學號：b07901112 系級：電機二 姓名：劉聿珉

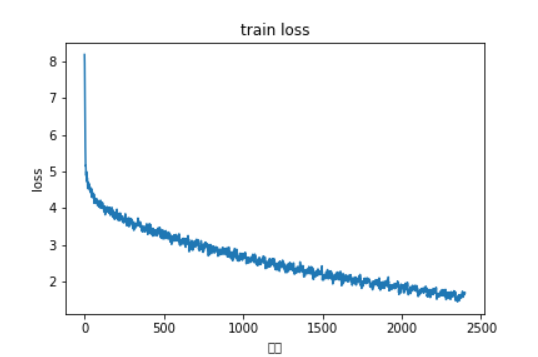
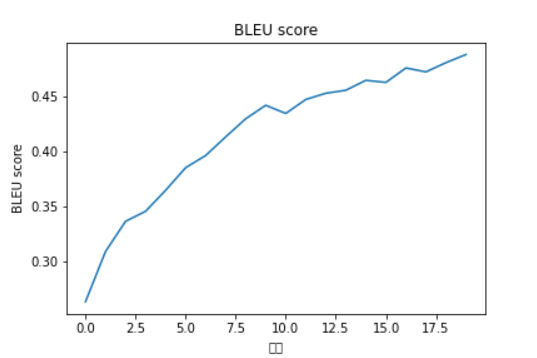
1. (20%) Teacher Forcing:
2. 請嘗試移除 Teacher Forcing，並分析結果。

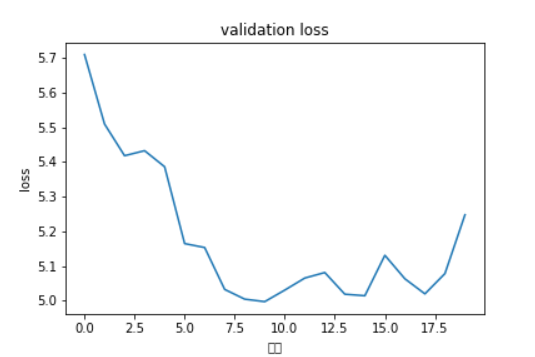
兩組數據皆沒有做attention、beam search。以這樣的條件下進行比較

有Teacher Forcing的圖還有result：

val [12000] loss: 5.268, Perplexity: 206.518, bleu score: 1.011

test loss: 5.5152, bleu\_score: 0.4521

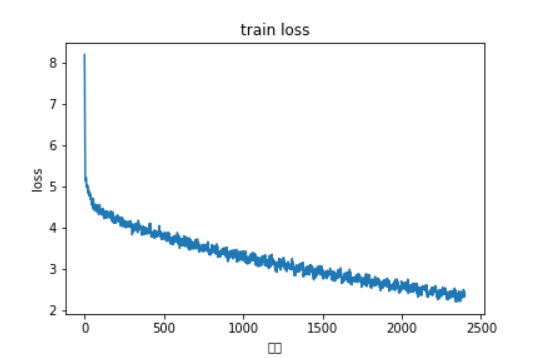
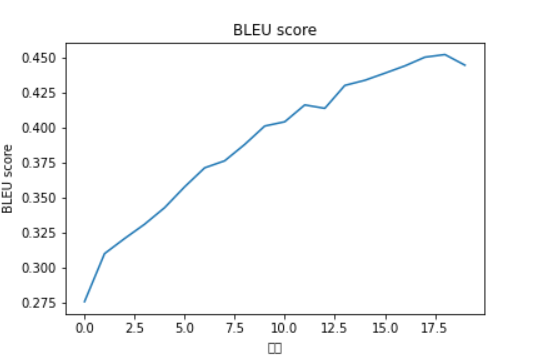


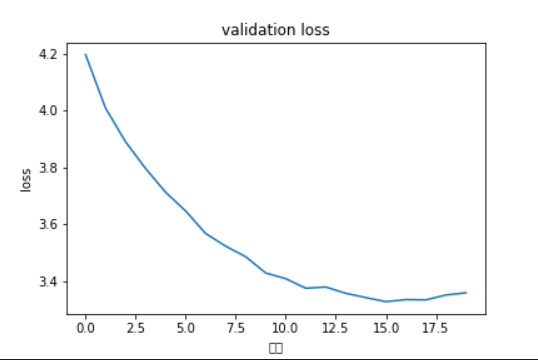


移除Teacher Forcing:

val [12000] loss: 3.394, Perplexity: 29.509, bleu score: 0.449

test loss: 3.6148, bleu\_score: 0.3752





我們從上面的模型可以看到有teacher forcing的Perplexity比沒有Teacher forcing的Perplexity還要高上許多，但是沒有teacher forcing的bleu score比起有teacher forcing的bleu score還要低得多。我想這些結果是符合理想的，因為沒有teacher forcing的model會在模型樹狀圖上面亂走，沒有人告訴他正確的output，所以bleu score自然就會非常的不好。而有teacher forcing的則是不斷的在沿著正確的output的那個分之在train 使得他對於其他分支並沒有良好的理解，所以只要一開始走到其他分支，就會使模型走到一個對他來說未知的領域，Perplexity自然就會高上許多。

而有teacher forcing的loss也比沒有teacher forcing的loss大上許多，由此可以知道，Teacher forcing對於training model不會是一個很好的策略。

1. (30%) Attention Mechanism:
   1. 請詳細說明實做 attention mechanism 的計算方式，並分析結果。

沒有做：Attention Mechanism

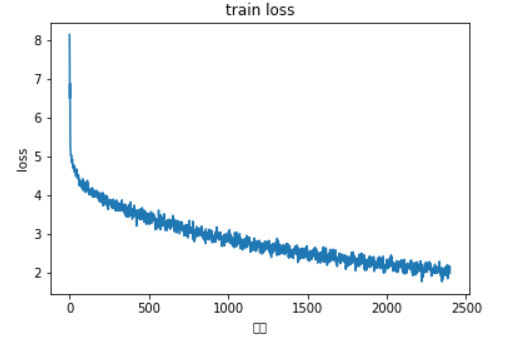
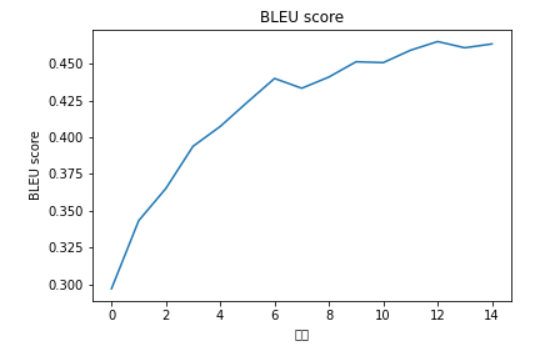
val [12000] loss: 3.421, Perplexity: 31.349, bleu score: 0.457

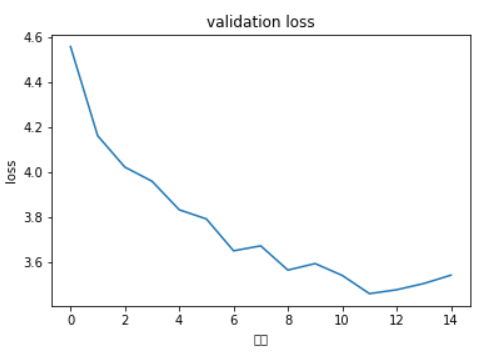
topk=3: test loss: 3.5807, bleu\_score: 0.4322

做Attention Mechanism:

val [12000] loss: 3.561, Perplexity: 35.024, bleu score: 0.459

test loss: 3.5991, bleu\_score: 0.4532





結果討論：

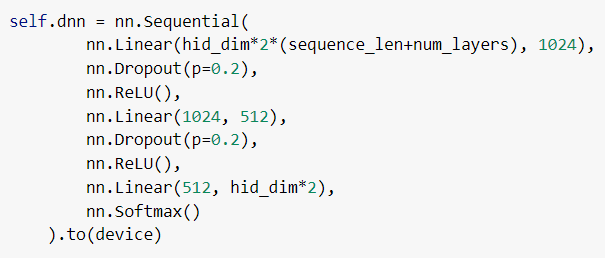
可以看到不管是loss或是bleu score有做attention的結果都比沒有attention的結果來的好。由此可以知道利用attention模型可以使模型注意力更集中在某些字詞上，不僅可以降低loss還可以使產生的語句更接近人所說的話(bleu score up!)

Attention的使用時機：

在decoder中input經過RNN之後馬上跟attention進行對接，之後再將所產生的東西送入全連接層（Fully Connected Layer）。

Attention的產生過程：

首先我會先將encoder output和decoder 的hidden vector reshape之後cat在一起，然後餵到一個RNN裡面，RNN模型結構如下：



然後output就是我們所需的attention了。

1. (30%) Beam Search:
   1. 請詳細說明實做 beam search 的方法及參數設定，並分析結果。

每次更新的方法

藉由decoder得到每個單詞的機率

計算累加的機率且存起來

求出新的分支

根據新topk分支，更新參數

找出topk的最新單字，當作下一輪的輸入

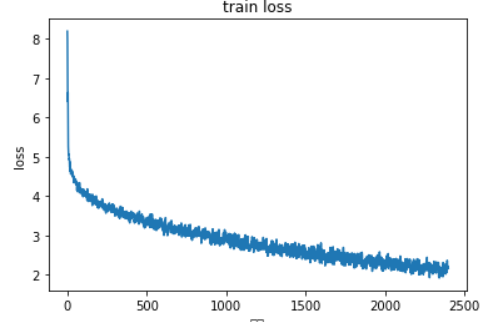
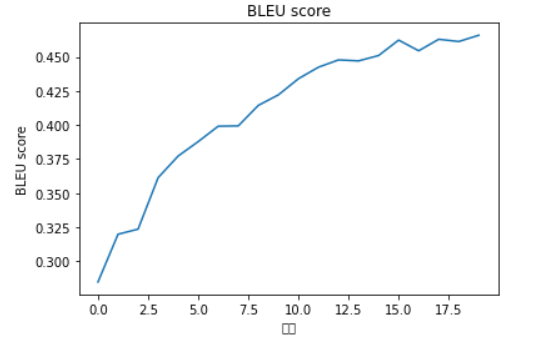
最後輸出紀錄的top1。

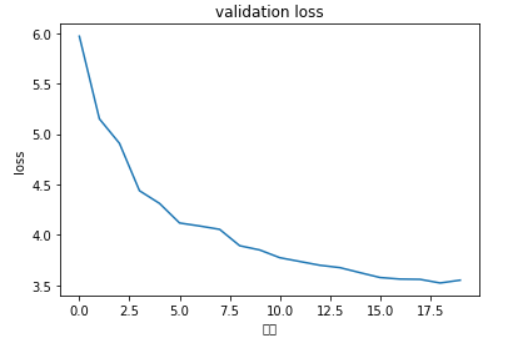
參數設定：

1. marked紀錄所走過的字
2. accuprob累積機率
3. probilitylist 紀錄topk分支進行下一次輸出後的topk機率
4. canlist 紀錄topk追蹤的單字編號
5. hlist紀錄topk分支的decoder hidden state
6. hidden\_buffer 的buffer

因為設定了不少變數，在這邊只舉出比較重要的幾個

實驗作圖：





Test: Without Beam Search:

test loss: 3.6322, bleu\_score: 0.4566

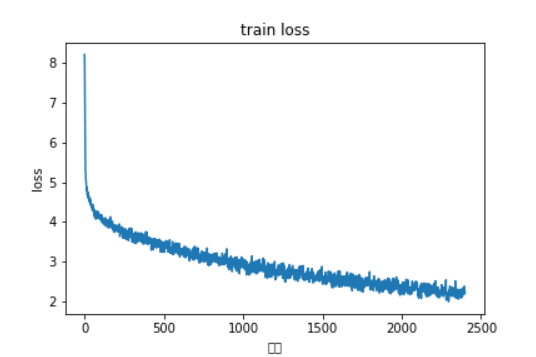
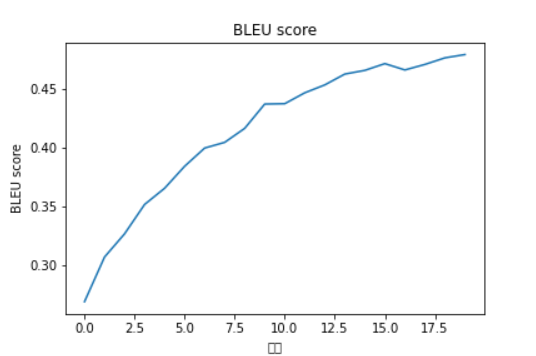
Test: With Beam Search:

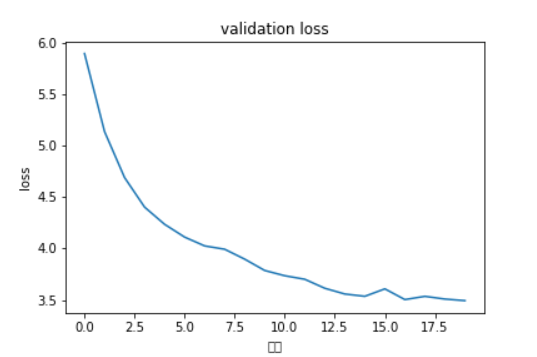
test loss: 3.6105, bleu\_score: 0.4393

這樣的結果跟預期的有些出入，沒有beam search得到的bleu score高於有beam search的bleu score。

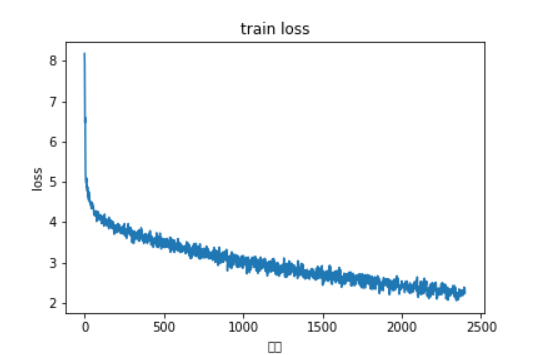
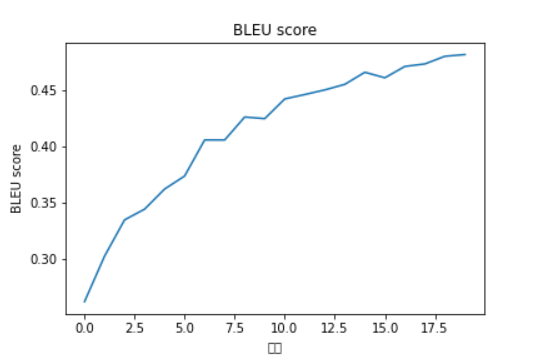
我想有可能是因為model沒有train好。或是在training的過程中，它將正確答案剪枝掉了，以至於得不到理想得答案。這部分就是beam search不太好控制的地方了，或許可以再用一些演算法改進。

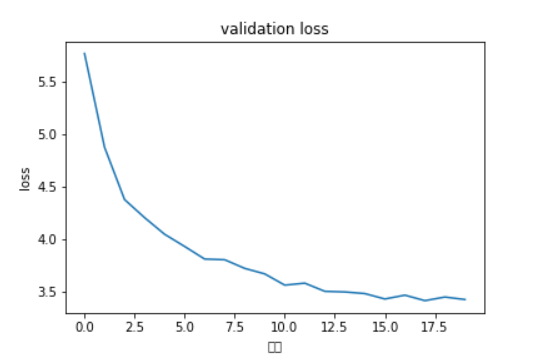
1. (20%) Schedule Sampling:
   1. 請至少實做 3 種 schedule sampling 的函數，並分析結果。
   2. 第一個schedule sampling



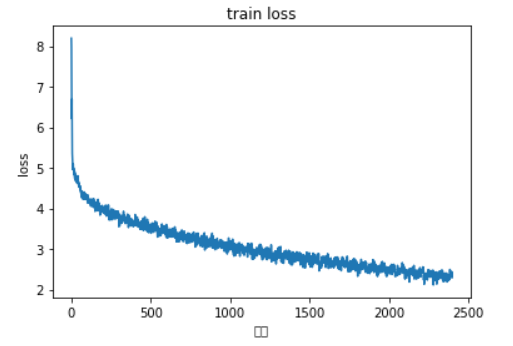
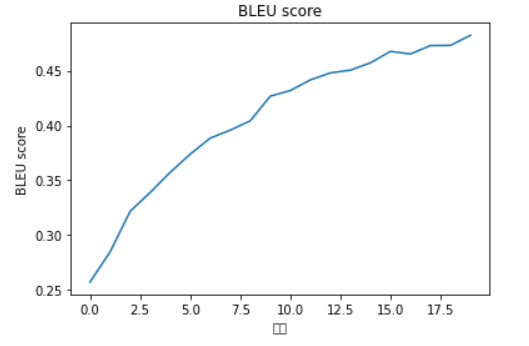


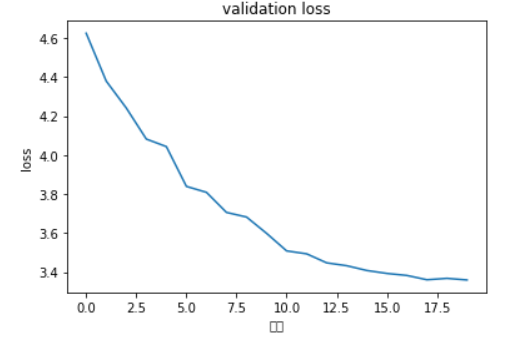
* 1. 第二個schedule sampling





* 1. 第三個schedule sampling





(a)linear

公式:teacher\_forcing\_ratio = max(bottom,init-slope\*step)

bottom=0.3，init=1，slope=0.03

結果:

val [12000] loss: 3.628, Perplexity: 37.756, bleu score: 0.458

test loss: 3.8174, bleu\_score: 0.4468

(2)exponential

公式: teacher\_forcing\_ratio= ratio\*\*step

ratio = 0.949

結果:

val [12000] loss: 3.436, Perplexity: 31.076, bleu score: 0.413

test loss: 3.621, bleu\_score: 0.4434

(3)inverse sigmoid

公式: teacher\_forcing\_ratio = tou/tou+exp(step/tou)

tou=6.218

結果:

val [12000] loss: 3.435, Perplexity: 29.241, bleu score: 0.428

test loss: 3.611, bleu\_score: 0.4455

在test bleu score方面，linear>inverse sigmoid>exponential，在test loss方面，inverse sigmoid<exponental<linear。

但是三者的bleu score 還是比teacher forcing的分數還要糟，我想可能是因為schedule參數需要再調整。