國立中興大學資訊科學與工程學系

資訊專題競賽報告

基於知識蒸餾的BERT QG模型壓縮

專題題目說明、價值與貢獻自評（限100字內）：

本專題證明，我們可以依靠知識蒸餾的方式將模型壓縮得更小，用最少的資源讓模型達到最佳效能。 我們發現，將訓練過程結合dropout與蒸餾變成兩段式訓練，能讓速度快1.7倍，參數量減少42%，且僅讓分數減少0.014分。

專題隊員：

姓名	E-mail	負責項目說明	專題內貢獻度(%)
方郁婷	4107056005@smail.nchu.edu.tw	研究論文 設計實驗 訓練模型 比對實驗結果 研究訓練方法	100

【說明】上述表格之專題內貢獻度累計需等於100%。

指導教授簡述及簡評：

指導教授簽名：

中　　華　　民　　國　　年　　月　　日

目錄

一、摘要

關鍵詞:BERT、深度學習、模型壓縮、知識蒸餾、QG model

此研究計畫的研究目的為**將訓練好的問句生成（Question Generation）模型使用知識蒸餾技術來進行模型的壓縮**，使模型跑得更快，記憶體占用更小，且減少所需的運算資源。

與以往的論文研究蒸餾哪層和使用哪個損失函數不同，我的研究比較偏研究訓練步驟與超參數調整，令人高興的事，比起傳統的知識蒸餾，我發現結合 dropout 和蒸餾的兩段式訓練，能讓速度快 1.7 倍，參數量減少 42%，且僅讓分數減少 0.014 分，使模型的效能達到最佳化。

二、專題研究動機與目的

◆ 為什麼選擇 QG model?

QG(Question generation) model[1][2]，是用來做問句生成的模型，可以根據文本內容自動出題，在偏鄉或師資不充裕這些可能沒這麼多資源去照顧學生的學習情況的地方，我們可以用問句生成自動出題，讓學生藉由回答問題來檢驗自己的學習情況，以此養成自主學習的能力。也因為現在的疫情，越來越多人減少外出，甚至停課，結合科技來學習也變得越來越重要，因此我認為 QG model 將是未來教育科技(EduTech)不可或缺的應用。

◆ 目前的 model 有甚麼缺點?

我們知道模型越大，效果越好，所以現在的模型規模也有越來越大的趨勢(如圖一)，但這對於使用上有許多缺點:

占用的記憶體多	可能無法部屬在邊緣端，只能部屬在雲端計算後藉由網路傳送，對於網路不好的地方(偏鄉)是個挑戰，也會增加網路的負擔。
延遲	層數過大會造成很大的延遲，讓使用觀感降低，訓練時也要花更多時間。
耗電、耗錢	運算量大造成更加耗電、耗錢。

這些缺點顯示出壓縮模型是非常必要的，研究[3][4]也表示，現今的模型的確是過於冗餘，藉由模型壓縮，我們能夠將模型壓縮得更小且將只降低些微的精準度，對於現今模型越來越大的趨勢來說，我認為模型壓縮是未來不可或缺的研究方向之一。

圖一:模型的規模近年來有越來越大的趨勢，甚至大到連下載都無法下載

圖片取自: DistilBERT [10]

以上的缺點在所有 model 中都有，其中對於 QG model 來說，延遲與計算尤為嚴重，這裡就來講一下 QG model 的運作流程[5]

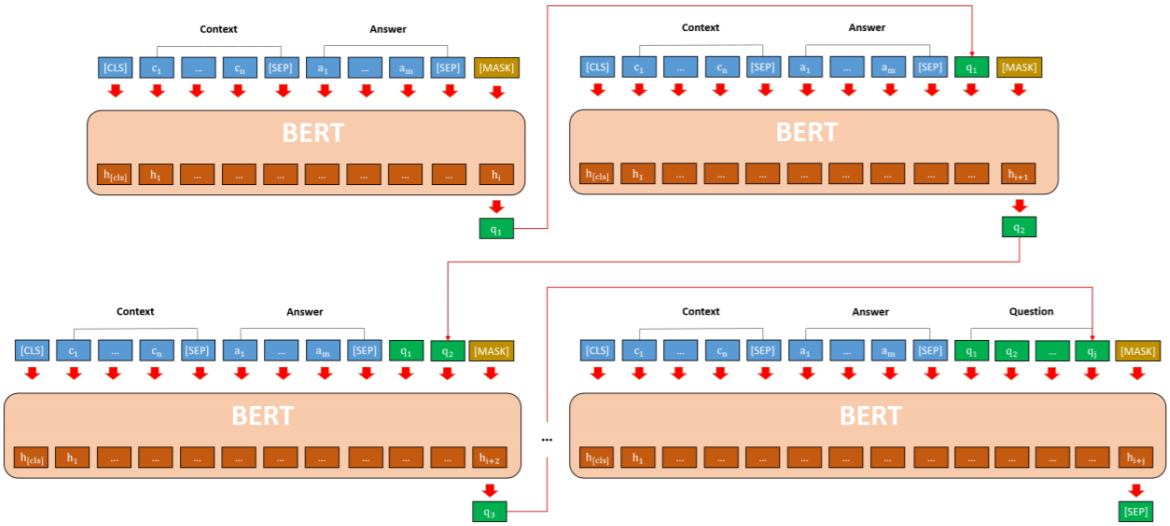


Figure 3: The BERT-SQG architecture

輸入範例

Context: 2010 年引進的廣州快速公交運輸系統，屬世界第二大快速公交系統，日常載客量可達 100 萬人次，高峰時期每小時單向客流高達 26900 人次，僅次於波哥大的快速交通系統，平均每 10 秒鐘就有一輛巴士，每輛巴士單向行駛 350 小時。包括橋樑在內的站台是世界最長的州快速公交運輸系統站台，長達 260 米。目前廣州市區的計程車和公共汽車主要使用液化石油氣作燃料，部分公共汽車更使用油電、氣電混合動力技術。2012 年底開始投放液化天然氣燃料的公共汽車，2014 年 6 月開始投放液化天然氣插電式混合動力公共汽車，以取代液化石油氣公共汽車。2007 年 1 月 16 日，廣州市政府全面禁止在市區內駕駛摩托車。違

反禁令的機動車將會予以沒收。廣州市交通局聲稱禁令的施行，使得交通擁擠問題和車禍大幅減少。廣州白雲國際機場位於白雲區與花都區交界，2004 年 8 月 5 日正式投入運營，屬中國交通情況第二繁忙的機場。該機場取代了原先位於市中心的無法滿足日益增長航空需求的舊機場。目前機場有三條飛機跑道，成為國內第三個擁有三跑道的民航機場。比鄰近的香港國際機場第三跑道預計的 2023 年落成早 8 年。

Answer: 10 秒鐘

疊代的過程:

iter 0

input_QWords [MASK]

label_QWord 廣

---

iter 1

input_QWords 廣[MASK]

label_QWord 州

---

iter 2

input_QWords 廣州[MASK]

label_QWord 的

---

iter 3

input_QWords 廣州的[MASK]

label_QWord 快

… iter 22

input_QWords 廣州的快速公交運輸系統每多久就會有一輛巴士？[MASK]

label_QWord [SEP]

就以上的範例來說，生成一個問題就讓 model 跑了 23 次(22 個字外加[SEP]這個字)，在延遲以及計算量大概會呈二十幾倍成長(視生成問題的字數而定)。因此，我想透過模型壓縮裡的知識蒸餾來解決這個問題。

三、專題重要貢獻

資工三 4107056005 方郁婷:

BERT 知識蒸餾論文閱讀研究

BERT 知識蒸餾訓練步驟研發
BERT 知識蒸餾實驗設計
BERT 知識蒸餾架構嘗試
BERT 知識蒸餾模型訓練與監控

四、團隊合作方式

雖然我是一人組團，但我每天都會激勵自己，使自己不要怠惰。

我只要做出一些成果就會與教授面談，在報告進度之餘向老師討論與指教研究時所遇到的困難與瓶頸，藉以適時的調整研究方向，使得研究可以順利進行下去。

五、設計原理、研究方法與步驟

◆ 如何壓縮模型?

此研究我選擇使用**知識蒸餾**的技術，壓縮的原理主要是**將大模型蒸餾成一個小模型**，藉由**減少模型層數**來減少所需運算時間與記憶體。

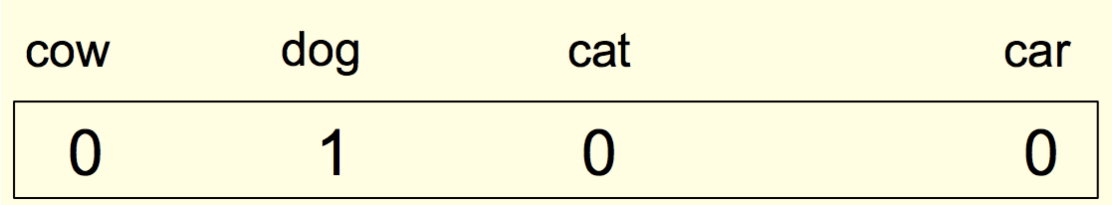
這可能會有一個疑問，為何不要一開始就訓練一個小模型就好了呢?

◆ 為什麼不直接訓練一個小模型?

對神經網絡來說，有一個共識：模型越大，層數越深，學習能力越強[6]，當面對一個任務時，我們並不知道多大的模型是合適的，如果**一開始模型過小，容易欠擬合，不能完成對數據的特徵提取**，直接使用小模型訓練，會碰到欠擬合、訓練不收斂、精度低，結果震盪等各種問題，這時通常會選用一個大模型進行訓練，然後再根據訓練情況進行調整。

◆ 知識蒸餾如何解決上述問題?

原始的標籤只有對與不對(0 or 1)，無法得知標籤之間的相關性，只能讓模型自己慢慢學習，但因為小模型的學習能力較弱，所以**對小模型來說，原始的標籤所含的資訊量太小了，無法對數據進行有效的數據的特徵提取**(如圖二)。



圖二:原始 label 只有對與不對(0 or 1)的資訊，對於小模型來說，無法有效學習。

如果我們能**藉由老師模型預測的答案擴大標籤所蘊含的資訊量**，就能讓小模型更好的學到東西(如圖三)，這樣的數字其實提供了更多的**隱含資訊（Dark knowledge）**，例如我們可以知道 dog 和 cat 比較相近與 dog 和 car 相差最遠等原始標籤無法給予的信息，讓模型在訓練中學習更多東西。

cow	dog	cat	car
10^{-6}	.9	.1	10^{-9}

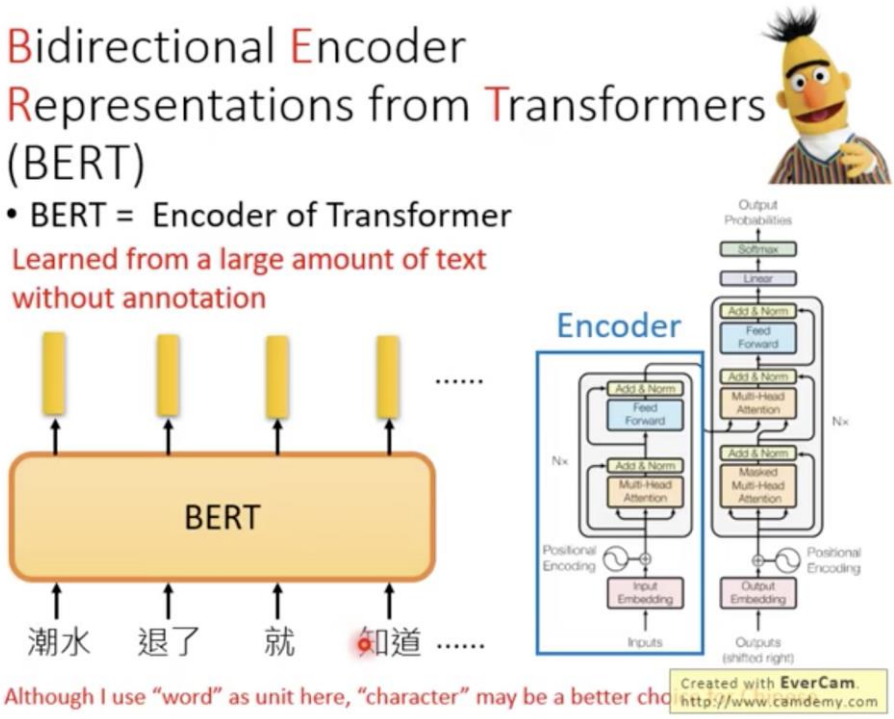
圖三:藉由老師模型可得知原始標籤無法給予的資訊，使學生模型學到更多訊息

◆ BERT 是什麼?

BERT 全名為 Bidirectional Encoder Representations from Transformers，是 Google 以無監督的方式利用大量無標註文本「煉成」的**語言預訓練模型**

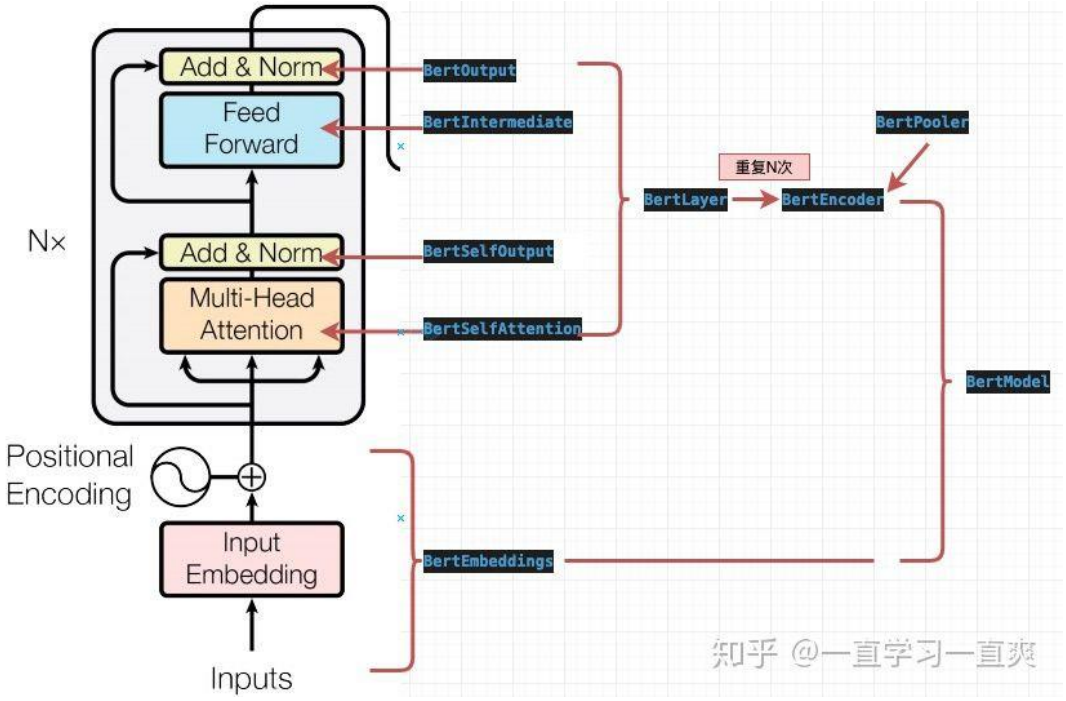
◆ BERT 有哪些層可以蒸餾?

BERT 的架構是 Transformer[7] 的 Encoder



圖四: [李宏毅教授 BERT 課程]

```
BertModel(
  (embeddings): BertEmbeddings(config)
  (encoder): BertEncoder(config) 12層
  (pooler): BertPooler(config)
)
```



圖五: BERT 架構

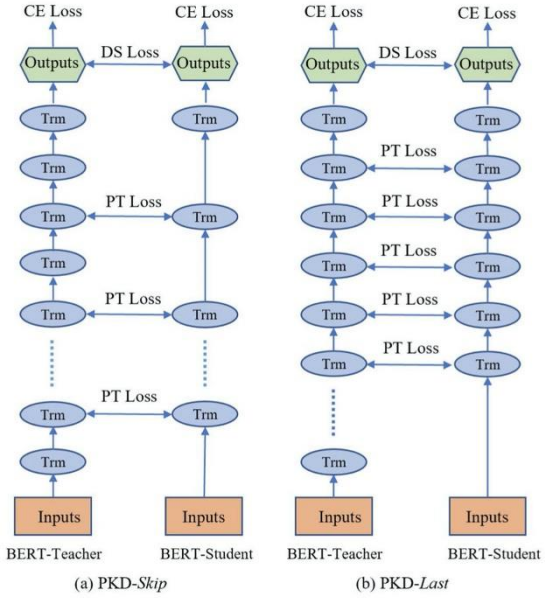
就以上的架構圖來說，大概可分為 embedding、attention layer(注意力層)、hidden layer(隱藏層)和 logit 輸出

◆ 知識蒸餾經典模型回顧

以下模型根據先後時間順序，每一個都有突破，因論文內容太多，所以如果只有細微的差別或是後面的模型沿用之前模型的方法就沒摘錄進去，我只摘錄我認為差別較大的改進。

Knowledge Distillation[8] (hinton, 2015)	知識蒸餾的始祖，提出 teacher/student 師徒概念的框架，蒸餾 **logit 輸出**
BERT-PKD[9] (2019,微軟)	為了避免在蒸餾最後一層時過擬合，BERT-PKD 還蒸餾了**中間隱藏層**的知識，讓學生模型更好地擬合，文中提出兩種蒸餾方法(圖六)，結果顯示 **PKD-skip** 要略好一點點 (<0.01)
DistilBERT [10] (2019,HuggingFace)	提出在預訓練階段進行蒸餾，並且藉由 **cosine embedding loss** 調整教師和學生的向量方向
TinyBERT[11] (2019,華為)	分別在預訓練和精調階段蒸餾教師模型，除了蒸餾中間的隱藏層和 embedding，還對**注意力矩陣**進行蒸餾，**注意力矩陣**採用教師-學生 **MSE** 作為損失函數，embedding

	同樣是採用 MSE 作為損失函數
MiniLM [12] (2020，微軟)	除了藉由 **KL 散度**蒸餾注意力矩陣，還蒸餾了 **Value 矩陣**，並且在實驗中**只蒸餾最後一層**。另外，MiniLM 還引入了**助教機制**，當學生模型的層數、維度都小很多時，先用一個維度小但層數和教師模型一致的助教模型蒸餾，之後再把助教的知識傳遞給學生。



圖六: PKD-skip 和 PKD-last 的差別

Table 1. Comparison with previous task-agnostic Transformer based LM distillation approaches.

Approach	Teacher Model	Distilled Knowledge	Layer-to-Layer Distillation	Requirements on the number of layers of students	Requirements on the hidden size of students
DistillBERT	$\text{BERT}_{\text{BASE}}$	Soft target probabilities Embedding outputs			✓
TinyBERT	$\text{BERT}_{\text{BASE}}$	Embedding outputs Hidden states Self-Attention distributions	✓		
MOBILEBERT	$\text{IB-BERT}_{\text{LARGE}}$	Soft target probabilities Hidden states Self-Attention distributions	✓	✓	✓
MINILM	$\text{BERT}_{\text{BASE}}$	Self-Attention distributions Self-Attention value relation			

圖七:各個模型的蒸餾比較，圖片取自 MiniLM[12]

研究方法與步驟:

需先準備老師模型與學生模型

老師模型(12 層):

使用全部的訓練資料訓練出一個 12 層的 BERT 模型

學生模型(6 層):

根據 Poor Man's BERT[13]，直接砍掉後面六層最能保持住原本模型的效果

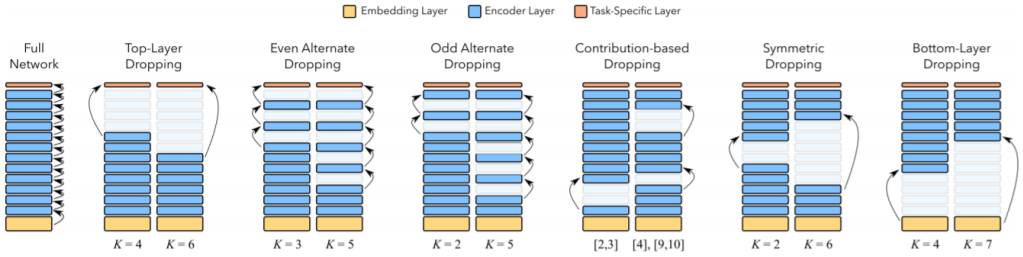


Figure 1: Layer-dropping Strategies

圖八:各種 drop layer 的方法

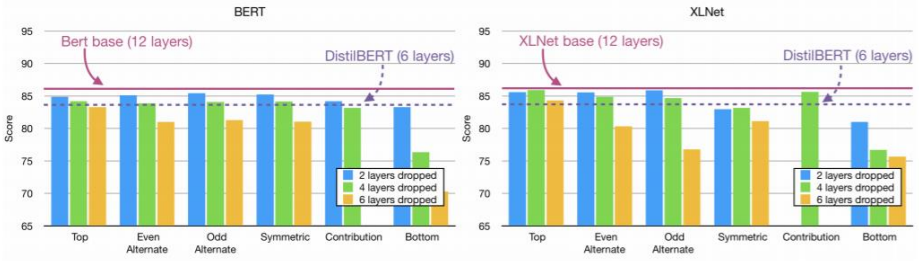
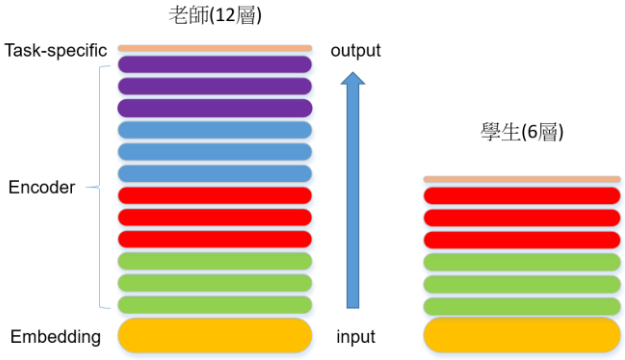


Figure 2: Average classification performance on GLUE tasks when using different layer-dropping strategies and when removing different numbers of layers for BERT and XLNet. Note that the contribution-based strategy selects layers based on the similarity threshold. In some cases it does not select (2,4 or 6) number of layers, which results in some missing bars in the figure.

圖九: 各種 drop layer 的方法的分數比較

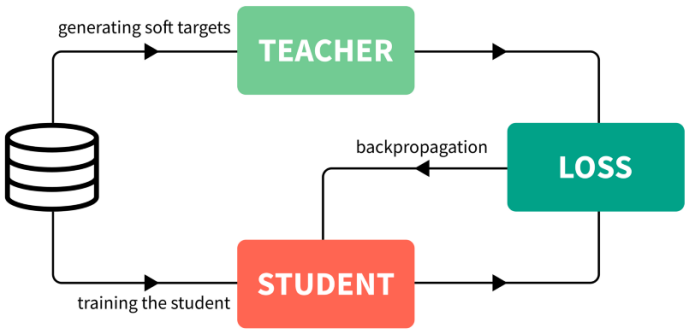
因此我都是直接拿老師模型 drop 後面的 layer 後作為學生的預訓練模型



圖十:產生學生預訓練模型的方法

知識蒸餾流程:

老師與學生各自輸出後，藉由損失函數擬合兩者間的答案，希望學生的答案與老師的答案能越接近越好。



圖十一:知識蒸餾流程

蒸餾方法:

根據 BERT-PKD 可以知道使用 PKD-skip 要略好一點點（<0.01），所以我之後是選擇用圖六的 PKD-skip 蒸餾(隔層)，但蒸餾的順序我提出了三個方法

圖十二:最上面那層是 logit，其餘 6 層分別對應 BERT 的 6 個 Encoder layer 每個 layer 包含隱藏層和注意力層

1 全層蒸餾

name	Bleu_1	Bleu_2	Bleu_3	Bleu_4	METEOR	ROUGE_L	avg_score	evalTime
SixLayer_distill2/epoch-4/	0.35658	0.26721	0.20791	0.16579	0.17803	0.32676	0.25038	6456.259298

2 疊加蒸餾

name	Bleu_1	Bleu_2	Bleu_3	Bleu_4	METEOR	ROUGE_L	avg_score	evalTime
SixLayer_distill/epoch-4/	0.35402	0.26611	0.20723	0.16528	0.17756	0.32787	0.249678	6402.046689

3 逐層蒸餾

name	Bleu_1	Bleu_2	Bleu_3	Bleu_4	METEOR	ROUGE_L	avg_score	evalTime
SixLayer_distill3/epoch-4/	0.35119	0.26299	0.20425	0.16255	0.17598	0.32452	0.246913	17334.96256

圖十三:從結果來看好像都差不多，但全層蒸餾的分數最高，所以我最後選擇使用全層蒸餾

Hint: avg_score = (Bleu_1 + Bleu_2 + Bleu_3 + Bleu_4 + METEOR + ROUGE_L)/6

損失函數:

Logit 使用 cross entropy

隱藏層和注意力層使用 MSE

資料集:

取自 DRCD 資料集，此資料集從 2,108 篇維基條目中整理出 10,014 篇段落，並從段落中標註出 30,000 多個問題。

六、系統實驗與實現

訂立 baseline:

在進行實驗之前，我先模仿前人的作法，先確認一下前人的方法是否有用，順訂立 baseline

直接 train:

	name	Bleu_1	Bleu_2	Bleu_3	Bleu_4	METEOR	ROUGE_L	avg_score	evalTime
'Six_train/epoch-4/	0.43459	0.34205	0.28154	0.23692	0.21427	0.40803	0.319567	6193.024373	

Hinton(蒸餾始祖):

	name	Bleu_1	Bleu_2	Bleu_3	Bleu_4	METEOR	ROUGE_L	avg_score	evalTime
0	../distill/HalfModel_a1/epoch-4	0.43514	0.34301	0.28231	0.23698	0.21644	0.41543	0.321552	18616.913703

全層蒸餾:

	name	Bleu_1	Bleu_2	Bleu_3	Bleu_4	METEOR	ROUGE_L	avg_score	evalTime
0	QG_model_six_all_distil/epoch-4/	0.45707	0.36442	0.30265	0.2566	0.22748	0.4274	0.33927	3438.479683

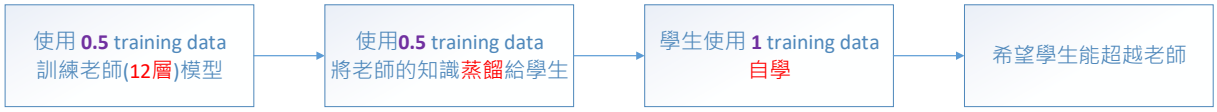
經由實驗後發現全層蒸餾分數最高，所以我之後的目標是 avg_score 要超越 0.33927

Hint: avg_score = (Bleu_1 + Bleu_2 + Bleu_3 + Bleu_4 + METEOR + ROUGE_L)/6

實驗步驟:

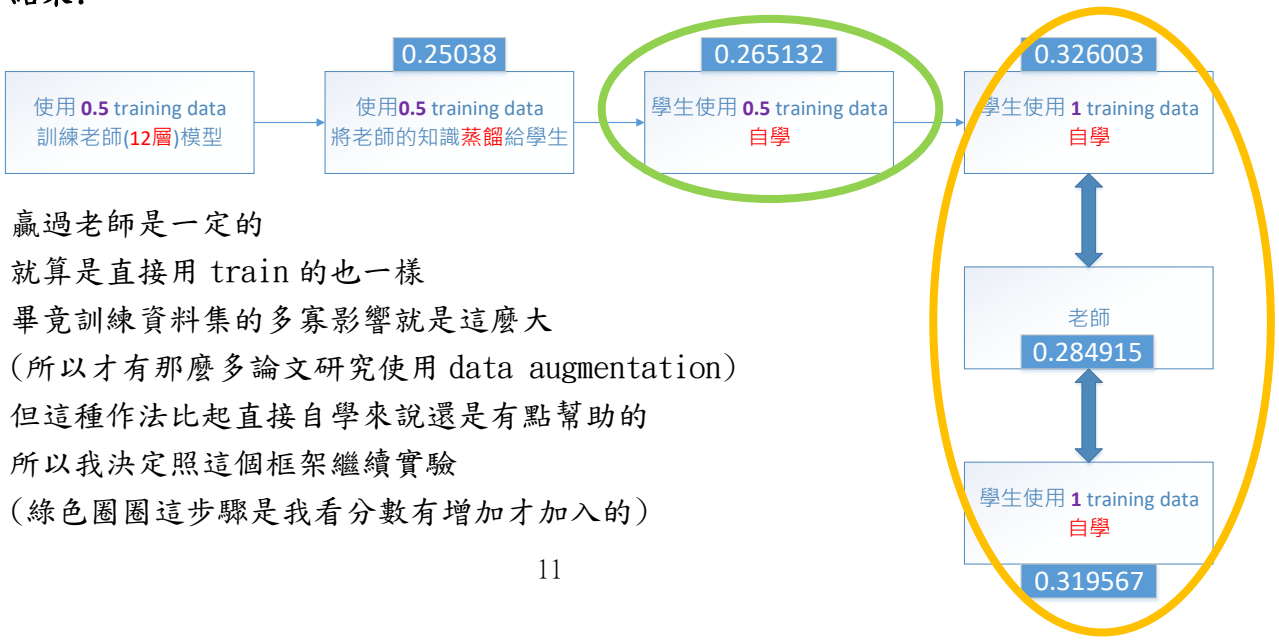
根據經典論文回顧的圖七可知，能蒸餾的層大部分都被蒸過了，因此一開始我其實有點無從下手，不知還能如何改進，向耀中老師請教後，老師給了一個青出於藍勝於藍的題目。

青出於藍勝於藍實驗目的:



Hint: 0.5 training data 代表使用全部訓練資料的 50%，並且是由前往後取 50%
因此，如果占比同為 0.5 代表是取一樣的訓練資料。

結果:



贏過老師是一定的

就算是直接用 train 的也一樣

畢竟訓練資料集的多寡影響就是這麼大

(所以才有那麼多論文研究使用 data augmentation)

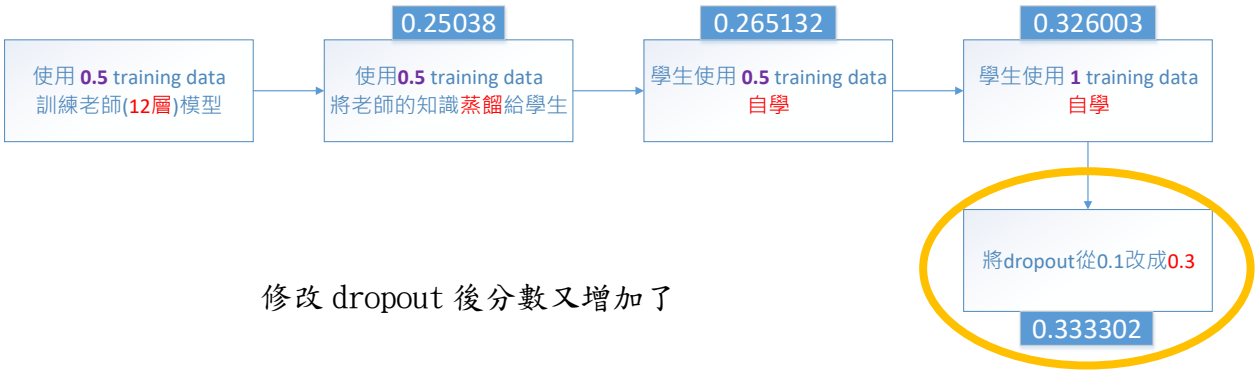
但這種作法比起直接自學來說還是有點幫助的

所以我決定照這個框架繼續實驗

(綠色圈圈這步驟是我看分數有增加才加入的)

之後因為我覺得 train 的步驟比較多，為了防止過擬合，所以我又修改了 dropout，將 dropout 從原本的 0.1 改成 0.3

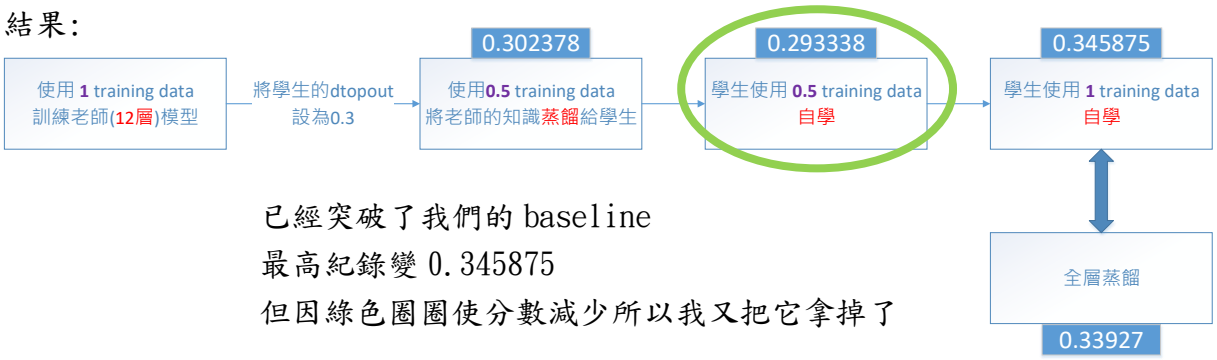
結果:



修改 dropout 後分數又增加了

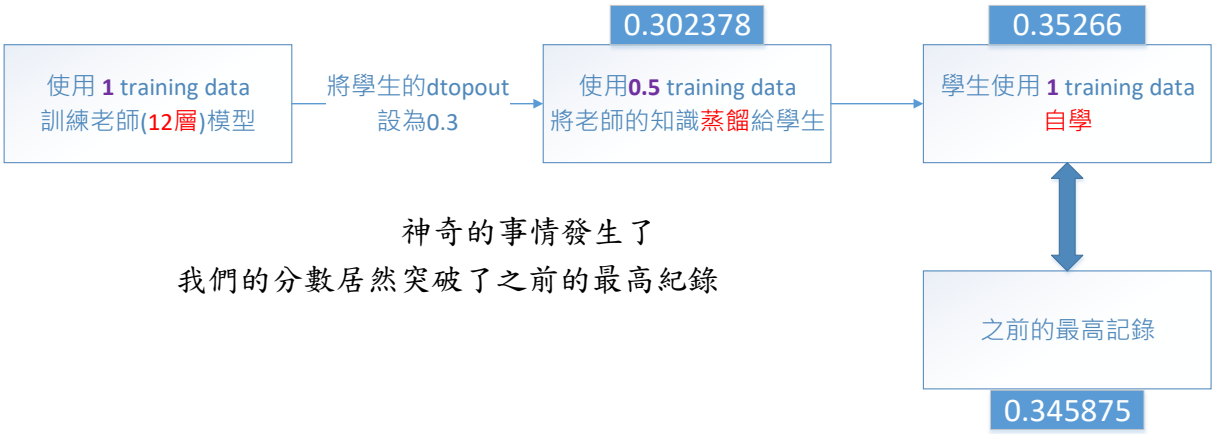
因為使用 0.5 training data 的老師比較爛，所以我想，用 1 training data 訓練出的老師效果應該會更好，並且將 dropout 在一開始就設為 0.3

結果:



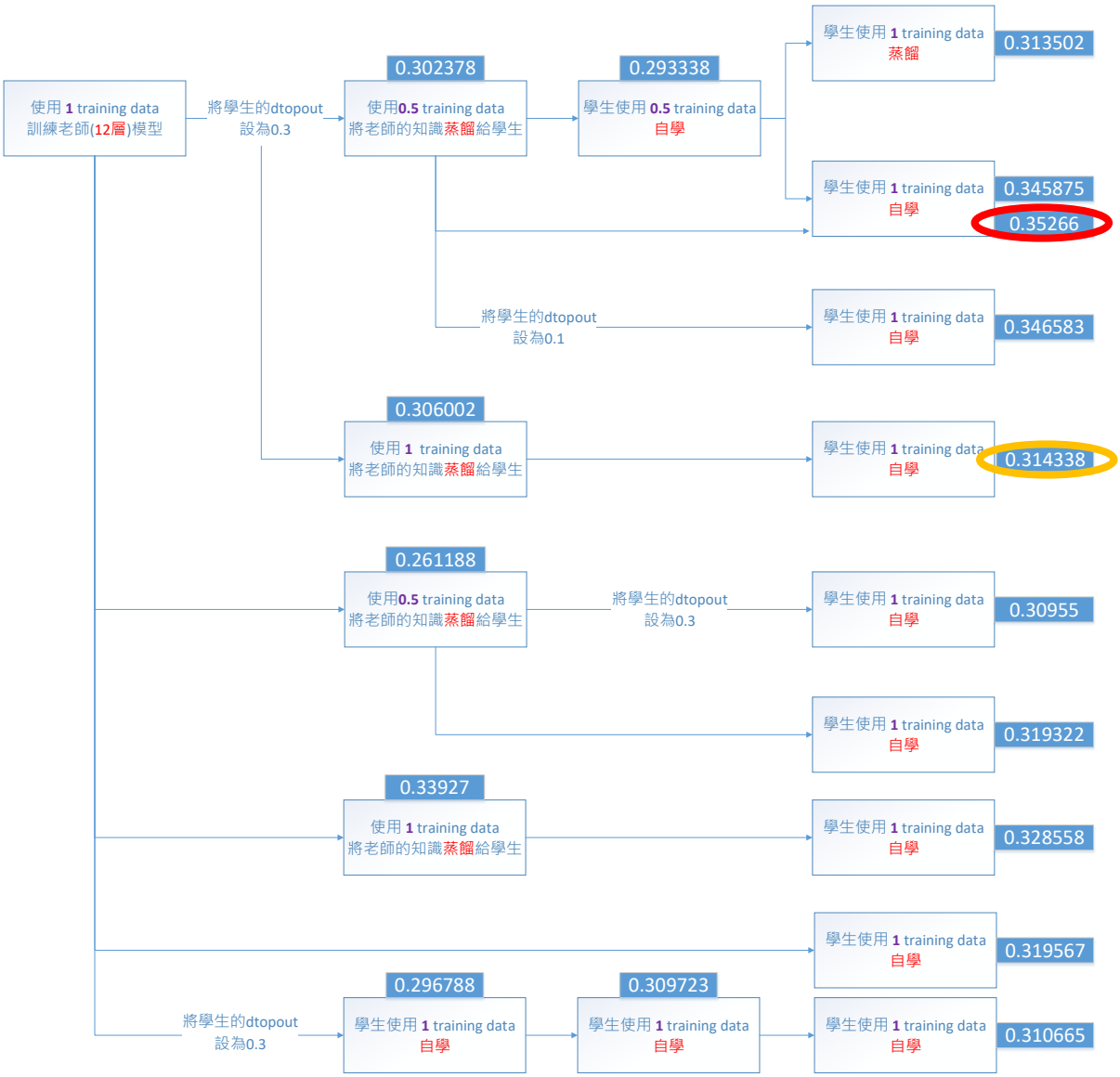
已經突破了我們的 baseline
最高紀錄變 0.345875
但因綠色圈圈使分數減少所以我又把它拿掉了

結果:



神奇的事情發生了
我們的分數居然突破了之前的最高紀錄

之後我又進行了其他的分支實驗，期待能找出最佳的實驗步驟
但最後還是發現原來的做法最好(藍底白色數字框代表 avg_score)



圖十四

七、效能評估與成果

DistilBERT 是 BERT 經過知識蒸餾後的縮小版預訓練模型，就如同 BERT 一樣，可以用它做特定任務的 fine-tune(DistilBERT 的老師是 BERT，而我的老師是先用 BERT train 好的 QG model)。我把 DistilBERT 作為知識蒸餾模型的比較指標，會選擇用它是因為只有它支援中文，但它是 multilingual 版本的，並無專門為中文蒸餾，這也增加了自己蒸餾的必要性(如果選擇的語言是中文的話)。

Hint:運算時間是指運算一份 testing data 所花的時間

模型	運算時間	參數量	分數
老師模型(target)	5993.456949	102290312	0.366758
一般訓練	6193.024373	59763080	0.319567
DistilBERT	16927.706819	177974523	0.319705

全層蒸餾	3438.479683	59763080	0.33927
我的方法	3518.408209	59763080	0.35266

換成百分比:

%數表示占原本老師模型的幾%

例: 一般訓練的時間%數 = 6193.024373/5993.456949 = 103%

模型	運算時間	參數量	分數
一般訓練	103%	58%	87.13%
Distilbert	282%	174%	87.17%
全層蒸餾	57%	58%	92.5%
我的方法	59%	58%	96.2%

經過實驗後，我發現最佳的訓練方法就是以下步驟:

至於為什麼蒸餾的 training data 只需要 0.5，以一般的情況來說，training data 不是要越多越好嗎?

就如同圖三所說的一樣，在學習貓的圖片時，老師也會告訴你貓與狗比較像。舉例來說，就像讀歷史一樣，好的老師會在教你中國史時順便連結歐洲和美洲歷史，所以雖然你只讀了中國史，但其實你已經把世界歷史讀完了，因此，使用 0.5 training data 足夠學習全部知識(可由如圖十四的分枝圖看出，使用 0.5 或 1 training data 蒸餾後的分數差不多)，但這些知識學生還沒有到能熟練運用，所以還是需要自學消化知識。

如果使用 1 training data，如圖十四的分枝圖所示，學生自學後的分數反而比較低(橘色圈圈)，推測是因為太過過擬合了。

發現這件事對我來說是個警惕，因為在一般的常識裡，training data 越多越能防止過擬合，但用在知識蒸餾裡反而會造成反效果，經過這次專題報告後，期待今後的研究我能夠以多方面考量，不要侷限在固有的思維，將自己限制住。

八、結論

Dropout 與蒸餾是絕配，雖然這兩種方法個別使用的話，效果可能都很有限，但搭配在一起竟然有意想不到的結果。因為蒸餾的特性就是知識量豐富、學習速度快、容易過擬合; Dropout 在一般情況下，雖然能防止過擬合，但學習速度慢，容易遇到瓶頸，與蒸餾結合後反而能將彼此的缺點相消，達成夢幻組合。另外要注意蒸餾時的 training data 不能太大，否則反而會造成反效果。

根據我的訓練方法，比起傳統的知識蒸餾，還多加了一個自學的步驟，就如同「**師父領進門，修行在個人**」一樣，老師只管指點門路，真正學到本領還是要

靠自己的努力，沒想到套用在深度學習裡竟也有同樣的效果，但這個步驟必須跟 0.5 training data 與 dropout 結合，否則無法發揮效果(由圖十四的分支圖可看出，直接自學分數反而會降低)。

就如同台灣之光台積電一直致力於減少奈米大小一樣，模型壓縮也一直致力於減小模型的大小，經由此次實驗我發現了模型壓縮的潛力與可能性，未來也希望此領域能繼續蓬勃發展，讓未來能夠壓縮出更省錢、省時、省空間的高效模型。

九、參考文獻

[1] Xinya Du, Junru Shao, Claire Cardie. 2017. Learning to Ask: Neural Question Generation for Reading Comprehension. arXiv:1705.00106

[2] Nan Duan, Duyu Tang, Peng Chen, Ming Zhou. 2017. Question Generation for Question Answering. ACL: D17-1090

[3] Paul Michel, Omer Levy and Graham Neubig. 2019. Are Sixteen Heads Really Better than One? arXiv:1905.10650

[4] Angela Fan, Edouard Grave and Armand Joulin. 2020. Reducing Transformer Depth on Demand with Structured Dropout. arXiv:1909.11556

[5] Ying-Hong Chan, Yao-Chung Fan. A Recurrent BERT-based Model for Question Generation. 2019. ACL: D19-5821

[6] Eldan R, Shamir O. 2016. The power of depth for feedforward neural networks. arXiv:1512.03965

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. 2017. Attention Is All You Need. arXiv:1706.03762

[8] Geoffrey Hinton, Oriol Vinyals and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531

[9]Siqi Sun, Yu Cheng, Zhe Gan and Jingjing Liu. 2019. Patient Knowledge Distillation for BERT Model Compression. arXiv:1908.09355

[10] Victor Sanh, Lysandre Debut, Julien Chaumond and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv:1910.01108

[11] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang and Qun Liu. 2019. TinyBERT: Distilling BERT for Natural Language Understanding. arXiv:1909.10351

[12] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. arXiv:2002.10957

[13] Hassan Sajjad,Fahim Dalvi,Nadir Durrani,Preslav Nakov. 2020. Poor Man's BERT: Smaller and Faster Transformer Models. arXiv:2004.03844

台大李弘毅: Network Compression (2/6)
(https://www.youtube.com/watch?v=7B8Cx7woQk4&t=7s&ab_channel=Hung-yiLee)

知識蒸餾 KnowledgeDistillation - CH.Tseng
(https://chtseng.wordpress.com/2020/05/12/%E7%9F%A5%E8%AD%98%E8%92%B8%E9%A4%BE-knowledgedistillation/)
比 Bert 體積更小速度更快的 TinyBERT
(https://zhuanlan.zhihu.com/p/94359189)
谷歌：任務無關輕量級預訓練模型 MobileBERT: 比 BERT 小 4 倍、快 5 倍
(https://kknews.cc/tech/jark3vl.html)
DistilBERT — 更小更快的 BERT 模型
(https://medium.com/nlp-tsupei/distilbert-%E6%9B%B4%E5%B0%8F%E6%9B%B4%E5%BF%AB%E7%9A%84bert%E6%A8%A1%E5%9E%8B-eec345d17230)
模型優化之知識蒸餾技術簡介
(https://zhuanlan.zhihu.com/p/298215569)
UDICatNCHU/PyTorch-Tutorial/Lesson_9_BERT_for_QG_on_DRCD_dataset.ipynb
(https://github.com/UDICatNCHU/PyTorch-Tutorial/blob/master/Lesson_9_BERT_for_QG_on_DRCD_dataset.ipynb)