Applied Deep Learning HW1

b05502087 資工系 王竑睿

1 Q1: Data processing (1%)

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

a. How do you tokenize the data.

- 使用 en_core_web_sm 作為語言模型，宣告出切割函數 nlp
- 利用 nlp 將 sentences 字串切割成 token 形成的 list 並且全部轉為小寫
- 再將 token 形成的 list encode 成數字(word index) 形成的 list

b. Truncation length of the text and the summary.

- text 中，每筆 sample 出來的 word index list 以 300 作為 truncation length
  - 超過 300 個 word index 則只取 300 個
  - 少於 300 個 則會 pad 到 300
- summary 中，每筆 sample 出來的 word index list 以 74 作為 truncation length
  - 超過 74 個 word index 則只取 74 個
  - 少於 74 個 則會 pad 到 74

c. The pre-trained embedding you used.

- 本次作業我使用的 pre-trained embedding 是 stanford 的 glove.840B.300d
  - 840 billion tokens, 2.2 million vocab, cased, 300-dimension vectors
  - `https://github.com/stanfordnlp/GloVe`

2 Q2: Describe your extractive summarization model. (2%)

1. Describe

a. your model

```
LSTMTagger(
  (word_embeddings): Embedding(96808, 300)
  (lstm1): LSTM(300, 100, batch_first=True, bidirectional=True)
  (lstm2): LSTM(200, 60, batch_first=True, bidirectional=True)
  (lstm3): LSTM(120, 30, batch_first=True, bidirectional=True)
  (lstm4): LSTM(60, 10, batch_first=True, bidirectional=True)
  (lstm5): LSTM(20, 5, batch_first=True, bidirectional=True)
  (hidden2tag): Linear(in_features=10, out_features=1, bias=True)
)
```

$$
\begin{aligned}
w_t &= Embedding(token_t) \\
h1_t, o1_t &= LSTM1(w_t, h1_{t-1}) \\
h2_t, o2_t &= LSTM2(o1_t, h2_{t-1}) \\
h3_t, o3_t &= LSTM3(o2_t, h3_{t-1}) \\
h4_t, o4_t &= LSTM4(o3_t, h4_{t-1}) \\
h5_t, o5_t &= LSTM5(o4_t, h5_{t-1}) \\
tagScore &= Linear(o5_t)
\end{aligned}
$$

- w_t 為第 t 個 token 經過 embedding 所形成的詞向量
- hi_{t-1} 為第i層 LSTM 在輸入第 (t-1) 個詞向量後的 hidden state
- hi_t 為第i層 LSTM 輸入第 t 個詞向量後的 hidden state, 將在下一次輸入詞向量時作為 hidden state 一起輸入
- oi_t 為第i層 LSTM 輸入第 t 個詞向量後的 output feature，會作為第 (i+1) 層 LSTM 的輸入
- 第五層 LSTM 的輸出 feature 再利用 Linear 轉換成 tag Score 來決定選或不選

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


```
{
    "mean": {
        "rouge-1": 0.18955216763988175,
        "rouge-2": 0.031405092479742,
        "rouge-l": 0.12895564482326544
    },
    "std": {
        "rouge-1": 0.07763919100062465,
        "rouge-2": 0.04020595557637563,
        "rouge-l": 0.055554785922348054
    }
}
```


c. the loss function you used.

- 本次作業進行 extractive summary 使用的 loss function 是 BCEWithLogitsLoss

$$
\begin{aligned}
BCELoss &= -\frac{1}{N}\sum(y_n \times \ln x_n + (1-y_n) \times \ln(1-x_n)) \\
BCEWithLogitsLoss &= -\frac{1}{N}\sum(y_n \times \ln sigmoid(x_n) + (1-y_n) \times \ln(1-sigmoid(x_n)))
\end{aligned}
$$

- 其中，x_n 為 predict 出來的值 (tagScore), y_n 為 target 的值
- BATCH SIZE 為 N

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

- 本次作業進行 extractive summary 使用的是 Adam optimizer
- learning rate 設為 0.001
- batch size 為 32

e. Post-processing strategy.

- 一個 sentence 中會有多個 token, embed 成詞向量並經過 model 後會預測出 0(不選) 或 1(選)
- 在一筆 training data 中有多個 sentence。取預測出 1 的 token 數佔總 token 數比例最高的句子作為此筆 training data 的 extractive summary
- 若有多個句子比例相同，則皆取出

$$summary = \underset{s \in \text{training data}}{\arg\max} \left(\frac{num(\text{tokens predicted as 1 in s})}{num(\text{tokens in s})}\right)$$

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

1. Describe

a. your model

```
AttnEncoderRNN(
  (embedding): Embedding(97513, 300)
  (gru): GRU(300, 300, bidirectional=True)
)
AttnDecoderRNN(
  (embedding): Embedding(97513, 300)
  (dropout): Dropout(p=0.1, inplace=False)
  (gru): GRU(300, 300, bidirectional=True)
  (out): Linear(in_features=1200, out_features=97513, bias=True)
)
```

Encoder

$$w_t = Embedding(token_t)$$
$$Eh_t, Eo_t = GRU(w_t, h_{t-1})$$

- w_t 為第 t 個 token 經過 embedding 所形成的詞向量
- Eh_t, Eo_t 為 w_t 經過 GRU 之後，GRU 的 hidden state 以及 output feature

Decoder

$$\begin{aligned} w_t &= Embedding(token_t) \\ wd_t &= Dropout(w_t) \\ wRelu_t &= relu(wd_t) \\ Dh_t, TEMPo_t &= GRU(wRelu_t, h_{t-1}) \\ similarity &= Cosine(Eo_t, TEMPo_t) \\ attentionWeights &= softmax(similarity) \\ attentionApplied &= attentionWeights * Eo_t \\ outputFeature &= cat(TEMPo_t, attentionApplied) \\ Do_t &= logSoftmax(linear(outputFeature)) \end{aligned}$$

- w_t 為第 t 個 token 經過 embedding 所形成的詞向量
- $wRelu_t$ 為 w_t 經過 dropout 以及 relu 函數之後的向量
- $Dh_t, TEMPo_t$ 為 w_t 經過 GRU 後，GRU 的 hidden state 以及輸出 feature
- 本次作業的 attention 採用 cosine similarity 來進行 match
  - 利用 $TEMPo_t$ (query), 與 encoder 的所有 output Eo_t (key), 計算 cosine similarity
  - 藉由向量間的相似程度來決定 attention 的權重 attentionWeights
  - attentionApplied 為所有 encoder outputs 經過 attentionWeights 加權得到的向量
  - 最後將 attentionApplied 與 decoder GRU 的輸出連接，並經過 linear 層以及 logSoftmax 輸出
- 對於每一個token詞向量，此輸出會是一個和 vocab 長度相同的向量
  - 意義為此 token 屬於某個字的機率取 logSoftmax

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


```
{
    "mean": {
        "rouge-1": 0.25269693742562316,
        "rouge-2": 0.06957193876739216,
        "rouge-l": 0.20682882235755065
    },
    "std": {
        "rouge-1": 0.12450755249907464,
        "rouge-2": 0.09502027948159368,
        "rouge-l": 0.114587071671803
    }
}
```


c. the loss function you used.

- 本次作業進行 attention summary 使用的 loss function 為 NLLLoss
- NLLLoss 將預測出的結果的 target 那項取出，去除負號作平均

$$NLLLoss = \frac{1}{N}\sum(-x_n[target_n])$$

- 其中，x_n 為 predict 出來的值, $target_n$ 為 target 的值
- BATCH SIZE 為 N

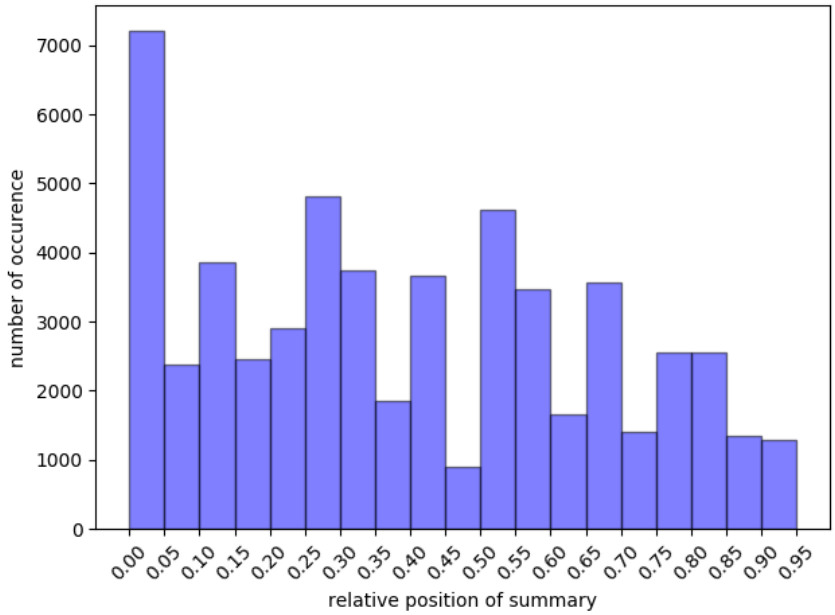
d. The optimization algorithm (e.g. Adam), learning rate and batch size.

- 本次作業進行 Seq2Seq+Attention summary 使用的是 Adam optimizer
- learning rate 設為 0.001
- batch size 為 32

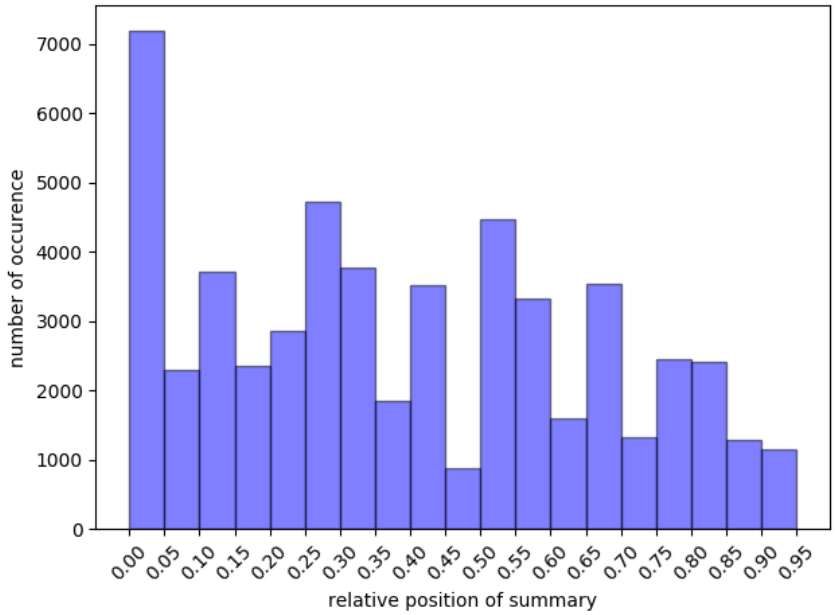
4 Q4: Plot the distribution of relative locations (1%)

Plot the distribution of relative locations of your predicted sentences by your extractive model, and describe your findings. (1%)

- validation data 中, 預測句子在 data 中的相對位置



- test data 中, 預測句子在 data 中的相對位置



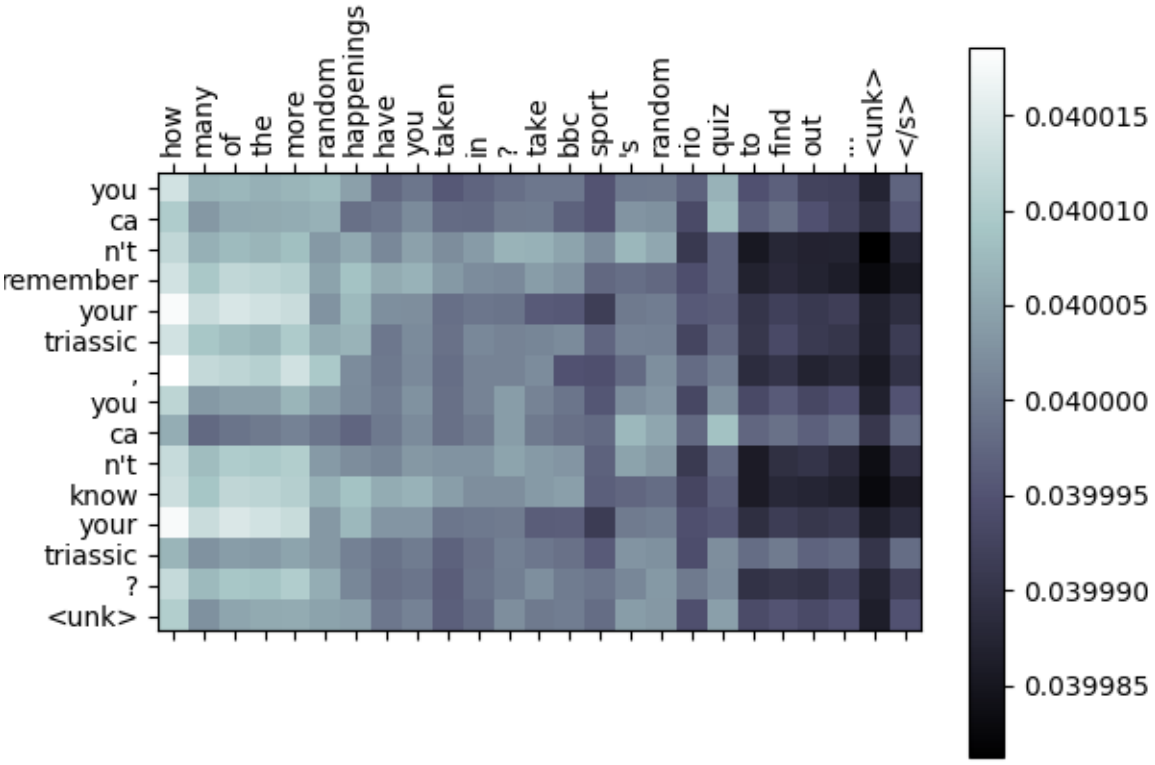
- 發現
  - 在 validation data 和 test data 中預測出來的相對位置分布幾乎相同
  - 預測出來的句子之相對位置在一開始(0-0.05)處佔據多數

5 Q5: Visualize the attention weights (2%).

1. Take one example in the validation set and visualize the attention weights (after softmax)

a. Readable text on the two axises. (0.5%)

b. Colors that indicate the value. (0.5%)



2. Describe your findings. (1%)

- 本次 attention 的實作並不算成功，因為最高 weight 與最低 weight 幾乎相差無幾

  - 我推測是因為在實作 attention 時未將 padding mask 掉,所以 model 的注意力從句子開始到結束逐步遞減

- 但在 <unk> 等較無意義的字元上，attention weight 相對有偏低

6 Q6: Explain Rouge-L (1%)

Explain the way Rouge-L is calculated.

- 令從 model 文檔中 summary 出來的內容為 X 且長度為 n, ground truth 的 summary 內容為 Y 且長度為 m
- 我們可以利用 Dynamic programming 求出 X 和 Y 的最長公共子序列 LCS(X,Y)
- Recall R 為 $\frac{LCS(X,Y)}{m}$

- Precision P 為 $\frac{LCS(X,Y)}{n}$
- Rouge-L 即為 $\frac{(1+\beta^2)\times R\times P}{R+\beta^2 P}$
  - 其中 β 為一個係數，可決定 Recall 和 Precision 兩者的重要性