架，使用 t5-small 的預訓練參數設置爲起始。超參數如下：由於我們在訓練最一開始的多任務模型時，兩項任務都來源於維基百科，因此我們將權重衰減設置爲 0.1，以防止我們的模型過度擬合維基百科，預熱步驟爲 1,000 步，學習率爲 1e-5，使用 Adam optimizer 和 linear scheduler 來優化。在知識融合任務中使用的 gumble-softmax [7]，我們設置 $\tau = 0.7$ 以獲得更平坦的分佈，並設置 $\pi = 0.2$ 使模型嵌入較少的資訊量。兩項任務在損失函數的係數都設成 1.0。輸入會抽取前 k=6 個章節段落，根據參考摘要平均字數來平均分配每個段落的候選摘要長度。

二、 循環模型

模型權重是拿生成模型作爲起始，學習率爲 1e-3，使用 RMSprop optimizer 和 linear scheduler 來優化。在非監督式和監督式訓練設置下，如果是萃取式方法，模型使用的是 SentenceTransformers 的預訓練模型，輸入一樣抽取前 k=6 個章節段落。所有循環模型下的方法，生成摘要函式的超參數都設置成一樣的超參數。而在監督式訓練和使用全部輸入的機制時，由於二分類問題，兩個類別的比例大約在 0.7:0.3，而且訓練時期只會取隨機起始的段落區間，會導致類別分佈更加不均，因此在二值交叉熵函數，類別係數設爲 [1,4]，也就是說我們會特別懲罰當眞值類別爲 1 但卻預測錯誤; 並且在損失函數裡，二分類任務和摘要任務的係數分別爲 [1,0.3]。

第四節　模型效能

一、 基礎模型

- Lead-3

  Lead 3 是一個非常簡單且廣泛使用的模型，它從輸入中提取前三個句子作爲摘要，常作爲多數摘要任務的強大基線 [31]，例如新聞摘要生成。由於我們任務中的摘要平均字數會大約等於前 3 句的字數，因此這裡依然用前 3 句來當作參考效能。

DOI:10.6814/NCCU202200337

- Oracle

  Oracle 也是摘要任務中常見的衡量指標之一，理論上可以視爲萃取式方法下的眞值標籤。然而在多數的摘要任務中，通常都是以人工手寫的摘要作爲標準答案，因此會將參考摘要和輸入作依照不同粒度 (句子/詞彙等級) 對齊，$oracle_{fragment}$ 的做法和 $oracle_{sentence}$ 分別參考 [6] 和參考 [14] 兩篇的方法。oracle 作法可以視爲該資料集的效能上限。

- $T5_{zero\ shot}$

  我們的摘要模型基於 T5，利用已經預訓練好的 T5-small(60M parameters) 做後續的微調實驗。因此，我們與零樣本模型進行了比較。

- 生成模型

  爲本篇第三章第一節到第四節中結合摘要和知識融合任務，訓練於維基百科文字所對應的模型。

從表4.4.2和4.4.4可以看出我們的生成模型在 NovalChapters 和 BookSum(book) 在 Rouge 和 METEOR 的分數上都贏過基線 Lead3 和零樣本模型，而 BookSum(chapter) 大概率是因□沒有調的很好，所以和零樣本模型效能差不多。而從 Oracle 的效能可以看出這 3 個文本雖然摘要中有不小的比例的文字片段是來源於原文，但是零散地分散在各個文句中，因此如何將零散的資訊彙整也是一個很重要的問題。

Method	R-1	R-2	R-L	METEOR
Lead-3	0.1521	0.0204	0.0907	0.0917
$oracle_{fragment}$	0.5127	0.2484	0.4010	0.7130
$oracle_{sentence}$	0.2969	0.0863	0.2031	0.0770
Unsupervised Abstractive Summarization				
$T5_{zero\ shot}$	0.2160	0.0257	0.1089	0.1453
生成模型	**0.2475**	**0.0343**	**0.1238**	**0.1591**

表 4.4.2　資料集 NovelChapters 的基線模型效能

DOI:10.6814/NCCU202200337

Method	R-1	R-2	R-L	METEOR
Lead 3	0.1956	0.0251	0.0994	0.1354
$oracle_{fragment}$	0.7512	0.5105	0.7512	0.6861
$oracle_{sentence}$	0.3036	0.0770	0.2027	0.1620
Unsupervised Abstractive Summarization				
$T5_{zero\ shot}$	**0.1886**	0.0336	0.0938	**0.2242**
生成模型	0.1850	**0.0347**	**0.0957**	0.2144

表 4.4.3 資料集 BookSum(chapter) 的基線模型效能

Method	R-1	R-2	R-L	METEOR
Lead 3	0.0630	0.0088	0.0431	0.0277
$oracle_{fragment}$	0.8478	0.7014	0.8477	0.8222
$oracle_{sentence}$	0.0698	0.0132	0.0543	0.0232
Unsupervised Abstractive Summarization				
$T5_{zero\ shot}$	0.1897	0.0234	0.0952	0.0943
生成模型	**0.2248**	**0.0373**	**0.1172**	**0.1021**

表 4.4.4 資料集 BookSum(book) 的基線模型效能

DOI:10.6814/NCCU202200337

二、 消融實驗

我們將訓練生成模型的兩個任務分別移除，探討不同任務對於生成模型的影響。從表4.4.5中可以看出 NovalChapters 中，只訓練摘要任務就會有一定的表現，在加上知識融合任務後，Rouge-1 從 0.2239 上升到 0.2259，而 METEOR 則從 0.1382 上升到 0.1402，兩個指標都有顯著的提升，而 Rouge-2 和 Rouge-L 則相對略微下降。這也符合知識融合任務的性質：簡化句子，提升句子中的重要字詞的影響力

Method	R-1	R-2	R-L	METEOR
摘要任務	0.2434	0.0233	0.1235	0.1560
知識融合任務	0.2100	0.0273	0.1125	0.1273
生成模型	**0.2475**	**0.0343**	**0.1238**	**0.1591**

表 4.4.5 資料集 NovelChapters 的消融實驗

而從表4.4.6中可以看出在 BookSum(chapter) 中，融合了摘要任務和知識融合任務這兩項任務後，在 ROUGE 和 METEOR 上都有了顯著的提升。表4.4.7中顯示 BookSum(book) 則是在摘要任務模型表現較好。

因此我們最後的結論是：整體來講，多任務學習在處理抽象層級較低的文件時，對摘要任務是有一定幫助的，而後續的實驗，我們將採用生成模型作爲循環模型實驗的起始。

Method	R-1	R-2	R-L	METEOR
摘要任務	0.1850	0.0347	0.0957	0.2144
知識融合任務	0.1804	0.0333	0.0932	0.2118
生成模型	**0.2116**	**0.0375**	**0.1075**	**0.2211**

表 4.4.6 資料集 BookSum(chapter) 的消融實驗

DOI:10.6814/NCCU202200337

Method	R-1	R-2	R-L	METEOR
摘要任務	0.**2309**	0.0370	**0.1216**	**0.1111**
知識融合任務	0.1676	0.0314	0.0981	0.0708
生成模型	0.2248	**0.0373**	0.1172	0.1021

表 4.4.7　資料集 BookSum(book) 的消融實驗

三、 循環模型實驗

NovelChapters 效能於下表4.4.8，在非監督式的訓練設置下，可以看到如果訓練標的只使用原本的輸入自己本身，會因爲模型架構多編碼了全局的資訊，而使得 Rouge-L 有略微提升，達到 11.81%，然而以它爲標的卻會有下面這個問題，模型難以理解當前輸入與下一個輸入間的關係，因此當傳遞前面的資訊時，其中的噪音也跟著增加，所以可以看到 Rouge-1 和 Rouge-2 明顯的下降了不少。

至於監督式訓練的設置下，效能有明顯的上升; 而全文輸入的效能在 Rouge-1 和 Meteor 上皆高於萃取式輸入。也就是□利用資料對齊的方式，讓更多的資訊量能被輸入進模型，而不局限於萃取後的段落。

Method	**Text**		**R-1**	**R-2**	**R-L**	**Meteor**
	input	target				
生成模型						
非監督式	萃取式段落	-	0.2475	0.0343	0.1238	0.1591
循環式模型						
非監督式	萃取式段落	段落本身	0.2423	0.00344	0.1386	0.1510
監督式	萃取式段落	參考摘要	0.2416	0.0393	0.1313	0.1340
監督式	全部段落	對齊過後的摘要片段	0.2514	0.0339	0.1146	0.2055

表 4.4.8　資料集 NovelChapters 循環模型效能

BookSum(chapter) 效能於下表4.4.9，在非監督式的訓練設置下，如果訓練標的是原本的輸入自己本身，在 Rouge-1 和 Rouge-L 有提升，然而因雜訊過多，Rouge-2 和 Meteor 下降，生成的摘要可能有些字是錯的。至於監督式訓練的設置下，萃取式輸入效能同樣在 Rouge-1 和 Rouge-L 有明顯的上升，然而依舊無法避免雜訊過多的問題。全

DOI:10.6814/NCCU202200337

文輸入因爲使用的標的是對齊後的摘要片段，所以在 Meteor 上會比其他兩個設置來得好；然而在 Rouge 上低於萃取式輸入。我們猜測有可能的問題在於: 這裡的二分類問題並不是眞正的對錯問題，而像是閘門開關，藉由開關決定什麼時候才要將連續的段落區間寫成摘要，因此當推論的時候，有時候會遇到極端的例子，多個段落只在少許段落甚至最後一個段落才輸出摘要，導致效能不好。

Method	**Text**		**R-1**	**R-2**	**R-L**	**Meteor**
	input	target				
生成模型						
非監督式	萃取式段落	-	0.1850	0.0347	0.0957	0.2144
循環式模型						
非監督式	萃取式段落	段落本身	0.2324	0.0266	0.1125	0.1561
監督式	萃取式段落	參考摘要	0.2551	0.0324	0.1231	0.1660
監督式	全部段落	對齊過後的摘要片段	0.2512	0.0315	0.1145	0.2046

表 4.4.9 資料集 BookSum(chapter) 循環模型效能

BookSum(book) 效能於下表4.4.10，由於資料筆數過少，生成出來的摘要已經過擬合，因此可以看到 METEOR 和 ROUGE 分數都很差。

Method	**Text**		**R-1**	**R-2**	**R-L**	**Meteor**
	input	target				
生成模型						
非監督式	萃取式段落	-	0.2248	0.0373	0.1172	0.1021
循環式模型						
非監督式	萃取式段落	段落本身	0.0000	0.0000	0.0000	0.0000
監督式	萃取式段落	參考摘要	0.1350	0.0189	0.0819	0.0558
監督式	全部段落	對齊過後的摘要片段	0.0054	0.0000	0.0053	0.0017

表 4.4.10 資料集 BookSum(book) 循環模型效能

DOI:10.6814/NCCU202200337

四、 其他模型比較

同時，我們考量了以下幾個不同的模型效能來和我們的實驗方法做比較。

- HipoRank

  HipoRank 是基於階層式的圖表示 (representation) 所建立的無監督排序模型，考量句子與句子和句子與章節間的交互關係，可以用於對長文檔進行提取摘要。本研究使用 HipoRank 的 SentBERT [20] 模型實作，因爲 SentBERT 在論文中的消融研究中表現最好。

- BertExt

  [14] 使用 BERT 模型，藉由修改後的輸入方式-額外多了標注不同文件的起始，探討文件向量對摘要任務的重要性，可應用於抽象式和萃取式摘要任務模型。

從表4.4.11可以看出，在 NovelChapters 資料集中 HipoRank 本身的模型效能就很好，和 $\text{oracle}_{\text{sentence}}$ 效能相當接近，而我們非監督式的模型會比 HipoRank 再低，一方面是因爲通常萃取式摘要的效能會比抽象式摘要好，另一方面是因爲我們使用萃取式提取輸入時，是以段落爲單位。而 BertExt 模型鑑於資料筆數太少，因此效能不好。

Method	R-1	R-2	R-L	METEOR
非監督抽象式摘要模型				
非監督式	0.2475	0.0343	0.1238	0.1591
非監督萃取式摘要模型				
HipoRank	0.2621	0.0333	0.1186	0.1856
監督萃取式摘要模型				
BertExt	0.1747	0.0248	0.1175	0.077
監督抽象式摘要模型				
監督式 (全部段落)	0.2514	0.0339	0.1146	0.2055

表 4.4.11 其他模型效能比較在 NovelChapters 資料集上

從表4.4.12可以看出，在 BookSum(chapter) 資料集上，基本上監督式訓練的萃取式段落設置效能大於 BertExt 方法，並且已經和 HipoRank 的效能差不多，只是我們的

DOI:10.6814/NCCU202200337

模型依舊有抽象式摘要模型的通病，產生的摘要有機率會生成出錯的字詞，因此在 METEOR 上分數依然落差很多。

Method	R-1	R-2	R-L	METEOR
非監督抽象式摘要模型				
非監督式	0.1850	0.0347	0.0957	0.2144
非監督萃取式摘要模型				
HipoRank	0.2518	0.0346	0.1110	0.2081
監督萃取式摘要模型				
BertExt	0.1825	0.0256	0.1218	0.0797
監督抽象式摘要模型				
監督式 (萃取式段落)	0.2551	0.0324	0.1231	0.1660

表 4.4.12 其他模型效能比較在 BookSum(chapter) 資料集上

在 BookSum(book) 資料集的模型效能列於表4.4.13，由於 HipoRank 方法無法處理過長的輸入，因此我們在這裡忽略不計，而我們的模型在監督式訓練設置下效果也不太好，也不予以考慮。從表4.4.13可以看出，非監督式模型的效能會比 BertExt 還好。

Method	R-1	R-2	R-L	METEOR
非監督抽象式摘要模型				
非監督式	0.2248	0.0373	0.1172	0.1021
監督萃取式摘要模型				
BertExt	0.0366	0.0044	0.0310	0.0118

表 4.4.13 其他模型效能比較在 BookSum(book) 資料集上

DOI:10.6814/NCCU202200337

五、 結果分析

以下我們以小說“亞當·貝德”第一個章節爲例，列於表4.4.14、4.4.15探討各模型在生成摘要時的特性。

Lead 由於是從原文直接選取，所以可以內容繁瑣，而 HipoRank 模型雖然將重要的幾句挑出，但仍舊描述細節過多。本篇提出的生成模型相比起零樣本的 T5 模型，可看到句子變得更加精簡，不過在此例中也可以看出參考摘要和原文文字仍有滿大的落差，所以效能無法有很大幅度的提升。

DOI:10.6814/NCCU202200337

模型	生成摘要
Lead	Alexey Fyodorovitch Karamazov was the third son of Fyodor Pavlovitch Karamazov, a land owner well known in our district in his own day, and still remembered among us owing to his gloomy and tragic death, which happened thirteen years ago, and which I shall describe in its proper place. For the present I will only say that this “landowner”—for so we used to call him, although he hardly spent a day of his life on his own estate—was a strange type, yet one pretty frequently to be met with, a type abject and vicious and at the same time senseless. But he was one of those senseless persons who are very well capable of looking after their worldly affairs, and, apparently, after nothing else.
HipoRank	Alexey Fyodorovitch Karamazov was the third son of Fyodor Pavlovitch Karamazov, a land owner well known in our district in his own day, and still remembered among us owing to his gloomy and tragic death, which happened thirteen years ago, and which I shall describe in its proper place. Fyodor Pavlovitch’s first wife, Adelaïda Ivanovna, belonged to a fairly rich and distinguished noble family, also landowners in our district, the Miüsovs. How it came to pass that an heiress, who was also a beauty, and moreover one of those vigorous, intelligent girls, so common in this generation, but sometimes also to be found in the last, could have married such a worthless, puny weakling, as we all called him, I won’t attempt to explain. Immediately after the elopement Adelaïda Ivanovna discerned in a flash that she had no feeling for her husband but contempt. [...]
$T5_{zero\ shot}$	Alexey Fyodorovitch was the third son of a landowner known in his own day. he was one of those senseless people who are very well capable of looking after their worldly affairs - and . he abandoned the child of his marriage with Adelada Ivanovna. a faithful servant of the family, Grigory, took the three-year-old Mitya into his care, despite his. ”It serves her right. God has punished her for her ingratitude” the general’s widow decided that it was a good thing. he did not refuse his formal consent to any proposition in regard to his children” education. the slaps she had given him, drove all over the. The Brothers Karamazov.Book 1, pred: the young man had an unmistakable influence over his father. he positively appeared to be behaving more decently and seemed at times ready to obey his son, though often extremely and even spitefully perverse. the younger brother, Alexey, had been a year already among us. he was the first of the three to arrive, and was cloistered there for the rest of his life.

表 4.4.14 小說亞當·貝德第一章在不同模型設置下生成的摘要以及參考摘要

DOI:10.6814/NCCU202200337

模型	生成摘要
生成模型	Fyodor Pavlovitch Karamazov was a landowner who was known for his gloomy and tragic death in the district, and he was one of the most shrewd, fantastical fellows in He was a father of the three-year-old Mitya Ivanovna, who had been married to Adelada and his wife, Grigory, and he was the first father to be able ”It serves her right. God has punished her for her ingratitude.” Fyodor Pavlovitch was a general’s widow, and he did not refuse his formal consent to any proposition in regard to his children. The general had given him slaps, but the general was not able The young man had an unmistakable influence over his father, who positively appeared to be behaving more decently and even seemed at times ready to obey his son, though often extremely and sometimes even spitefully perverse. Alexey, a young brother of Alexa, was born in the United States, and was buried in syria. He was the first member of the family to arrive at the monastery. It is the second
循環式模型(全部段落)	Alexey Fyodor Pavlovitch Karamazov, a landowner well known in our district in his own day, and still remembered among us owing to his gloomy and tragic death, which happened thirteen. The elopement Adelada Ivanovna discerned in a flash that she had no feeling for her husband but contempt. The marriage accordingly showed itself in its true colors with extraordinary rapid rapidity. “One would think that you’d got a promotion, Fyodor Pavlovitch, you seem so pleased in spite of your sorrow,” scoffers said to him. Many even added that he was glad of” [...]
Reference	Fyodor Karamazov and his three sons have just been reunited after many years, and the novel’s first chapters concern themselves mostly with the family’s backstory. We meet Fyodor, a ”muddle-headed” eccentric who has led a reckless and selfish life. Though many thought him too impulsive to be crafty, he died with 100,000 rubles, proving that he must have been keen in some ways. He has few friends and many enemies, and he is an enigma to all. He married a fiery, romantic woman named Adelaida, who thought his lifestyle was ”bold” After bearing him a son whom they named Dmitri, she ran off with a tutor. Fyodor was crushed by her desertion, but he also relished the idea of his humiliation so much that those who heard him talking about his situation thought that he somehow enjoyed his position as a cuckold. [...]

表 4.4.15 小說亞當·貝德第一章在不同模型設置下生成的摘要以及參考摘要

DOI:10.6814/NCCU202200337

第五章　結論

本篇提出了新的摘要任務架構，利用和以前不同的非監督式學習模型以及新的假資料的建構方式，藉由多任務學習的方式，將摘要任務和知識融合任務一起訓練，有效的解決了來源領域資料不足的問題，並且在原模型上有效地提升了該效能和摘要的品質。除此之外，我們也考慮了在監督式訓練下，加入循環式的架構，讓靠後的輸入能得到前文的資訊，並且設計了兩種不同方式的輸入和標籤，探討不同設置下的模型效能。

實驗結果顯示本研究在非監督式的設置下，與原本的零樣本模型至少平均大約有十多個百分點以上的增幅，而模型生成的摘要也更加精簡。而在監督式的設置下，在資料筆數足夠的情況下，都會比非監督式模型的效能來的更高。然而雖然監督式模型能幫助模型更好的生成摘要，但仍然需要大量的已標記資料，才能有不錯的成績，因此如何去平衡資料量與模型設計是之後可以改進的部分之一。

未來研究會針對輸入的單與資料和參考摘要如何對齊做研究探討，在抽取式輸入的設置下，輸入即決定生成的摘要是否合理，於實驗中可以看到本文輸入端是用提取後的自然段落，而 [4] 模型則以句子爲單位下去作挑選，從實驗結果可以看出段落中仍會有冗餘的資訊未被過濾，因此資料集的切分粒度以及如何設計段落與句子間的關係也是一個很有趣的議題。在實驗中也可以看出將參考摘要對齊後作爲標籤並未有更好的表現，這個也會是本研究後續需要再繼續深入研究的方向。

DOI:10.6814/NCCU202200337

參考文獻

[1] Stefanos Angelidis and Mirella Lapata. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3675–3686, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[2] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[3] Jan A. Botha, Manaal Faruqui, John Alex, Jason Baldridge, and Dipanjan Das. Learning to split and rephrase from wikipedia edit history, 2018.

[4] Yue Dong, Andrei Mircea, and Jackie C. K. Cheung. Discourse-aware unsupervised summarization of long scientific documents, 2021.

[5] Quentin Grail, Julien Perez, and Eric Gaussier. Globalizing BERT-based transformer architectures for long document summarization. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1792–1810, Online, April 2021. Association for Computational Linguistics.

[6] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies, 2020.

[7] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[8] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.

[9] Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization, 2021.

DOI:10.6814/NCCU202200337

[10] Faisal Ladhak, Bryan Li, Yaser Al-Onaizan, and Kathleen McKeown. Exploring content selection in summarization of novel chapters, 2021.

[11] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[12] Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain, July 2004.

[13] Peter J. Liu*, Mohammad Saleh*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*, 2018.

[14] Yang Liu and Mirella Lapata. Text summarization with pretrained encoders, 2019.

[15] Erwin Marsi and Emiel Krahmer. Explorations in sentence fusion. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*, 2005.

[16] Ani Nenkova, Sameer Maskey, and Yang Liu. Automatic summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, page 3, Portland, Oregon, June 2011. Association for Computational Linguistics.

[17] Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Comput. Linguist.*, 28(4):399—408, dec 2002.

[18] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

DOI:10.6814/NCCU202200337

[19] Noraini Mohd Razali, John Geraghty, et al. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1–6. International Association of Engineers Hong Kong, 2011.

[20] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[21] Amin Riazi. Genetic algorithm and a double-chromosome implementation to the traveling salesman problem. *SN Applied Sciences*, 1(11):1–7, 2019.

[22] Tobias Rohde, Xiaoxia Wu, and Yinhan Liu. Hierarchical learning for generation with long source sequences, 2021.

[23] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

[24] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks, 2017.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[26] Peter West, Ari Holtzman, Jan Buys, and Yejin Choi. BottleSum: Unsupervised and self-supervised sentence summarization using the information bottleneck principle. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3752–3761, Hong Kong, China, November 2019. Association for Computational Linguistics.

[27] Wei Xu and Ralph Grishman. A parse-and-trim approach with information significance for Chinese sentence compression. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation (UCNLG+Sum 2009)*, pages 48–55, Suntec, Singapore, August 2009. Association for Computational Linguistics.

DOI:10.6814/NCCU202200337

[28] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 2020.

[29] Jingqing Zhang, Yao Zhao, Mohammad Ahmad Saleh, and Peter J. Liu. Pegasus: Pretraining with extracted gap-sentences for abstractive summarization by sequence-to-sequence models, 2020.

[30] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online, July 2020. Association for Computational Linguistics.

[31] Chenguang Zhu, Ziyi Yang, Robert Gmyr, Michael Zeng, and Xuedong Huang. *Leveraging Lead Bias for Zero-Shot Abstractive News Summarization*, page 1462—1471. Association for Computing Machinery, New York, NY, USA, 2021.

[32] Markus Zopf. Auto-hMDS: Automatic construction of a large heterogeneous multilingual multi-document summarization corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

DOI:10.6814/NCCU202200337