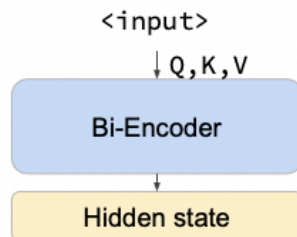


## Q1: Model (2%)

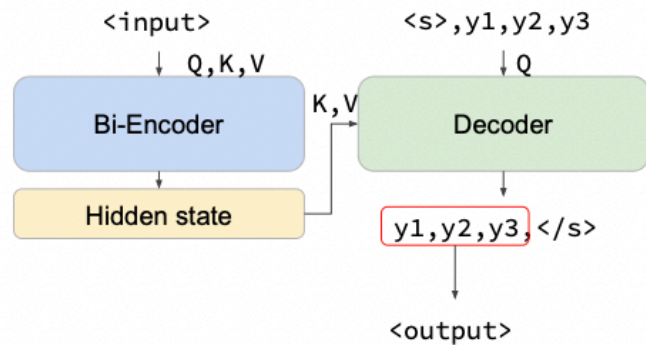
### Model (1%)

Describe the model architecture and how it works on text summarization.

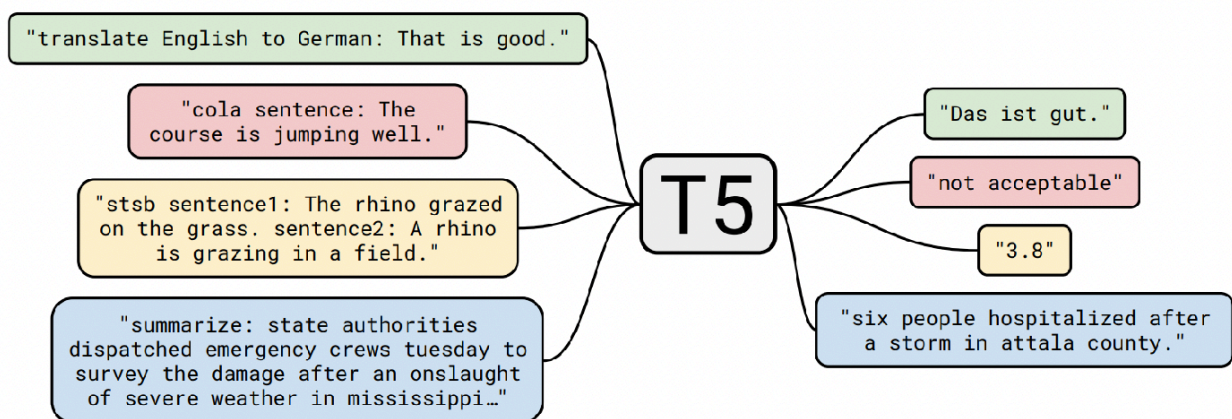
HW2: BERT



HW3: T5



T5 不同於 BERT 是 encoder-decoder 架構 (如上方右圖),  $\langle \text{input} \rangle$  是文章,  $\langle \text{output} \rangle$  則是輸出的 summary,  $y_i$  是一個字典的機率分佈, 代表  $y_i$  這個字的機率分佈。特色是 pre-train 的時候把很多本來不是 seq-to-seq 的 task 也用 seq-to-seq 來做, 如下圖範例有數值評分預測、分類問題等等。



#ADL 2022  
#Homework 3  
R10946029，陳婉如

## Preprocessing (1%)

Describe your preprocessing (e.g. tokenization, data cleaning and etc.)

```
tokenizer = AutoTokenizer.from_pretrained("google/mt5-small")

def preprocess(articles):
    encode_articles = {}
    maintext = [article["maintext"] for article in articles]

    # Tokenize
    encode_articles = tokenizer(maintext,
                               max_length=max_length,
                               truncation=True,
                               padding=True)

    if 'title' in article[0].keys():
        titles = [article["title"] for article in articles]
        encode_articles['title'] = tokenizer(titles,
                                             max_length=56,
                                             truncation=False,
                                             padding=True)

    return encode_articles
```

T5 tokenizer 是以 google 開源的 sentencepiece 專案為基礎，SentencePiece 切 subword 的方法有 byte-pair-encoding (BPE) [Sennrich et al.] 以及 unigram language model [Kudo.]。

我僅使用文章前 384 字、title 的部分則取 56 字（原因是 training data 中最長 title 是 56 字）。

## Q2: Training (2%)

### Hyperparameter (1%)

Describe your hyperparameter you use and how you decide it.

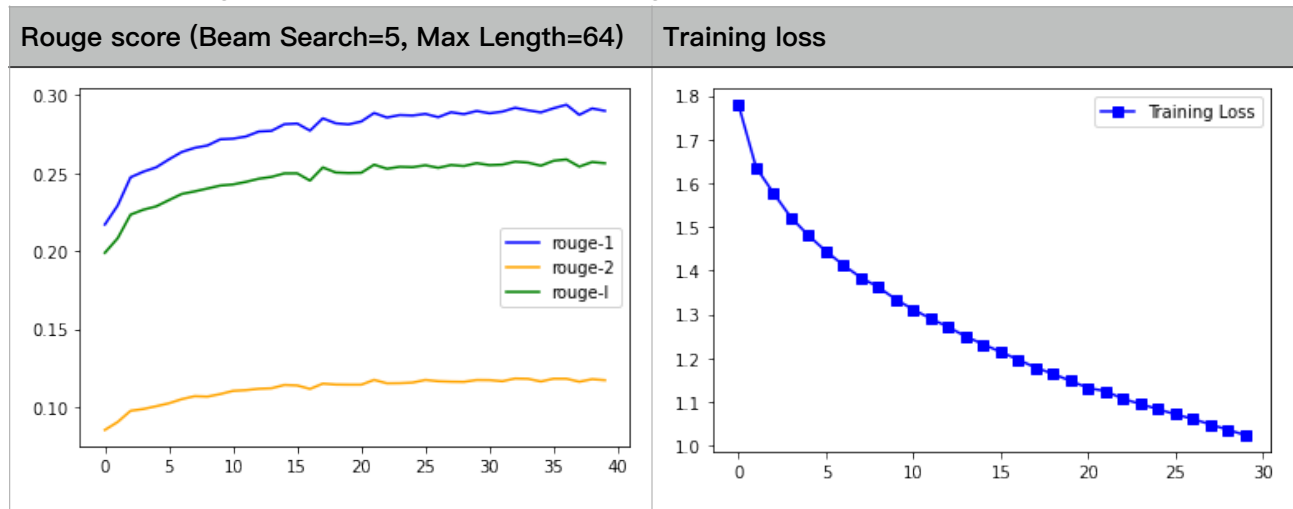
Batch size: 32

Optimizer: AdamW

Learning rate:  $1e-4$

### Learning Curves (1%)


Plot the learning curves (ROUGE versus training steps)



### Q3: Generation Strategies(6%)

#### Strategies (2%)

Describe the detail of the following generation strategies:

	Description
Greedy	 <p>每次選最高機率的字。</p>
Beam Search	每次選機率最大的字，不一定會是最佳路徑。但因為計算所有可能的 word 組合，time complexity 會很高，因此會決定要觀察 k 條 path，取最終機率最大的那條。也就是說當 k=1 時，等同於 Greedy。
Top-k Sampling	從大到小排序前 k 的字 sample
Top-p Sampling	從大到小排序機率加起來小於 p 的字 sample
Temperature	$P(w_t) = \frac{e^{s_w/\tau}}{\sum_{w_i \in V} e^{s_{w_i}/\tau}}$ <p>字在算 softmax 的時候指數多除上一個參數 temperature hyperparameter <math>\tau</math>，所以當 <math>\tau</math> 大，機率分佈會比較請向於 uniform 分佈，反之則會集中於某幾個字。</p>

#ADL 2022  
#Homework 3  
R10946029, 陳婉如

### Hyperparameters (4%)

Try at least 2 settings of each strategies and compare the result.

What is your final generation strategy? (you can combine any of them)

#### Sample v.s. Greedy

	Sample	Greedy
Rouge-1	0.2290	0.2802
Rouge-2	0.0734	0.1039
Rouge-l	0.1986	0.2459

從上面結果可以看到 Greedy 遠勝於單純的機率 sample strategy, 原因應是 sample 整個字典很容易 sample 到不好的字。

#### Greedy v.s. Beam Search

	Greedy	Beam Search (n=3)	Beam Search (n=5)
Rouge-1	0.2802	0.2899	0.2899
Rouge-2	0.1039	0.1157	0.1173
Rouge-l	0.2459	0.2555	0.2564

Beam search 愈大愈好, 符合我們的預期, 因為當K=字典總字數即為最佳解。

#### Greedy v.s. Top-K

	Greedy	Top-K (K=10)	Top-K (K=50)
Rouge-1	0.2802	0.2500	0.2289
Rouge-2	0.1039	0.0837	0.0737
Rouge-l	0.2459	0.2167	0.1987

Top K 越大分數越差, 原因推測是 sample 的時候可能選到不好的字。

#### Greedy v.s. Top-P

	Greedy	Top-P (P=0.5)	Top-P (P=0.8)
Rouge-1	0.2802	0.2699	0.2540
Rouge-2	0.1039	0.0979	0.0884
Rouge-l	0.2459	0.2358	0.2211

Top-P 的結果跟 Top-K 結果類似, 範圍越廣分數越低, 也可以從上面的效果推估機率分佈應該相當的 spike。

#ADL 2022  
#Homework 3  
R10946029，陳姵如

#### Greedy v.s. Top-P (P=0.8)+Temperature

	Greedy	Top-P 0.8+Temperature 0.7	Top-P 0.8+Temperature 1.3
Rouge-1	0.2802	0.2683	0.2285
Rouge-2	0.1039	0.0969	0.0718
Rouge-l	0.2459	0.2341	0.1976

分數跟想像中的預測相符，當 temperature 小會把分佈變得更 spike，那用 top-p 能 sample 的字就越少，越接近 greedy。反之，temperature 大，分佈變常態分佈，能 sample 的字變多也類似 p 變大或 top-k k 變大的效果，越有機會選到不好的字。

**\*Final generation strategy: beam search (num\_beams=5)**