工科碩一 R08525060 吳祐鴻

Q1. Describe how do you use the data for extractive.sh, seq2seq.sh, attention.sh:

a. How do you tokenize the data.

extractive.sh:

將資料集中的每個句子的標點符號過濾，再依空格分割成單字。建立一個字典將資料集的所有單字放入字典裡，之後將每個句子裡的單字轉成整數，每個數字都是字典裡的索引值。

seq2seq.sh, attention.sh:

將 train.jsonl 的 text 和 summary 的所有單字集合成字典，留下頻率最高的 2000 個單字，再分別將 text 和 summary 裡所有的單字轉為整數，每個數字都是字典裡的索引值。

b. Truncation length of the text and the summary.

extractive.sh:

最大句字長度設為 327 個單字。

seq2seq.sh, attention.sh:

text 最大長度 300

summary 最大長度 40

c.The pre-trained embedding you used.

三個模型都是用 GLOVE

Q2: Describe your extractive summarization model.

Describe

a. model :

令輸入文章為 X 其中的第 t 個單字為 X_t，經過 embeding layer 後每個單字 X_t 降維成 300 維的向量得到輸出 W_t。表示如下:

$$W_t=\text{Embedding}(X_t,300)$$

之後將轉換後的文章 W 輸入一個雙向 biLSTM(W,h,c)，其中 h,c 為 LSTM 的 hidden state，

$$\text{正向輸入 } W_t,h_t,C_t = \text{LSTM}(W_{t-1},h_{t-1},C_{t-1})$$

$$\text{反向輸入 } \text{Revese\_}W_t\ \text{Revese\_}h_t,\ \text{Revese\_}C_t = \text{LSTM}(W_{t-1},h_{t-1},C_{t-1})$$

把每一個 W_t ,$\text{Revese\_}W_t$ 串接起來成 Y 再過 linear(dense)層降維成 600 維:

$$Y = \text{Concatenate}\ (W_t\ ,\text{Revese\_}W_t)$$

$$Y = \text{dense}(Y,\ 600)$$

其中 dense(x,y)為全連接層 x 為輸入 data，y 為輸出維度。最後經過 3 層全連接層，透過 sigmoid 層輸出。

$$Y = \text{dense}(Y,\ 100)$$

$$Y = \text{dense}(Y,50)$$

$$Y = \text{dense}(Y,50)$$

Result= sigmoid(Y,1)

b. performance of your model.(on the validation set)


```
{
    "mean": {
        "rouge-1": 0.18563166985601928,
        "rouge-2": 0.03133856370128954,
        "rouge-l": 0.12417066324423062
    },
    "std": {
        "rouge-1": 0.07395020620395142,
        "rouge-2": 0.0386704749027753,
        "rouge-l": 0.05250110134298402
    }
}
```


c. the loss function you used.

因為 label 的資料裡的 1 和 0 的比例為 1:13 是一個 imbalance 問題，所以使用 tensorflow 裡的 weighted_cross_entropy_with_logits，公式如下。其中y_n為 label data，$\hat{y}_n$為模型預測的類別，pos_weight 為權重。由於資料裡的 1 和 0 的比例為 1:13 故將 pos_weight 設為 13。

$$Loss = \frac{-1}{N}\sum_{n=1}^{N}[y_n log(sigmoid(\hat{y}_n) * pos\_wight + (1 - y_n) log(1 - (sigmoid(\hat{y}_n))]$$

d. The optimization algorithm (e.g. Adam), learning rate and batch size.

Adam. learning rate=0.001, batch size=256

e. Post-processing strategy.

句子裡的每個 token(word)，都預測為 1 時才挑那句。

Q3: Describe your Seq2Seq + Attention model. (2%)

a. model description:

Encoder:

將資料輸入 Embeding layer 和一層 Bidirectional Lstm ，Lstm 有三個輸出分別為 encoder output 和 c state , h state。

Decoder:

將 encoder 的輸出的 c state, h state，經過一層 LSTM，令輸出為 decoder output。令輸入文字 X 其中的第 t 個單字為 X_t

$$\text{decoder output} = \text{LSTM}(X_{t-1}, h_{t-1}, C_{t-1})$$

Attention:

令 attention 中的 query=decoder output，key=value=encoder output。使用 dot product attentio 實作如下:

$$\text{score} = \text{dot}(\text{key}, \text{query})$$

Attention weight =sigmoid(score)

context_vector=Dot(Attention weight,value)

最後將 context_vector 和 decoder output 串接起來，經過 softmax 得到答案。(討論:陳仲軒 R08525056)

b.performance of your model.(on the validation set)


```
{
    "mean": {
        "rouge-1": 0.2547332217423299,
        "rouge-2": 0.07127595955240999,
        "rouge-l": 0.21061360305163015
        },
    "std": {
        "rouge-1": 0.12766167677429513,
        "rouge-2": 0.09921275936483091,
        "rouge-l": 0.11903646076440517
      }
}}
```

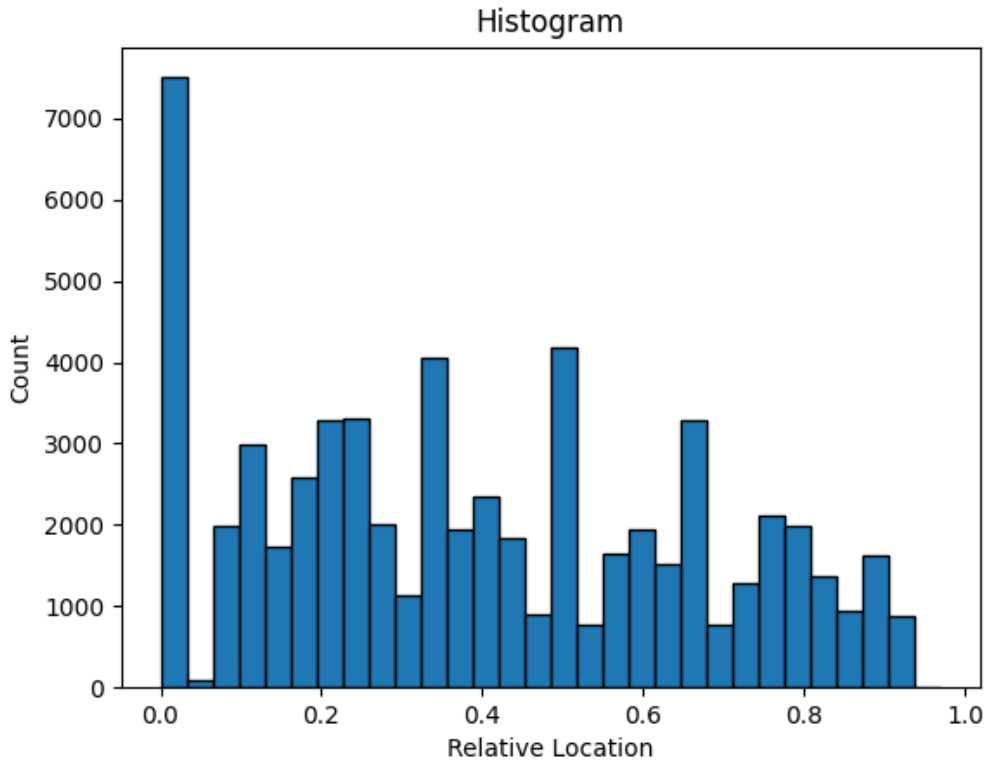

d.the loss function you used.

Loss function: sparse_categorical_crossentropy

Adam. learning rate=0.001, batch size=256

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

從圖中可知，挑第一句(relative location=0)的比例最高，總共出現 7000 次，挑中間句(relative location=0.5)和 relative location=0.3 的比例為次高，大約有 4000 次。其它句的比例會隨著隨著相對位置上升而下降，可知此模型傾向挑一的文章裡前半段的句子。。



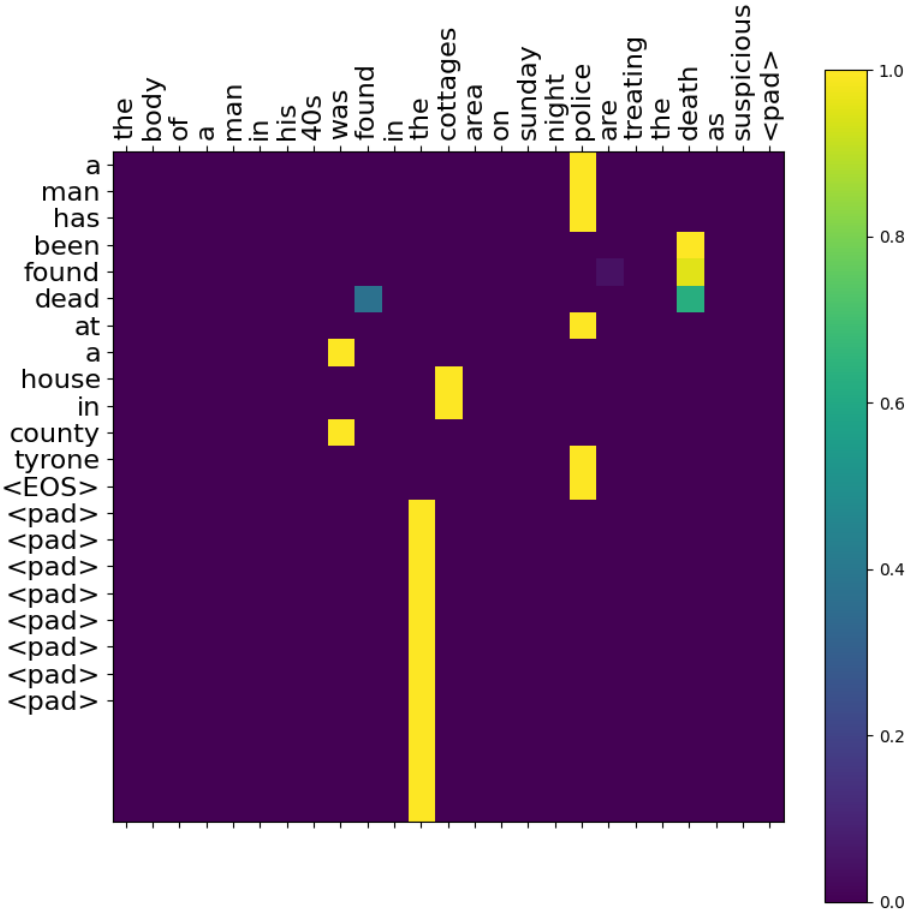
Q5: Visualize the attention weights (2%).

圖片說明:

x 軸為 text

y 軸為 summary:<BOS>代表句子的開頭、<EOS>代表句子的結尾、<pad>代表填補零讓每句的長度都一樣。

下圖為 valid set 中的其中一句，”the body of a man in his 40s was found in the the cottages area on sunday night police are treating the death as suspicious”，生成摘要”a man has been found dead at a house in country tyrone”。其中可以看到當預測 man 時，police 的 weight 較大，而當預測 house 時 cottages 的 weight 較大。可以推測當模型看到看到一個名詞，會預測出相近字義的字，如看到 police 生成 man、看到 cottages 生成 house。



Q6: Explain Rouge-L (1%)

Rouge-L 中的 L 代表 Longest common sequence(LCS,最長共同子序列)，將人工寫成的參考句子 X 長度為 m 和機器生成的句子 Y 長度為 n 和找出共同子序列，如下式:

$$R_{LCS} = \frac{\mathrm{LCS(X,Y)}}{\mathrm{m}}$$

$$P_{LCS} = \frac{\mathrm{LCS(X,Y)}}{\mathrm{n}}$$

$$F_{LCS} = \frac{(1+\beta^2)R_{LCS}P_{LCS}}{R_{LCS}+\beta^2 P_{LCS}}$$

其中 LCS(X,Y)是 X,Y 的 LCS 的長度，R_{LCS}為 Recall，P_{LCS}為 Pecision，F_{LCS}代表 F measure ，β=P_{LCS}/R_{LCS}，如果為 F1 score，則設 $\beta = 1$