2-2 Chat bot

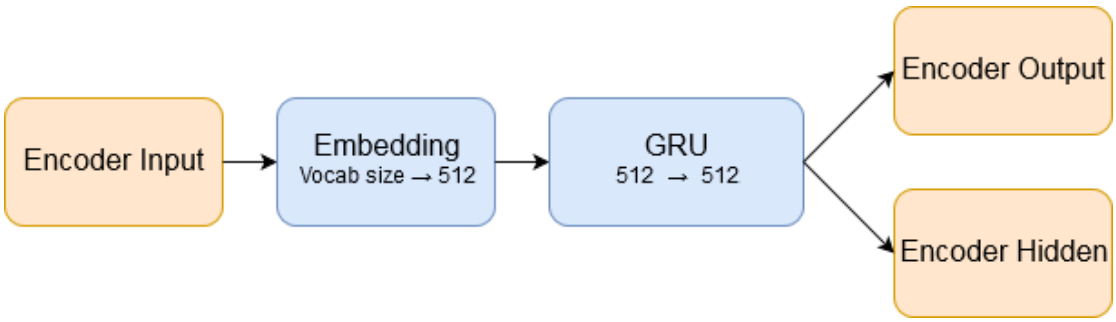
組員：b03902101 楊力權 b03902102 廖廷浩 b03902093 張庭維

一、Model Description

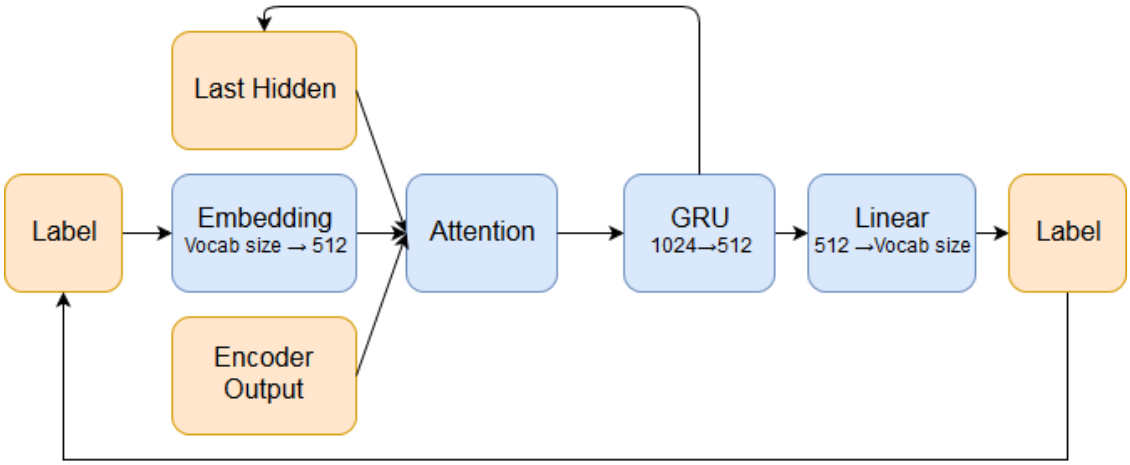
(a.) Preprocess

取 clr_conversation 裡的詞來做 tokenize，為了減少太多不常見的詞造成運算數度及資源不太必要的消耗，因此只選取至少出現 20 次以上的詞

(b.) Encoder:



(c.) Decoder



(d.) Train Detail:

- Optimizer: SGD
- Learning Rate: 0.1
- Loss Function: CrossEntropy

二、How to improve your performance

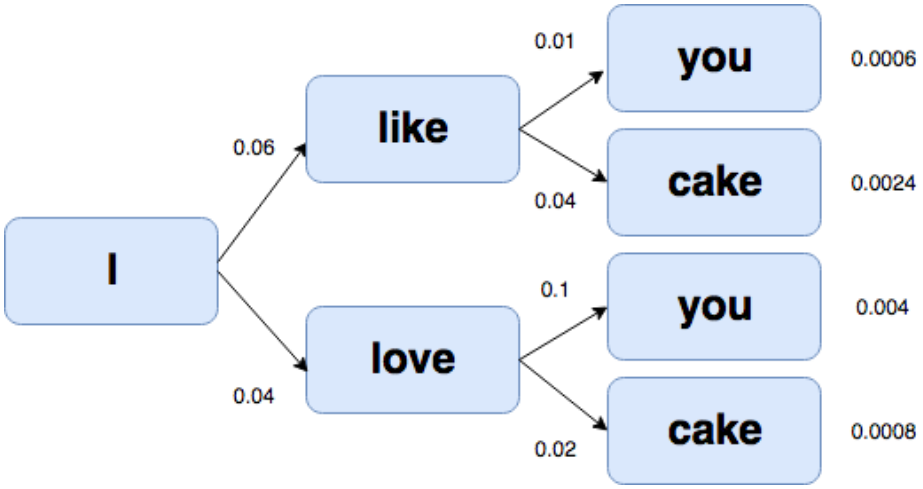
這次 2-2 的 project，我們嘗試了 Beam Search、Word2vec 及 Attention，而 Attention 的表現在 2-1 時討論過了，因此不在此多做說明。

(a.) Beam Search

Brief：

對於一句句子來說，每一個詞之間都有存在一定的關係，因此在算機率的時候，有可能某個字後面出現機率最高的詞，再接上第三個字之後並不會比另外

連續的三個字的詞還要高，譬如： I like 雖然大於 I love，但 I love you 出現機率卻是三個連字中最高的，因此在 test 時利用 beam search 可以找到更高機率會出現的句子。



Implement:

利用 torch 的 topk function 可以取得在最終 Linear Layer 的 Output 中前 n 大的 index 及他們的 value，我們以這些 value 當作評斷他們機率高低的標準來作 Beam Search，而取前 n 大的 n 即為 Beam width。由於最後一層並沒有過 Softmax，因此 score 是用上下層的 value 相加算得的(利用 softmax 當機率的方法是將 value 放在 e 的次方項，因此有過 softmax(機率)的值相乘，應該與沒過 softmax 的 value 相加的值有正相關)

因為 Beam Width 越大，運算時間可能過久，因此我們只取 3(僅利用 Output Vector 前 3 大的值)，並用 window=2 或 window=3 做實驗(看後面一個，或是看後面兩個)

Beam Search Window	Perplexity	Correlation Score
1(without beam search)	59.457866	0.29292
2	55.488428	0.30280
3	46.892638	0.30201

Sum:

做 Beam Search 的時間遠較沒有做的久(window=2 時大約 3 至 5 倍, window=3 時 20 倍以上)，並在 Correlation Score 上並無大幅度的提高。Window size = 2 時 Correlation 的表現較 Window size = 3 時更佳，有點令人意外，在觀察輸出的結果時，發現 Window=3 重複的句法相對的多不少(例如：'我在我在我在')，不確定結果為何如此。

另外，做 Beam Search 的 model 是使用比較早期的 model，因此 Correlation Score 當時還比較低。

(b.) Pre-trained Word2Vector

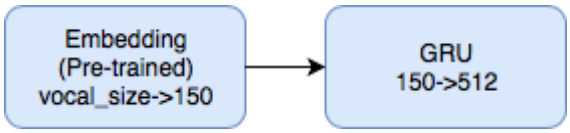
- Brief：

在原本的結構中，要把每個單字都 map 到自己的 embedding，然後再透過訓練

時一起訓練這些 embedding，但使用這個方法，還不如先用更大的 dataset 訓練一個 Word2Vector，而把訓練好的 embedding 直接餵給 model，訓練時就不必再訓練他；不但可以省時間還可以拿到更精準的 word embedding。

- Implement：

使用 Gensim 套件，使用維基提供中文 dataset，再使用 jieba 斷詞，用 Gensim 作成維度 150 的 word to vector model，所以架構中的 embedding layer 變成如下



- Result：

在使用 schedule_sampling_ratio=0.5，dim=256，使用 attention 的狀況下。

	Perplexity	Correlation Score
不使用 pre-trained W2V	39.84	0.32133
使用 pre-trained W2V	44.96	0.30253

做出來發現加入 Word2Vector 的結果反而退步，可能因為 dataset 的差異(不論是字彙的規模或是字彙的用途)，導致結果不好，又有可能是訓練 Word2Vector 前的前處理，做得不夠好(實驗後才發現 word2vector model 中根本沒有「你」這個字)。此外由於最後時間不太夠，因此沒有辦法對 Word2Vec 上述的問題做調整。

三、Experiment Result and Setting

(a.)對 dimension 做實驗

同樣使用 Attention，num_layer=1

dim	Perplexity	Correlation
128	33.37	0.3018
256	35.84	0.3213
512	31.23	0.3373
1024	42.79	0.3213

只要 dimension 大於 256 結果就差不多，而 512 得到結果最好。

(b.)對 layer_num 做實驗

同樣使用 Attention，dim=512，但是 layer_num=2 以及 layer_num=4 時完全 train 不起來，loss 永遠在數百且完全不下降。

四、心得

這個作業做得很挫折，因為做了超過一週，而 train 一次又要幾乎一天，loss 都會降下來，但是結果仍然無法過 Correlation baseline，不論怎麼調整結構或想新的方法(attention、beam search、word to vector 等)，以及調參數都還是與 baseline 離得很遠。