Applied Deep Learning

Homework 1

Q1: Data processing

	自己建的	使用助教 sample code
Tokenize data	nltk.word_tokenize	Spacy en_core_web_sm
Turncation length	max_sum=307 max_text=541	max_sum=80 max_text=300
Pretrain embedding	word2vec-google-news-300	glove.840B.300d

自己建的 processing 是用 nltk tokenize 的和助教使用的 spacy 略有不同，而我自己建的 turncation length 則是取 trainData、valData 裡的每個的 max_text、max_sum 的 length，助教 smaple code 部份我沒有調整參數，pretrain embedding 的部分我是使用 word2vec-google-news-300 當 pretrain, 再用 corpus(train、val、test 裡所有的 word)來 fine tune word vector，利用 genism Word2Vec Model train 5 個 epoch ，沒有使用 lowercase、也沒有設定 unk，所以最後 vocab_size 約 24 萬個字、而助教 sample code 部分則是用 glove.840B.300d 當作 wordvector，並且有使用 unk 及 lowercase，vocab_size 約 9 萬多字。

之後的 extractive.sh, seq2seq.sh, attention.sh 皆是使用助教的 sample code 因為在我的 preprocess 下 extractive 的表現不好最佳狀況約:
rouge-1: 0.1658 | rouge-2: 0.0256 |rouge-l: 0.1145
而在助教的 sample code 下表現提升許多約:
rouge-1: 0.1835 | rouge-2: 0.0290 |rouge-l: 0.1271

可能造成的原因應該是 max_sum、max_text 太大了 padding 增多，再加上 vocab_size 又很大 model 要能完全學習很困難。

Q2: Describe your extractive summarization model

a. Describe model
Input(batch,seq_len) embedding(vocab_size,hidden=300)→
embedded(batch,seq_len,300)
embedded(batch,seq_len,300) biLSTM → out(batch,seq_len,600)
out(batch,seq_len,600) Linear(600,2)+sigmoid → out(batch,seq,2)

b. performance of your model.
rouge-1: 0.186、rouge-2: 0.0286、rouge-L: 0.130 (when epoch=30)

c. loss function
BCEwithLogitsLoss,其中 pos_weight 設為 11.63

d. optimization algorithm
adam、 lr=0.001 、batch_size=64

e. Post-processing strategy.
每筆 data 裡取 sentence 中有包含最多 target word 的前兩名 sentence 當作最後的 summary，所以每筆 summary 做多會有兩個 sentence 最少則沒有 sentence。

Q3: Describe your Seq2Seq + Attention model.

a. Describe model
attnEncoder:
input(batch,seq,) embedding(vocab_size,hidden=300)→
(batch,seq_len,300)
Embedded(batch,seq_len,300) biLSTM → out(batch,seq_len,600),hidden 取最後一個 seq 的 hidden(batch,1,300)

decoder_hidden=hidden
decoder_key=out

一次進去一個一個 word(第一個<sos>)

attnDecoder:
input(batch,1) embedding(vocab_size,hidden=300)→ (batch,1,300)

Embedded(batch,seq_len,300) GRU(hidden=decoder_hidden) →
gruout(batch,1,300),hidden(batch,1,300)(下一個 word 要用的 hidden)

decoder_key (batch,seq_len,600) Linear(600,300)→
key=value (batch,seq_len,300)

gruout(batch,1,300).transpose(1,2) → query(batch,300,1)

softmax(key*query) → attn_weight (batch,seq_len,1)
attn_weight.transpose(1,2)*value → weight_sum (batch,1,300)

weight_sum,query concat→ out(batch,1,600)
out Linear(600,vocab_size)+Logsoftmax→ out(batch,1,vaocab_size)

接著再進去下一個 summary word 直到 summary word_len(80)結束、此處使用的是老師上課教的 general attn，不過 train 不太起來、debug 很久找步道原因…

b. performance of your model.

rouge-1: 0.2282 rouge-2: 0.0564 rouge-l: 0.1866

c. loss function

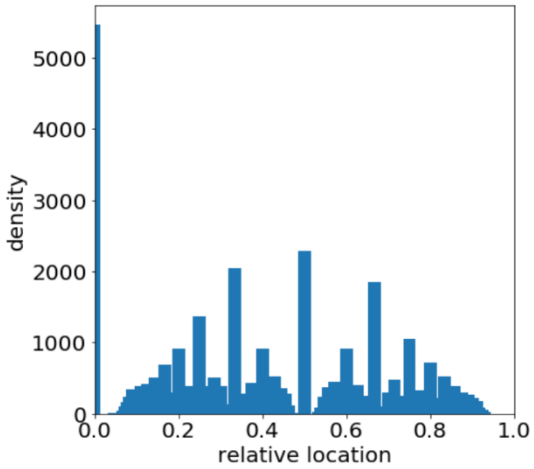
NLLLoss with ignore_idx=padding_idx

d. optimization algorithm

adam、 lr=0.001 、batch_size=32

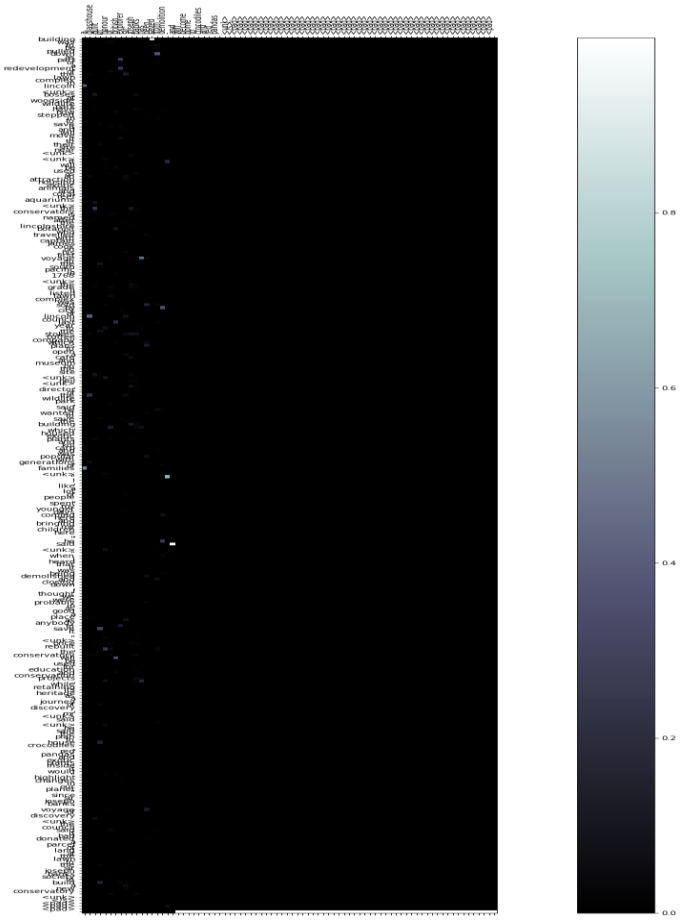
Q4:Plot the distribution of relative locations (1%)

下圖是針對 validation dataset(20000 筆資料)所做的 relative locations plot

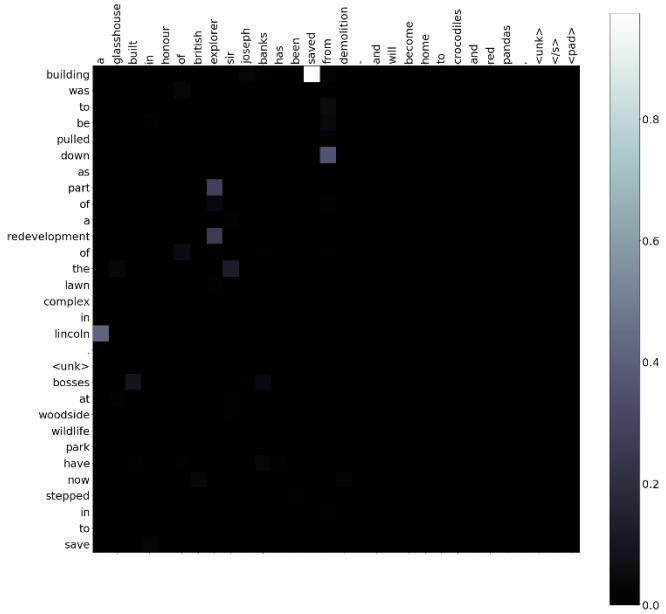


結果顯示大部分預測的 data 會選取在第一個句子、最後一句則完全沒被選到過，其他地方分布蠻平均的，不過很奇異的是 relative location 在 0~1 之間很像是對稱的。

Q5: Visualize the attention weights (2%).



左圖 x 軸為 summary t、y 軸為 text，由左圖可看到下方一直條白線，代表 text 最後一個 pad 貢獻給 summary 部分後半的 pad



左圖為上面那張圖放大之後的 attention，可以看到有某些字 text 會貢獻給 summary 比較多所以 attention weight 會比較高。

Q6: Explain Rouge-L (1%)

首先要知道LCS(longest common subsequence)，每一個給定序列中最長的子序列，例如 s1: 2 5 7 9 3 1 2、s2: 3 5 3 2 8 ，LCS(s1, s2) 就是 5 3 2 長度為 3，

左下是 Rouge-L Recall、precision 公式，其中 m 是 reference sentence words count、n 是 prediction sentence words count、LCS(X,Y)則是兩個 sentence 的最長的子序列長度，再將左邊多出結果帶入右下 F1-score 公式得出 Rouge-L.

$$R_{lcs} = \frac{LCS(X,Y)}{m} \quad (2)$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \quad (3)$$

$$: 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Rouge-L 的優勢在於不需要連續匹配，而是按順序匹配。