ADL HW1 Report. R08944034 洪商荃

I. Data processing

A. 我在早鳥階段就先做了當時 extractive ,tokenize data 的方式是以先用 NLTK 將每段句子先切成 word 然後再轉成小寫字母最後再去查我預先 pre_train 的 glove 檔換成 300 維向量,後來與助教釋出的 sample_code 是做差不多的事情,所以後面兩個 model 都是用助教的 smaple code preprocessing 去處理。

B. 在 text summary 長度取捨部分 我在 extractive 設一個 text 字長上限是 175 是根據實驗的發覺太長我的 GPU 會 out of memory 因為我已經每個 word embedding 都用 300 維的去表示了,所以句子不能太長。然後發覺 175 效果是不好的,然後 predict 階段我只會選擇最高的去預測,所以我 extractive 他只會選擇 model 覺得最重要的那句去 predict。然後 s2s 和 s2s with attention 是做差不多的所以統一講,因為都是使用助教的 sample_code 然後有去研究 發覺助教是設定 text 長度最長是 300 summary 是 80 然後如果裡面沒有超過這個上限就以那個 batch 的最長字串去當上限。

C. 我使用的是"glove_6B_300d.txt"發覺他好像跟他的檔名不太符合我原本以為有 60 億的 word 可是大概只有 40 萬,然後我在 extractive 會把它全部抓進來存成 dict 的格式,可是後面 s2s 和 attention 是使用助教的 sample code 發覺助教只有去存取 train test valid 有出現的字 這樣子優點是比較不會佔太多空間,不過預測不出來那三個 txt 沒出現的字就是了,不過使用助教的 sample code 還是有 11 萬個字所以應該還是非常足夠。

II. Describe your extractive summarization model(extractive)

A. 我使用$ouptut = LSTM(input_size = 300, hidden_size = 50, num_layers = 3, bidirectional = True)$因為發覺模型一開始用很複雜可是效果也沒有比簡單的好,所以後來模型都盡量簡單,原本想使用 dropout 可是我在 train 的表現跟 valid 不會差很多,所以應該沒有很 over_fitting.

B.
```
{
    "mean": {
        "rouge-1": 0.19596604576711202,
        "rouge-2": 0.029985425957960308,
        "rouge-l": 0.14336565969819123
    },
    "std": {
        "rouge-1": 0.09204915800201606,
        "rouge-2": 0.05032635412206232,
        "rouge-l": 0.07066782037919016
    }
```
一開始在測分數就很接近 0.17 多了可是我當時不知道助教給的分數是要乘以 100 的所以我其實

滿早就過 baeline 了 發覺助教 bseline 在前面兩個 model 都很好過!覺得很佛心~

C. Loss_function 使用助教投影片所說的 nn.BCEWithLogitsLoss()
D. optimization algorithm 看目前好像深度學習都是用 Adam 達到不錯的效果於是就用了 然後 learing_rate 用 0.001。
E. 因為每個字都會算出一個機率,我就算那句話的平均機率(全部字的機率累加/一句話幾個字)然後選擇句子機率最高的那句當成預測,而不是選擇有沒有>0.5 來預測因為我發覺我的 model 不太會預測>0.5 的機率會變成可能一個 text 選不出最佳的來當 summary。

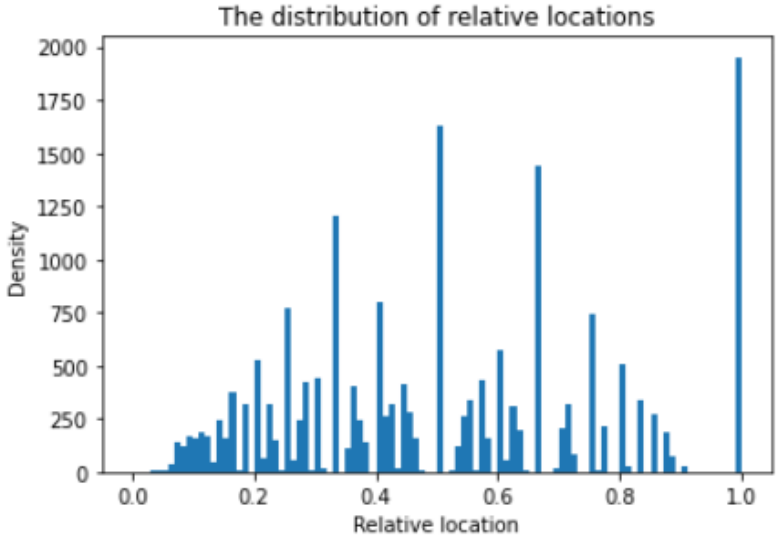
III. Describe your extractive summarization model(s2s with attention)

A. 我在 s2s 和 attention 都是使用 GRU 發覺效果不會比較差,但訓練時間明顯少很多。因為這邊有使用兩個模型分別為"encoder" 和 "decoder" Enocder 的部分會先用一個$embed_vecotr = embedding(word)$然後再接一個$output\ hidden = GRU(input_size = 300, hidden_size = 150, num_layers = 2\ bidirectional = True)$ Decoder 部分我是參考 **pytorch tutorial** 實作的 跟助教投影片的架構有些許差別,s 各層連接順序如下$embedding -> Linear(input = embediing + prev_hidden) -> GRU -> Linear$。

B.
```
"mean": {
    "rouge-1": 0.20477049128111463,
    "rouge-2": 0.04322445433623938,
    "rouge-l": 0.1630414881881149
},
"std": {
    "rouge-1": 0.11101191958140662,
    "rouge-2": 0.06976706841687563,
    "rouge-l": 0.09482696966212457
}
```
效果有比單純 s2s 好滿多的可是距離助教的 baseline 還有一點我覺得是我 model 還沒去學到資訊因為我如果直接預測 train 的結果也只高 baseline 高一點而已,不過因為這個 train 還滿花時間的,所以來不及試更多東西,只好做多少先這樣子。

C. Loss_funtion 使用 cross_entropy 因為我們現在可以算是在做預測,預測每個 deocder 需要去選擇哪一個字來當它 output.
D. optimization algorithm 一樣用 Adam 達到不錯的效果於是就用了 然後 learing_rate 用 0.001。(有看到助教在討論區有透露他的 model 資訊可以發覺應該我的 loss_function 和 optimization algorithm 都沒問題)

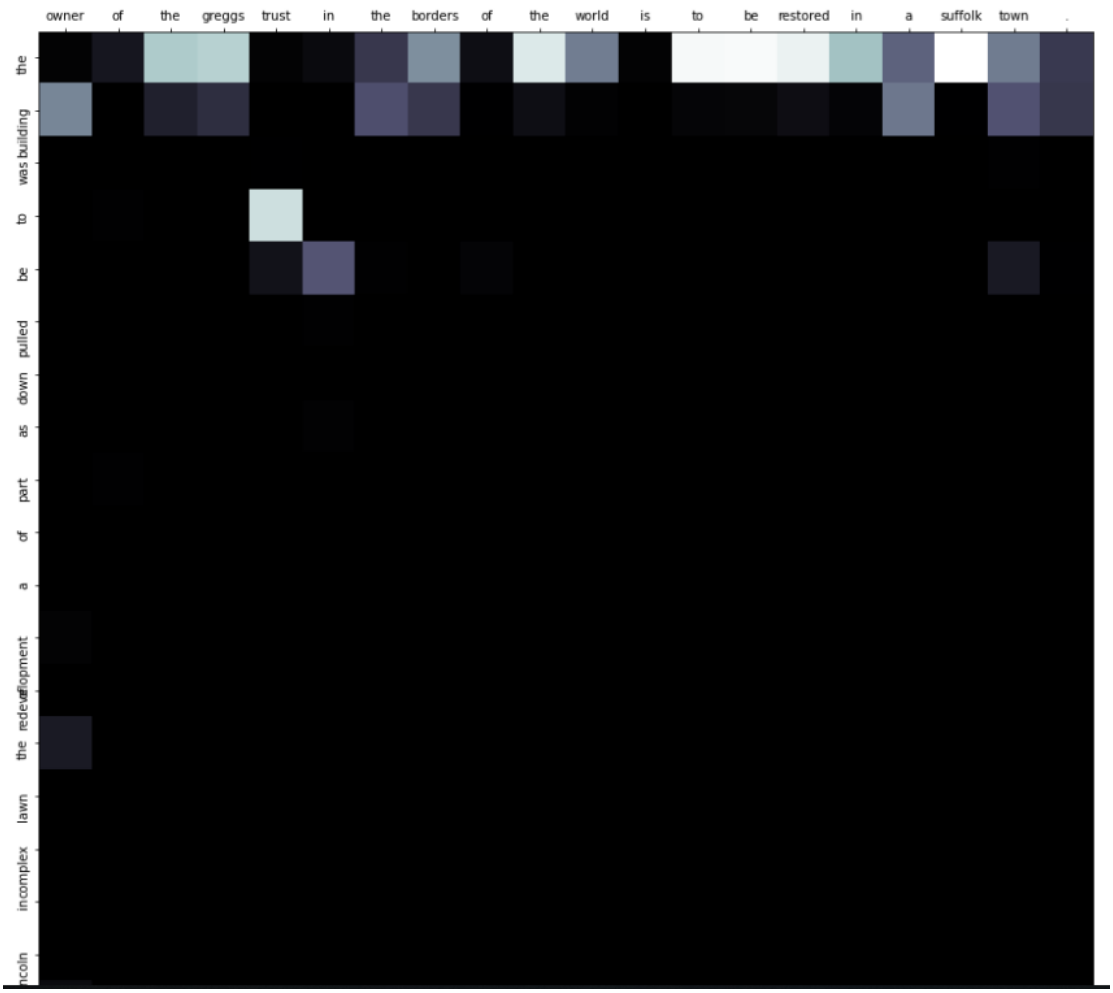
IV. Plot the distribution of relative locations



上圖是我根據 model 1 預測出來的結果畫出的直方圖 縱軸代表次數 因為有 2 萬筆資料(只去蒐集 valid 的資料)所以累積是 2 萬。

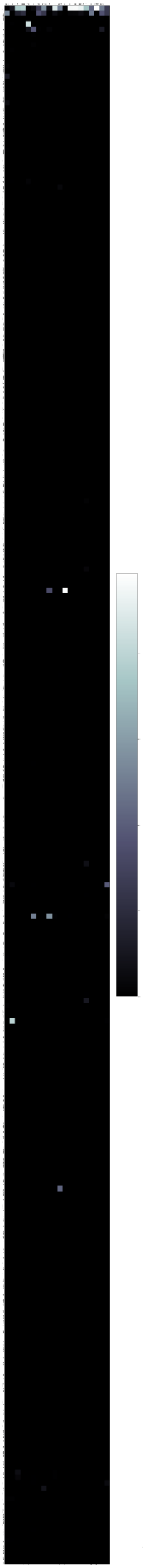
要從分布來討論什麼的話.可以發覺我的模型不會刻意去預測偏向哪一邊的, 因為如果 overfitting 的話感覺如果按照文章架構感覺猜中間是最重要的那句是有可能的,所以可以由此觀察我的 model 應該不太 overfitting。

V. Visualize the attention weights



這邊無法擷取完整的圖 因為縱軸有 297 個字(encoder_ouput size = text 長度)所以會變得太大,不過我由這個圖就可以說明我目前模型的問題了,模型遇到了有點 overfitting 的問題,我在猜可能是因為訓練時因為 train_data 他很常 summary 出現前面的 words,所以我 attention 的係數可能就前面幾個很大,導致我模型在 valid 一直效果不好,可是在 train 的方面表現不錯,可能要再多做 drop_out 部分 或者 fine_tune。

完整圖如下:



可以從完整圖大概的顏色分布,可以了解到我 attention 幾乎都只

著重在上半部,應該可以往這個方向來做調整。

VI. $R_{lcs} = \frac{LCS(X,Y)}{m}$ $P_{lcs} = \frac{LCS(X,Y)}{n}$ $F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}}$

其中$LCS(X,Y)$代表兩個字串 X,Y 最長公共子序列長度,m,n 分別代表兩個 text 所含的 word 數量,而上述的$\boldsymbol{F_{lcs}}$**就是 Rouge-L**。而β為一個很大的數。