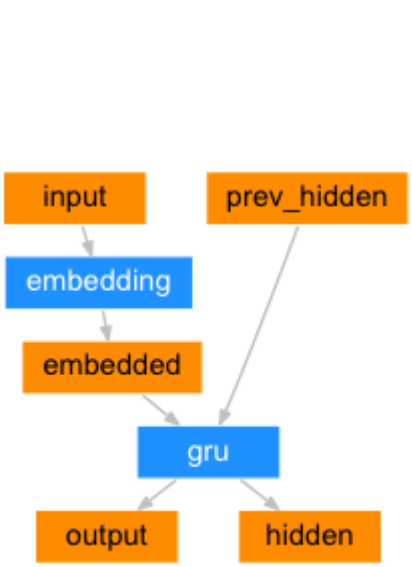
HW2-2

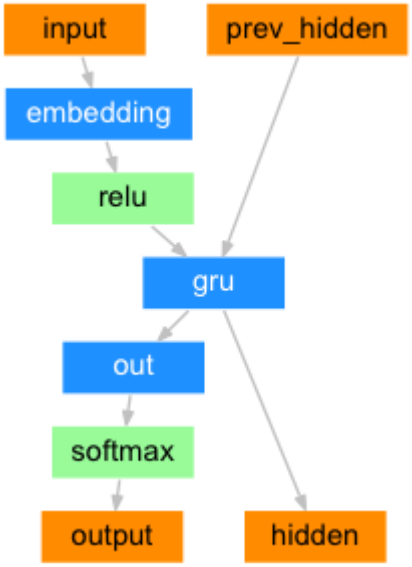
ID & Name : R06946009林庭宇、R06946006李筑真、R06946015黃永翰

1. Model description (2%)

- Describe your seq2seq model



Encoder



Decoder

在Encoder的部分，input先通過embedding layer後，再丟入gru裡面訓練，而gru最後一個time step所吐出的向量，彙整了整個句子的意思，稱為context vector，將context vector作為Decoder隱藏層的初始值，而Decoder每個time step的output經過softmax後會變成每個字詞所對應的機率，機率最高的即為此timestep所預測的字詞，當預測出<EOS>Token代表整句預測完畢。

```
EncoderRNN(
  (embedding): Embedding(7284, 512)
  (gru): GRU(512, 512, num_layers=2, dropout=0.1, bidirectional=True)
)
DecoderRNN(
  (embedding): Embedding(7284, 512)
  (embedding_dropout): Dropout(p=0.1)
  (gru): GRU(512, 512, num_layers=2, dropout=0.1)
  (fc): Linear(in_features=512, out_features=512)
  (out): Linear(in_features=512, out_features=7284)
  (softmax): LogSoftmax()
)
```

model的詳細架構

2. How to improve your performance (3%)
(Please do the method different with hw2-1)

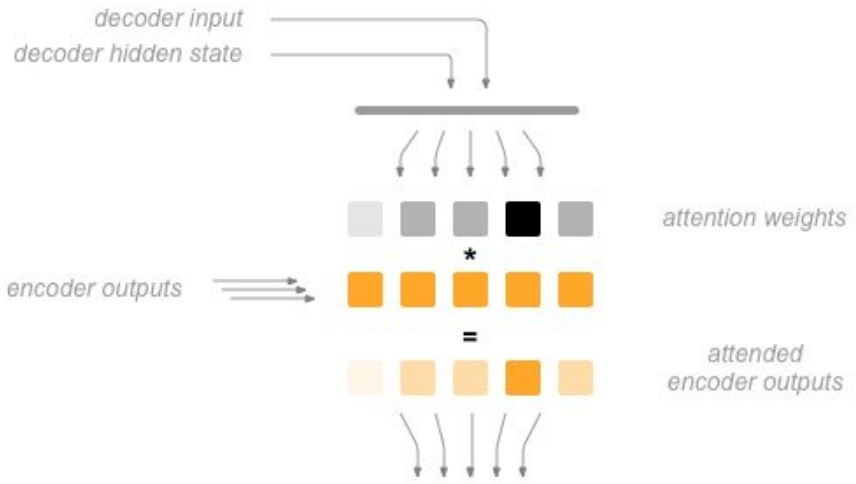
- Write down the method that makes you outstanding (1%)

Attention

```
EncoderRNN(
  (embedding): Embedding(7284, 512)
  (gru): GRU(512, 512, num_layers=2, dropout=0.1, bidirectional=True)
)
LuongAttnDecoderRNN(
  (embedding): Embedding(7284, 512)
  (embedding_dropout): Dropout(p=0.1)
  (gru): GRU(512, 512, num_layers=2, dropout=0.1)
  (concat): Linear(in_features=1024, out_features=512)
  (out): Linear(in_features=512, out_features=7284)
  (attn): Attn(
  )
)
```

model with attention的詳細架構

- Why do you use it (1%)



一言以蔽之，利用Attention來模擬人類的注意力機制，在每個time step給予句子中重要的部分較高的權重，以此增進模型Performance。attention的計算仰賴於decoder的hidden state和對應來源的enocder hidden state。將算出來的scores正規化使其所有加總起來為1，而在score計算上我們採用簡單的states間相乘去計算，如下式。

$$score(h_t, \bar{h}_s) = h_t^\top \bar{h}_s$$

$$a_t(s) = align(h_t, \bar{h}_s) = \frac{exp(score(h_t, \bar{h}_s))}{\sum_{s'} exp(score(h_t, \bar{h}_{s'}))}$$

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

	沒加Attention (50000 epochs)	沒加Attention (120000 epochs)	有加Attention (50000 epochs)	有加Attention (120000 epochs)
Perplexity	13.61967	14.76246	19.06878	21.53715
Correlation Score	0.53646	0.53656	0.60815	0.61933
Training Loss	3.32350	3.15610	3.49268	3.20182

加入Attention後Correlation Score增加，代表Performance有所上升，但是Perplexity反而增加，這裡推測是chat-bot可以產生更豐富的回答，也因此預測下一個字的不確定性跟著變高，因此Perplexity上升。

3. Experimental results and settings (1%)

- parameter tuning, schedual sampling ... etc

Training iteration = 120000

- Batch size = 64
- GRU dimension = 512 2 layers
- Learning rate = 0.0001
- Gredient clip =50
- Adam Optimizer
- Dropout = 0.1
- (MIN_LENGTH, MAX_LENGTH) = (5, 26)

討論在20000 iters時，參數調整後performance的變化:

	基於上述設定	gredient clip改為30	learning rate改成0.005	單層神經元數改成256	batch size設成128
training loss	3.93155	3.76152	101.75742	4.00716	3.84409
perplexity	10.24895	10.63605	109.60610	8.71469	14.26948

correlation score	0.48715	0.50457	生成句子太長無法算	0.45436	0.52402

首先，發現不管如何，訓練的iteration變多，perplexity就會變大，理論上是越小越好，代表我們見過的句子出現的概率越大，所以從另一個角度切入，當我們生成新句子的能力上升，很有可能也使perplexity提高。所以在這裡比較時我們參考在correlation score這個指標去判斷誰的生成比較好，得到一個結論是將gredient clip調小跟batch size設大一點可以加速收斂。

在iterations=120000時，沒有attention跟有加attention的句子生成比較：

輸入句：你的夢想是什麼？
標準答句：我想要有個真正的法國女僕 人
沒加attention：我的夢想是什麼?
有加attention：我的夢想是,我的夢想

輸入句：孩子,不會有事 的
標準答句：爸爸？爸爸，他還好嗎？
沒加attention：我不知道,我猜他是我兒子
有加attention：爸爸,我不想聽你的

輸入句：有些 時候，人們 有權 得到 更 多
標準答句：有些 時候，人們 的 信念 必須 得到 回報
沒加attention：人們總是互相誤解,明白嗎?
有加attention：有些時候,人們都得到回報

在隨機抓的三個句子中，有加attention的句子回答得比較好。另外句子的生成看起來算是通順的，這可以歸功於使用teacher forcing去訓練。

分工表

學號姓名	負責工作	比例(%)
R06946009 林庭宇	HW2-2	25
R06946006 李筑真	HW2-2	25
R06946015 黃永翰	HW2-1	50