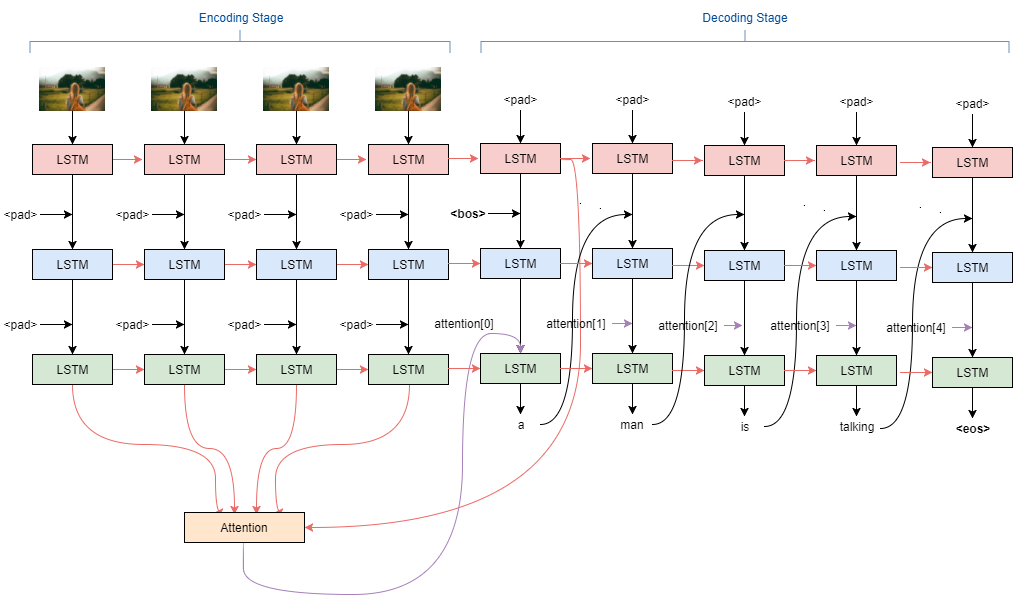
ADL x MLDS Fall 2017

Homework 2 Report

Student: CSIE R06922068 Yu-Jing Lin 林裕景

# Model Description

以下是我這次作業所建構的模型架構圖：

使用Python+TensorFlow套件進行實作。

參考〈Sequence to Sequence -- Video to Text〉的S2VT模型，流程像seq2seq分為encode和decode兩個stage，先輸入完所有影片的frames後再開始預測句子，不同處為decode時的文字並非傳到第一層RNN而是第二層，讓兩個RNN分別學習影片和句子的特徵。除此之外，我也加上Attention mechanism，使模型在預測時可以focus在前面encode影片時的不同time step，以提升預測句子時的精確度。

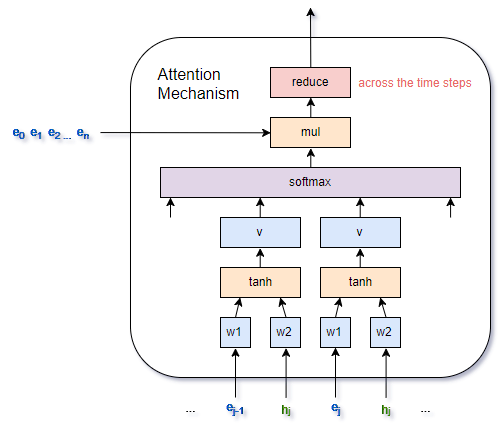
資料preprocessing的部分，我做了一個MSVD dataset loader，便於讀取training和testing的資料，以及sentence encoder來做word tokenizing和詞頻統計。

我在S2VT模型中的RNN使用了BasicLSTMCell或GRUCell，可以用gradient descent、Adam、RMSProp三種optimizer，有Dropout layer的設置，可以自訂outputs的keep propagating rate，並implement了scheduled sampling，支持linear和exponential兩種decaying rate。

此外，還有依照詞頻調整的initial projecting bias。並支持pre-trained word embedding的Word2Vec、GloVe和FastText初始化S2VT模型中的word embedding。這兩個方法讓模型在訓練時收斂速度加快。

# Attention Mechanism

我使用的attention mechanism參考了Dzmitry Bahdanau提出的global attention mechanism，參考了全局的狀態，另外還有人提出local attention mechanism但是在這裡我沒有實作。使用在encoding時所有第一層的RNN之state h，加上decoding時目前第二層的RNN state h，也就是接受完前一個字詞後的state h。經過以下架構的attention mechanism後輸出作為當前step的第三層RNN之輸入。



我將encoding時第一層RNN在每個time step的state h（以下稱為enc state）乘上一個矩陣w1，將當前的第二層RNN之state h乘上一個矩陣w2並複製time step個，以符合encoding time step的數量，然後兩兩做element-wise的加法後經過非線性的tanh函數，再乘上v矩陣，最後對time step維度做softmax，得到的矩陣（或稱做mask）為每個time step該取百分之多少的enc state。

將這個mask與enc state做element-wise相乘後，即為此階段要輸入給第三層RNN的attention。

其中，w1、w2和v就是這個attention mechanism在training phase時會訓練到的變數。

# How to improve your performance?

起初，我先用一般的seq2seq做這次task，主要使用TensorFlow中的seq2seq套件，建立基本的seq2seq模型，並以這個model做special mission。後來加上DropoutWrapper、MultiRNNCell、AttentionWrapper和BeamSearchDecoder等等，都用上後，覺得預測的語句似乎沒有很大的提升，於是轉向助教在投影片中提到的S2VT模型。

用TensorFlow實作了S2VT後，發現效果其實也差不多，於是實作了一些其他的機制，如同避免overfitting的Dropout layer，避免exposure bias的scheduled sampling。以及最重要的attention mechanism，原先是兩層RNN架構，attention吃前一次的state，但在幾次實驗後，覺得這樣的架構不合適，因而改為三層RNN構造。後來也嘗試調整句子的長度和過濾掉出現次數較少的文字。經過以上多次實驗後，我的model預測出來的BLEU分數明顯上升許多。

而我也發現，由於每支影片對應十幾個描述，若在training時只使用同一個句子很容易overfit預設的句子，造成預測時大失準，於是我在每個epoch時給每段影片隨機抽取一個句子，增加多樣性，並且每個epoch還隨機排序訓練影片的順序，避免模型從data固定的順序中學到不該學的東西。

再來，我繼續思考可以做甚麼樣的改進，想到既然我們用了word embedding在模型裡面，是不是可以匯入外面的pre-trained word embedding，於是我抽取了training data中出現的文字，從Word2Vec、GloVe和FastText三種方法中拿出對應的embedding vectors，存成numpy array，再餵給S2VT模型。然而，許多出現的字沒有在pre-trained embeddings中，常見的stop words、極罕見的字，甚至有阿拉伯文字，因此這個word embedding得是「可訓練的」。經過幾次實驗後，發現雖然pre-trained embeddings可以縮短收店時間，然而到後來的loss和accuracy都差不多，我猜測是這些embedding終究還是會被訓練成差不多的樣子。

# Experimental Results and Settings

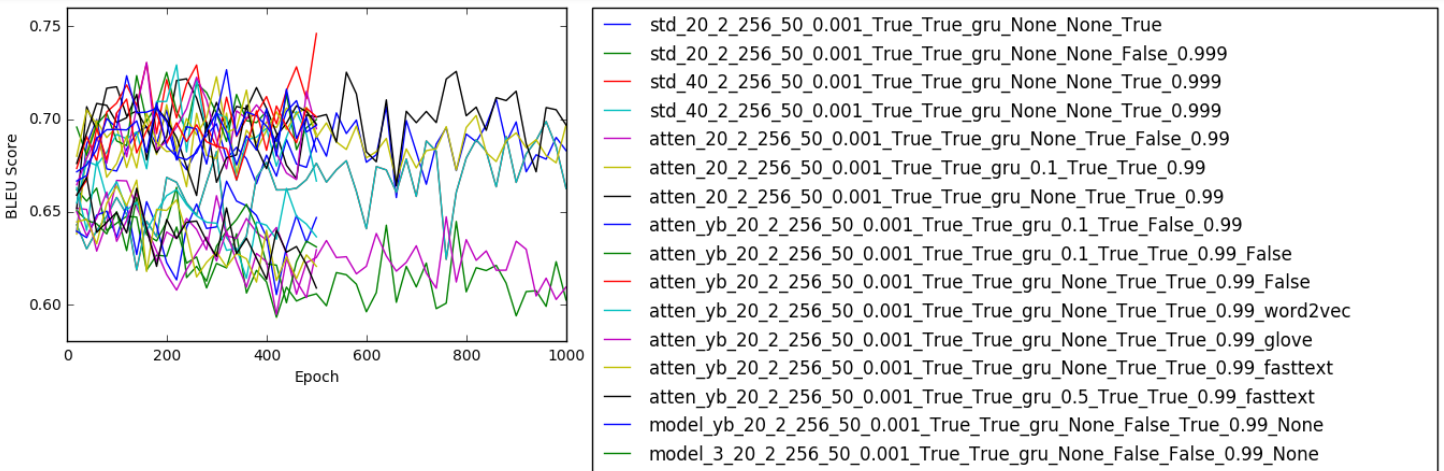
**我的實驗環境是：**

* CPU: Intel Xeon處理器E3-1230 v3
* RAM: 8G
* GPU: GeForce GTX 980
* OS: CentOS Linux 7

使用的工具為Python 3.5.4 with TensorFlow 1.3.0 and Keras 2.0.7。

**我的實驗結果為：**

每20 epoch一次，以新制BLEU score計算，詳細的圖表在results資料夾中。



模型名稱的格式為〈模型名稱\_label長度\_濾字門檻\_RNN單元數\_batch大小\_學習係數\_是否隨機label\_是否抽換順序\_RNN種類\_dropout率\_是否attention\_是否sampling\_sampling的遞減係數\_sampling遞減模式\_sampling遞減每epoch數\_pre-train的embeddings〉。每次實驗的epoch數有500也有1000，所以圖表中的折線長短不一。

從分數的部分觀察，可以注意到幾個有趣的地方：（可以到results資料夾中看分類的結果）

### Scheduled Sampling提升

使用了scheduled sampling的結果之BLEU分數普遍維持在0.7分，有些甚至到0.74接近0.75分，因為模型透過自己的output來判斷下個字，模擬出在預測時沒有true label可以參考的情況；而沒使用scheduled sampling的結果會因為overfitting，分數越來越低，到1000 epoch左右剩下約0.6分。

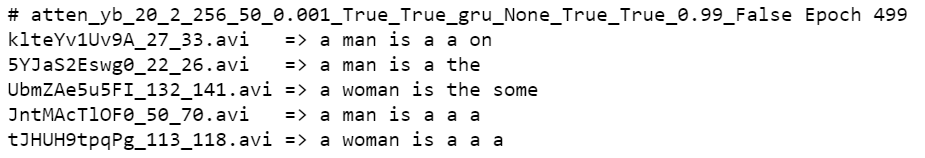
### Attention Mechanism對於分數沒有多大幫助

每種組合都嘗試過的結果發現attention mechanism對分數沒什麼幫助，但是比較過有無使用attention的模型所預測出來的結果後，有使用的句子似乎有稍微合理一點，這是分數所看不到的東西。

### Dropout layer的沒必要性

沒有必要使用Dropout layer，因為overfitting在我這次的實驗影響不大，應該說使用dropout layer後，結果也沒有比較好。

然而，許多BLEU分數高的預測結果，打開一看：（以全部分數最高的為例，單看special mission）



**<- BLEU分數：0.74多**

結論：分數一點都不重要！（？

應該說，分數高確實表示比較多的正解文字被包含在預測內（可能主要是不重要的非關鍵字），但因為描述語句是主觀的東西，單靠比對所謂的標準答案並無法正確的表現出結果的正確性，因此這次作業才會使用peer review的方式評分吧！

# Reference

Sequence to Sequence Learning with Neural Networks (2014). Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Available at: https://arxiv.org/abs/1409.3215.

Sequence to Sequence -- Video to Text (2015). Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, Kate Saenko. Available at: https://arxiv.org/abs/1505.00487.

Neural Machine Translation by Jointly Learning to Align and Translate (2014). Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. Available at: https://arxiv.org/abs/1409.0473.

Sequence To Sequence Attention Models In DyNet (Github, 2016). Tal Baumel. Available at: https://talbaumel.github.io/attention/.

Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. Samy Bengio, Oriol Vinyals, Navdeep Jaitly, Noam Shazeer. Available at: https://arxiv.org/abs/1506.03099.

Recurrent Neural Network Regularization (2014). Wojciech Zaremba, Ilya Sutskever, Oriol Vinyals. Available at: https://arxiv.org/abs/1409.2329.