HW2-2: Chatbot

Model description (A,B,C)

Describe your seq2seq model (2%)

基本的架構和 HW2-1 相同，不再贅述，不同處會在以下說明。

- 我們最好的 model 使用 word-based 去做的，而 word vector 為 100 維 (用 gensim 的 fastText 先 pretrain 好，之後跟著 model 一起 fine-tune)。

- 字典取 min count = 10，有 40000 多個 words

- Training data 取整個 dataset，本來因為 dataset 太多，train 一個 epoch 要 4.5 個小時，但是後來發現把 Training data 弄小一點並不能 train 得比較好。

- Encoder 餵 100 維的 word vector，並且拿最後一個 hidden state 當作 decoder 的 initial hidden state(並非 trainable 的)，另外就是因為我們是拿最後的 hidden state，所以 input 的 padding 我們會做在前面，而 output 則是會 PAD 在 <EOS>後面。

- Decoder 餵兩個東西的 concatenation: ground truth 的 word embedding，或自己前一個 output word 的 embedding (依照 scheduled sampling 去決定選取的機率); 以及 attention-weighted (用 decoder hidden state 跟 encoder outputs 算出來的) sum of encoder outputs。

- Decoder 和 Encoder 都是使用 256 的 GRU 兩層。
- Attention 的做法和 HW2-1 相同。

- Loss 只算 ground truth <EOS> 之前的 token 去做 cross entropy，並且用 RMSprop optimizer, learn rate = 0.001
- 總共 train 了三個 epoch，每一個 epoch 有 270 萬筆 data，batch 大小為 128。
- Testing 的時候 batch 取 100，會在第三部分說明。

- 做出來的結果: correlation = 0.58464
  Perplexity = 7.836960

How to improve your performance (e.g. attention, scheduled sampling, beam search...) (A,B,C)

Write down the method that makes performance outstanding (1%)

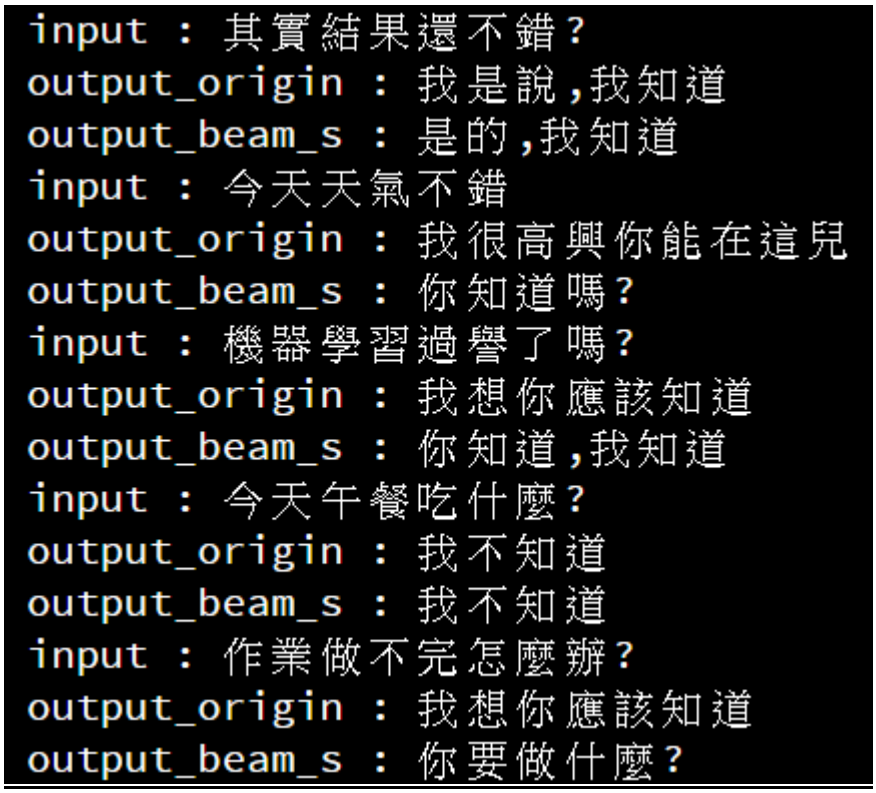
- 我們在這題除了實做了 2-1 的 attention 以及 schedule sampling 以外，還實做了 Beam search
- Beamsearch 就是並非在每個時間點都對 distribution 取 argmax，因為這樣可能最後得到的成果不好，如果在每個 infer 的時間點 keep 住 k 條路線(相乘機率最大的)，這樣子得到了 k 條路線後最後再取機率最高的那條路線。

Why do you use it (1%)

- 我們原本是沒有做 beamsearch 的，但是 output 出來的結果會發現怎麼幾乎所有的回應都是用”我”來開頭，而有一些出來的結果還非常差，這時候就想到了 beamsearch，可以避免掉一直用"我"開頭的情況，讓 output 更活潑一點而且回答得比較順暢。

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

- 以下是實際做出來的結果



- 沒有用 beamsearch 的(output_origin)幾乎都是”我”開頭，而使用了 beamsearch 之後可以發現 output 有非常大的變化，雖然有些和原本的相同，但是絕大多數的 output 都發生了改變，結果還不錯。

- 在這裡 beamsearch 的 K 我選用了 2，因為只要 K 大一點點整體的速度就會慢超多，另外就是在 output testing data 的時候我並沒有採用這個技術，理由還是一樣，太慢了.....10 分鐘內無法跑完 test_input.txt，即使是 K=2 都會超時，因此雖然這個技術的效果不錯，但是卻無法用來處理大量的資料。

Experimental results and settings (A,B,C)

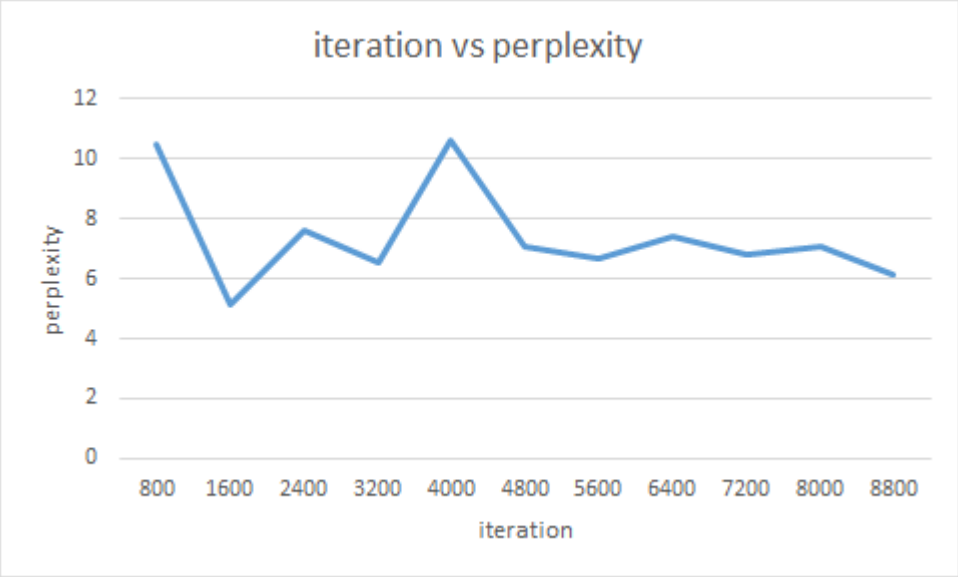
Describe hyperparameter tuning, scheduled sampling... etc (1%)

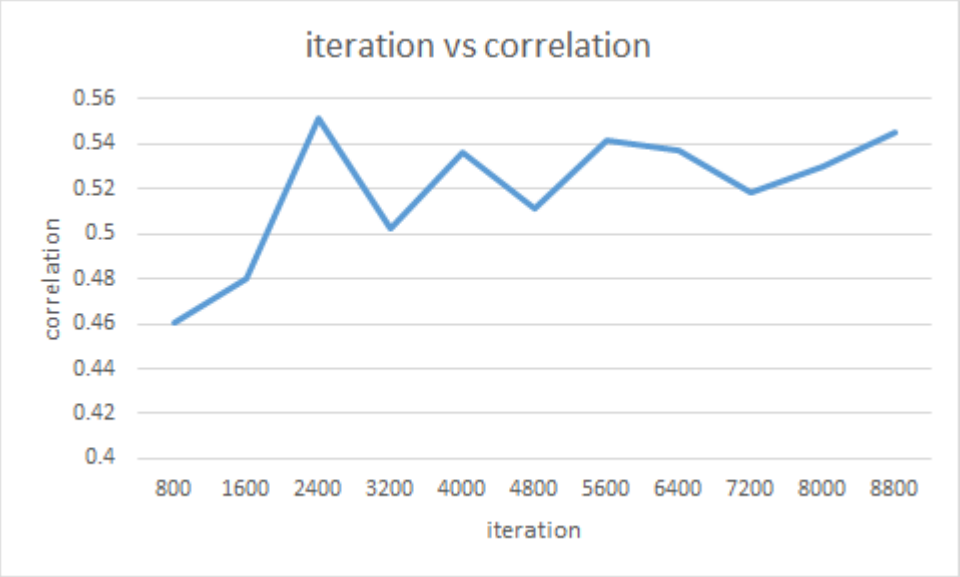
1. **Testing batch**

因為 testing data 太多，所以需要用 batch 的形式來做，但是會產生一個大問題，那就是 padding 的長度會不同，也就是必須要 pad 到跟最長的 input 一樣長才行，舉個例子，<PAD><BOS>文字<EOS> 和 <BOS>文字<EOS> output 出來的結果會不太一樣，而 batch 越大，出現長句的可能性會越高，因此出來的答案會非常不一樣。

2. **Iteration v.s. Correlation & Perplexity**

以下的這張圖是 iteration 對兩項指標的做圖





從這兩張圖可以發現，從第 2400 個 iteration 之後其實 perplexity 和 correlation 都沒有差很多，主要都是在上下震盪，loss 也沒有 improve，卡在 4.7 附近。

3. 1 layer GRU v.s. 2 layer GRU

以下是實驗結果(根據上一題的實驗結果，使用了 train 了 10000 個 iteration 的做比較)

1 layer 512	2 layer 256
Correlation = 0.53654	Correlation = 0.53731
Perplexity = 7.405	Perplexity = 10.594

可以發現兩種的 correlation 差不多，而 perplexity 則是 2layer 的比較大。

4. Learning rate

我們發現 lr = 0.001~0.0001 這個 order 的效果最好，loss 掉的速度最快，如果是 0.01 的話，一開始 loss 會在 2 位數徘徊，最後再慢慢得掉下去(因為 learning rate 是 adaptive 的)，而 0.00001 的話則是會爛掉。

5. 字典大小

一開始字典開的蠻小的，因為 train 出來會有太多的<UNK>，所以最後還是把字典開大一點。