U.S. Patent Phrase to Phrase Matching

組別:第一組
組員:吳逸邦、許紘碩、黃振嘉、楊智翔

一、題目介紹

這次我們參加的是Kaggle上的競賽，題目為"U.S. Patent Phrase to Phrase "[註一]，賽期為03/21/2022-06/20/2022。參賽者需要訓練出一個可以判斷與專利相關短語的文字模型，在無網路以及有限的GPU與TPU的使用量下盡力取得最佳的成績。在數據集中有到成對的短語（一個題目和一個目標短語），目的在於評估它們的相似程度，從 0（完全不相似）到 1（含義相同）。此競賽與標準語義的相似性任務之不同之處在於此處的相似性是在專利的上下文中進行評分的，特別是其 CPC 分類（Cooperative Patent Classificcation 版本 2021.05），它代表專利所涉及的主題。例如，雖然短語“bird”和“Cape Cod”在正常語言中語義相似性可能較低，但如果在“house”的上下文中考慮，它們的含義相似性會更接近。

數據橫列標題

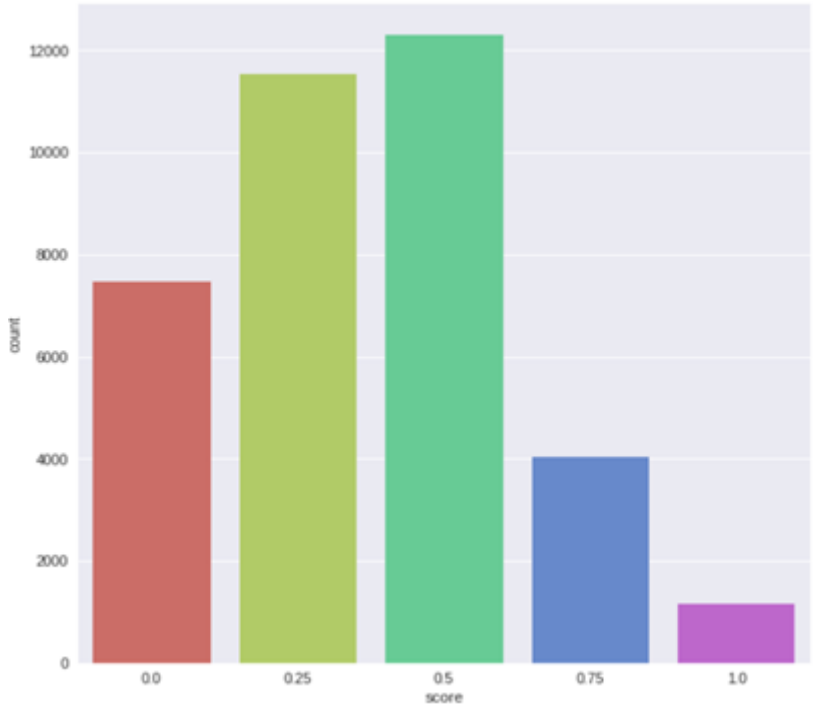
1.id - 編號(不重複)
2.anchor - 題目短語
3.target - 目標短語
4.context - the CPC 分類(版本 2021.05)
5.score - anchor與target的相似程度(0, 0.25, 0.5, 0.75, 1.0)

CPC分類（版本 2021.05）:

這邊我們提供大分類的標題供參考，更多詳細資訊請見官方首頁[註二]與表格[註三]。
A: Human Necessities
B: Operations and Transport
C: Chemistry and Metallurgy
D: Textiles
E: Fixed Constructions
F: Mechanical Engineering
G: Physics
H: Electricity
Y: Emerging Cross-Sectional Technologies

二、分析+前處理

分析:



(圖一)score分布數量長條圖

觀察(圖一): 1.0 和 0.75的較少(特別是score = 1.0 的)

(圖二)Anchor字數的長條圖

觀察(圖二): 兩個字的比例最多

前處理:

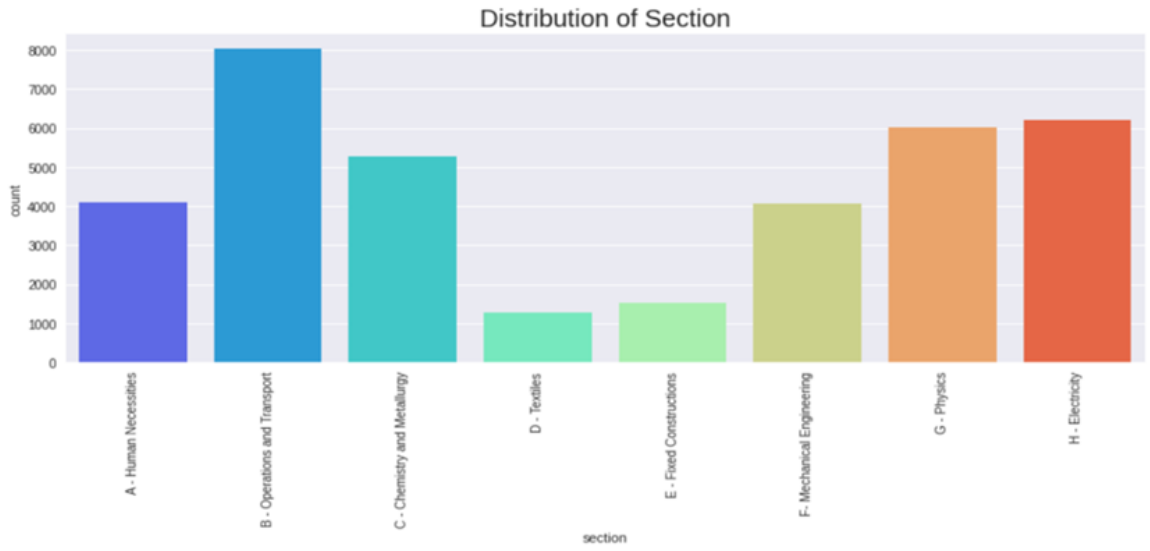
我們有找到一個title.csv檔，裡面紀錄CPC的分類，然後我們將Context不同的開頭藉由CPC分類產生新的clolumn

	id	anchor	target	context	context_text	text
0	4112d61851461f60	opc drum	inorganic photoconductor drum	G02	PHYSICS. OPTICS	opc drum[SEP]inorganic photoconductor drum[SEP...
1	09e418c93a776564	adjust gas flow	altering gas flow	F23	MECHANICAL ENGINEERING; LIGHTING; HEATING; WEA...	adjust gas flow[SEP]altering gas flow[SEP]MECH...

Tokenizer:　　我們最後丟到model的text切字如下圖

```
train['text'] = train['anchor'] + '[SEP]' + train['target'] + '[SEP]'  + train['context_text']
test['text'] = test['anchor'] + '[SEP]' + test['target'] + '[SEP]'  + test['context_text']
```

選擇了[‘anchor’] [‘target’]和[‘context_text’]



(圖三)Context各個Section的分類

觀察(圖三): D、E Section數量較少

三、模型:DeBERTa

模型介紹:

DeBERTa 1.0版本在BERT的基礎上有三個主要的改進點:

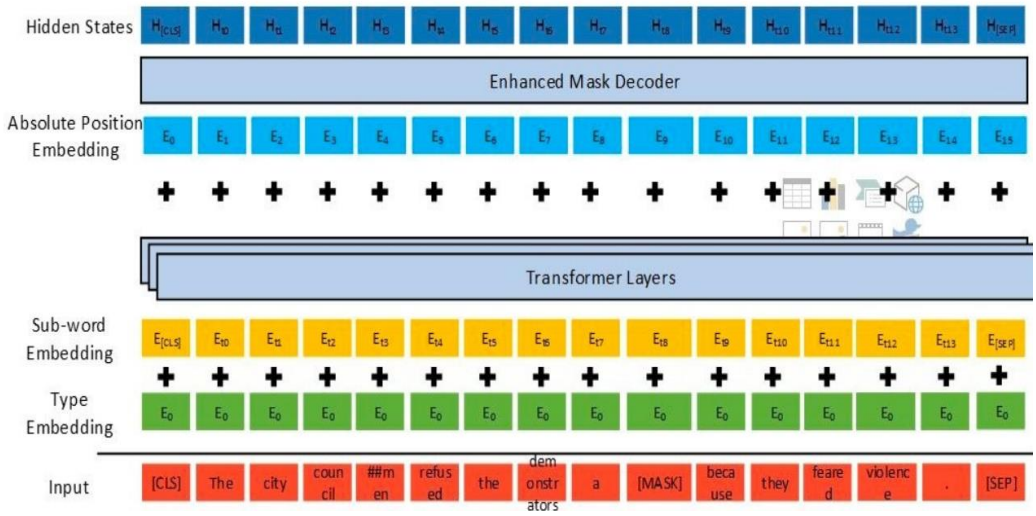
1.第一個是注意力解耦機制，動機是來自於因為一組詞的 Attention 不光取決於內容，還和它們的相對位置有關，比如 deep learning 兩個字連在一起的時候關係比分開的時候還要強，因此讓

每個字用兩個向量去表達它的內容和位置，那token之間的注意力權重可以由內容和位置之間的解耦矩陣組成，他的公式取了其中三項content-to-content, content-to-position, position-to-content，由於使用的是相對位置編碼，position-to-position這一項並不能提供更多額外的訊息，因此沒有加入這項。

$$\tilde{A}_{i,j} = \underbrace{Q_i^c K_j^{cT}}_{\text{(a) content-to-content}} + \underbrace{Q_i^c K_{\delta(i,j)}^{r\,T}}_{\text{(b) content-to-position}} + \underbrace{K_j^c Q_{\delta(j,i)}^{r\,T}}_{\text{(c) position-to-content}}$$

2.第二個是用EMD來強化預訓練的輸出層，而它主要是要解決預訓練和經條的不匹配問題

原始的BERT存在預訓練和微調不一致問題。預訓練階段，隱層最終的輸出輸入到softmax預測被mask掉的token，而微調階段則是將隱層最終輸出輸入到特定任務的decoder。這個decoder根據具體任務不同可能是一個或多個特定的decoder，如果輸出是概率，那麼還需要加上一個softmax層。為消除這種不一致性，DeBERTa將MLM與其他下遊任務同等對待，並將原始BERT中輸出層的softmax替換為「增強後的mask decoder(EMD)」，EMD包含一個或多個Transformer層和一個softmax輸出層。至此，結合了BERT和EMD的DeBERTa成為了一個encoder-decoder模型。



3.第三個改變是預訓練時引入對抗訓練

而DeBERTa 3.0 則是改變預訓練任務，讓模型在下遊任務上的表現可以更好

分別對DeBERTa 拿掉新改入的部分(下圖)

DeBERTa$_{base}$	**86.3/86.2**	**92.1/86.1**	**82.5/79.3**	**71.7**
-EMD	86.1/86.1	91.8/85.8	81.3/78.0	70.3
-C2P	85.9/85.7	91.6/85.8	81.3/78.3	69.3
-P2C	86.0/85.8	91.7/85.7	80.8/77.6	69.6

EMD：沒有 EMD 的 DeBERTa 模型。
C2P：沒有content-to-position這一項的 DeBERTa 模型。
P2C：沒有position-to-content這一項的 DeBERTa 模型。

這邊是要說明這些改變如果拿掉會對模型造成的影響，主要是要顯示這些改變的重要性，那從表格可以看出來，DeBERTa拿掉任何一個部分都會讓模型變差，比較之下去掉EMD下降的比較少，這個部分也可能是比較不重要的，因為在3.0的版本中已經不再使用MLM作為預訓練任務了。

6.最後拿DeBERTa去跟其他模型做比較
可以從下方表格清楚看到，DeBERTa明顯表現得比其他模型好很多。

Model	SQuAD 1.1	SQuAD 2.0	MNLI-m/mm	SST-2	QNLI	CoLA	RTE	MRPC	QQP	STS-B
	F1/EM	F1/EM	Acc	Acc	Acc	MCC	Acc	Acc/F1	Acc/F1	P/S
BERT-Large	90.9/84.1	81.8/79.0	86.6/-	93.2	92.3	60.6	70.4	88.0/-	91.3/-	90.0/-
RoBERTa-Large	94.6/88.9	89.4/86.5	90.2/-	96.4	93.9	68.0	86.6	90.9/-	92.2/-	92.4/-
XLNet-Large	95.1/89.7	90.6/87.9	90.8/-	97.0	94.9	69.0	85.9	90.8/-	92.3/-	92.5/-
DeBERTa-Large[1]	95.5/90.1	90.7/88.0	91.3/91.1	96.5	95.3	69.5	91.0	92.6/94.6	92.3/-	92.8/92.5
DeBERTa-XLarge[1]	-/-	-/-	91.5/91.2	97.0	-	-	93.1	92.1/94.3	-	92.9/92.7
DeBERTa-V2-XLarge[1]	95.8/90.8	91.4/88.9	91.7/91.6	**97.5**	95.8	71.1	**93.9**	92.0/94.2	92.3/89.8	92.9/92.9
DeBERTa-V2-XXLarge[1,2]	**96.1/91.4**	**92.2/89.7**	**91.7/91.9**	97.2	**96.0**	72.0	93.5	**93.1/94.9**	**92.7/90.3**	**93.2/93.1**
DeBERTa-V3-Large	-/-	91.5/89.0	**91.8/91.9**	96.9	**96.0**	**75.3**	92.7	92.2/-	**93.0/-**	93.0/-

四、TRAINING

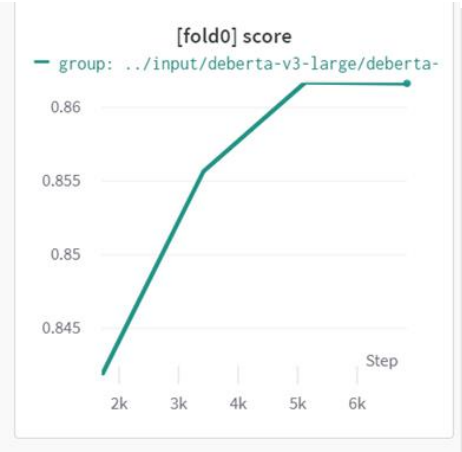
Training Hyperparameters and Loss function:

1.Learning rate : 2e-5
2.Batch size : 16
3.Loss function : BCEWithLogitsLoss
3.Optimizer: AdamW
4.Scheduler: get_cosine_schedule_with_warmup (transformer library提供)

Training策略:

使用Cross Validation(4 fold CV)
利用sklearn中的stratifiedKfold()函式，使每個class的資料能均勻散佈在每個fold中，以此題目來看，class就是5種不同的score。我們把資料切成4folds，每個fold訓練4 epochs

使用weight and bias來記錄訓練過程loss和pearson coefficient score的變化，下圖為其中一個fold的pearson score和training loss



Training Result:

Fold 0: Avg Training loss: 0.49776 Avg Valid loss: 0.54506 Pearson: 0.85854
Fold 1: Avg Training loss: 0.49937 Avg Valid loss: 0.54271 Pearson: 0.86164
Fold 2: Avg Training loss: 0.50191 Avg Valid loss: 0.54314 Pearson: 0.8576
Fold 3: Avg Training loss: 0.49921 Avg Valid loss: 0.54405 Pearson: 0.85771

CV Score: 0.8588

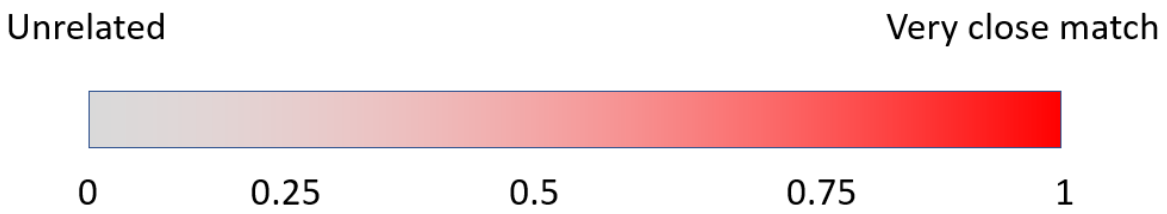
Inference method:

把每個fold訓練好的model讀進來，把各個model預測的結果曲平均數作為final prediction

LeaderBoard Score:: 83.38

五、DISCUSSION

1. 一開始以為題目很像五分類的問題，將output neuron設成5做五分類(0,0.25,0.5,0.75,1五類)，但是效果不好，後來發現因為這五種score彼此之間有關連性，所以比較像是做regression，output一個數值，可能是0.35也可能是0.8之類的，直接拿這數值做pearson



2. 一開始使用BERT模型，pearson score大約是0.803，後來改成Deberta-v3-large，但是Deberta-v3-large的參數比較多，train一個epoch要花很多的時間

3. 這分project，我們在時坐上很多東西都是try and error，比較greedy的試著去增進結果，像是修改attention layer裡面的參數，或是修改hyper parameters等，但也因為一個epoch需要很大量的時間，所以也沒辦法做太多次的嘗試

[註一]U.S. Patent Phrase to Phrase Matching | Kaggle
[註二]Home | Cooperative Patent Classification
[註三]Table | Cooperative Patent Classification