

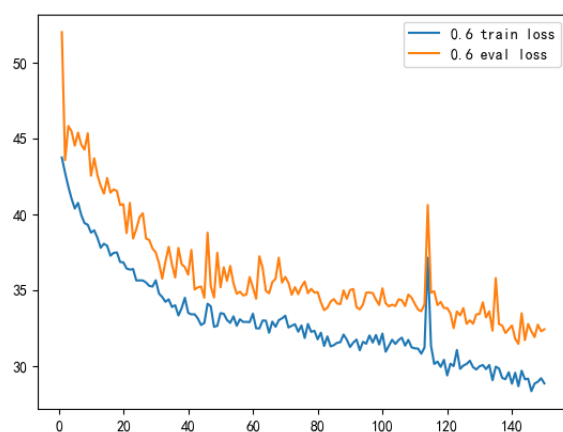
# MLDS 作業2-1

R05229014 鄒適文 r05229016 羅章碩

## Hw2-1 Video Captions

### 1. Model description(3%)

首先利用訓練資料中的詞做成字典，並將出現頻率小於3的字拿掉後字典大小為2971個字，接下來設定詞向量維度為64，並初始給隨機值，隨著 model 訓練的過程中更新詞向量。使用 encoder 的部份為4層的 LSTM，units 分別為1024,512,512,512，在 encoder 的部份使用 Luong 的 Attention 方法，Attention 中的 units 是1024。decoder 的部份第一層為1024的 LSTM，其 initial state 為 encoder output 的第一層加 Attention weight，之後通過一個層 Dense，activation function 為 relu，再通過 Softmax 得到最後的類似機率分佈，再使用 argmax 獲得最後答案。在訓練的過程中使用 scheduled sampling，使用正確答案的機率為40%，而在測試集的時候有使用過 beam search 和餵前一刻 model output 的方式來測試準確度。在訓練的過程中並沒有使用全部的 caption 搭配 video(共兩萬多筆資料)去做訓練，只拿了先 shuffle 過後的4250筆資料做訓練，batch size 調64，epochs 為150，使用的 optimizer 為 Adam，學習率為0.001。使用此 model 在 testing 上的平均 bleu score 為0.6795，而如果 train 到200 epochs 則可以到0.6995，但因為不小心把 train 到200 epochs 的 model 誤刪了，所以在 github 上只放分數為0.6795的模型。以下是在 training 過程中 cross entropy 的變化。可以看到 evaluation 的 cross entropy 都比較高，原因是 evaluation 時是使用 greedy sampling 的方式，所以 model 前面的字沒 predict 好可能後面就跟著錯下去，也就是所謂的一步錯，步步錯。



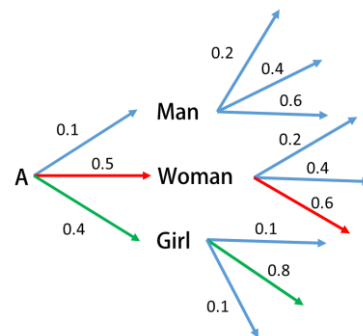
### 2. How to improve your performance(3%)

- write down the method that makes you outstanding.(1%)

我使用的方式為在 testing 的階段時使用 beam search 的技巧。





- Why do you use it.(1%)

因為在搜尋的過程中雖然在第一個詞彙的機率不是最高，但有可能接下來可以選到高機率的詞彙，所以全部的詞彙機率乘起來之後得到的機率可能是最高的，例如下圖中，一般利用 Argmax 會選到紅色那條路徑，但是這條路徑的機率是 $0.5 \times 0.6 = 0.3$ 並不是裡面最高的一條路徑，反而是綠色那條路徑 $0.4 \times 0.8 = 0.32$ 會比較高，因此使用 Argmax 有可能會錯過機率更高的那一條路徑，所以這邊我使用 beam search 的技巧，看可不可以將準確度提高。在 beam search 中，我使用的 beam width 設為5，保存全部機率最大的五筆結果。



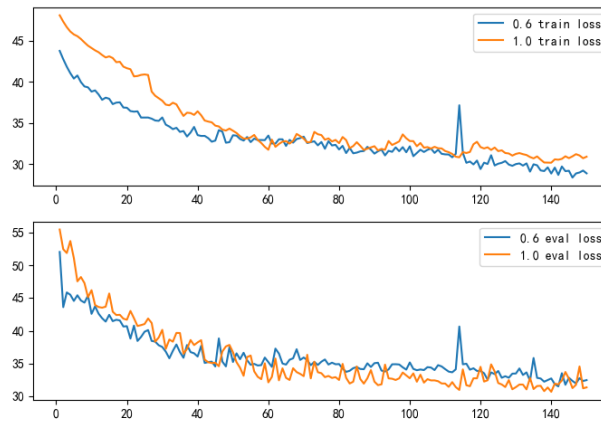
- **Analysis and compare your model without the method.(1%)**

在沒有使用 beam search 之前得到 bleu score 的分數為0.6995 (train 到 200 epochs 的模型)，其他五個結果的 bleu score 分別為0.6787、0.7017、0.6639、0.6700、0.6596，在這五個結果中可以看到有一個分數(0.7017)是比較高一點點的，不過因為使用 bleu score 不是一個很好的評斷方法，所以有特別去比較分數為0.7017這組的句子有沒有比0.6995來的好。比較之後可以看到0.7017這組的句子有比較多句子是比較好的，像是下面表格中第一和第二句，分數為0.7017的確實有講的比較精準，然後第三句兩個結果都還不錯，但是最後一句兩個都不好，尤其是0.6995的那個更是直接崩潰吐出 A is a of 不知道是想表達甚麼的句子。其實看了一下這些產生比較差的句子的影片後發現通常影片裡主角是動物(除了貓、狗外)或是整個場景很大主角很小的時候，通常 model 產生的句子都不好，第一個原因有可能是在訓練資料中比較常看到主角是人，造成 model 不太能分辨出動物，第二個原因可能是場景太大造成畫面裡東西太多，model 不太知道焦點該擺在哪所致。除此之外，還可以發現 model 產生的句子常常詞彙會重複，造成句子會變成 A man is a man 之類的句子，有可能是在 Attention 時沒有用 regularization term，造成 model 可能一直看同一個 frame 的東西而忽略了其他的 frame，這次在作業中沒有對 attention weight 使用 regularization term，或許這個部分可以當成未來改進的方向。

	分數為0.7017的模型	分數為0.6995的模型
	A man is cooking in a pot	A man is cooking a pot
	A woman is cutting a carrot	A woman is a carrot
	A man is playing a guitar	A man is playing guitar
	A girl is doing the of	A is a of

### 3. Experimental results and settings.(1%)

(1)Scheduled sampling：比較了0.6機率使用 greedy sampling 的方式和完全使用 greedy sampling 的方式，bleu score 分別得到0.6795和0.6656。從下圖 loss 隨 epochs 增加時的變化可以發現，完全使用 greedy sampling 的方式在 training loss 上都比較高一些，訓練過程中也下降的比較慢，但是在 testing 的過程中有一度比0.6機率使用 greedy sampling 的 loss 還低，代表說雖然 training 過程中學得比較慢，但是因為和 testing 一樣是全部都是 greedy sampling 所以並沒有 mismatch 的存在，所以最後 eval 的 cross entropy 差不多，不過 bleu score 上就有差一些。綜合以上所述，我認為之後可以嘗試在前面的過程中使用較高機率的 teacher forcing，之後學到一定程度後減低 teacher forcing 的機率，讓學習速度較快一些。



(2)Beam search：因為不小心誤刪 200 epochs 的模型，所以有重新訓練一個 150 epochs 的模型，發現在 200 epochs 的模型中，幾乎其他的 beam width 的 bleu score 都比較低，只有一條是高於原本的。但是在 150 epochs 的模型中，原本的 bleu score 為 0.6795，但是一樣開了 beam width 為 5 的結果中，可以發現其中三條 beam 分數都比較高(0.6870、0.6869、0.6800)，代表使用 beam search 的結果是好好有壞的。

(3)在 testing 時使用的小技巧：在 FB 社團留言中，有一位同學提到可以將機器 predict 的結果作處理，像是 a a man man is a man man，把前後重複的字詞刪掉變為 a man is a man。我也有嘗試對結果做這種處理，bleu score 從原本 0.65 上升到 0.67，如果單純是想衝高 bleu score 這是一種小技巧，不過最好還是把 model train 好才是最佳解決辦法 XD

## 分工表：

R05229016 羅章碩負責 2-1

R05229014 鄒適文負責 2-2