r08944035 呂翊愷

ADL Homework #1

Q1: Data processing (1%)

Tokenize的部分，extractive, seq2seq, attention都是使用相同的方式。首先使用spacy套件將train, valid, test資料讀取出來，並切成一個一個token，而我所使用的pre-trained embedding檔是glove.840B.300d，如果該字有在pre-trained embedding中的話，就將該字加進embedding vector中，有自己的index；如果該字沒有在pre-trained embedding中的話，就將該字視為unknown words，將該字標為embedding vector中的<unk>。此外，每個句子的前後，加入<s>和</s>標記。為了讓每筆資料的長度相同，我設定每筆資料為300個字。若原始資料的長度不足，將剩餘的長度以<pad>補足；若原始資料長度太長，將多餘的字捨棄。針對<pad>, <s>, </s>, <unk>這些特殊符號，也在embedding vector中有自己的index，分別是0, 1, 2, 3。

針對extractive的summary，將ground truth的所有token的label標成1，不是ground truth的token的label則標成0。由於我將每筆資料的token數設為300個，若原始資料的長度不足，將剩餘的label全部標為-100，方便日後在算loss的時候能將這些padding的部分忽略掉。

seq2seq和attention的summary，設定每筆資料最多長度為80，多餘的部分將被去除掉，每個label都是該token的embedding vector的index，若為padding的部分，則將label標為0，方便日後在算loss的時候能將這些padding的部分忽略掉。

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

Model結構

- $w_t = embedding(idx_t)$，idx_t是第t個token的index
- $out_t, h_t, c_t = LSTM(w_t)$，$w_t$是第t個token的word embedding vector
- $l_t = linear(out_t, 1)$，out_t是LSTM的output，經過linear後維度從hidden layer * 2變成1

將句子中的每個token index傳進model，model透過embedding會利用index轉成對應的embedding vector，再將vector傳進LSTM，LSTM的output再用linear變成1維的vector。

Performance


```
{
    "mean": {
        "rouge-1": 0.1985423582131631,
        "rouge-2": 0.030352856594727463,
        "rouge-l": 0.14546640595128338
    },
    "std": {
        "rouge-1": 0.09221944460449706,
        "rouge-2": 0.050690563507859686,
        "rouge-l": 0.07077562216228946
    }
}
```

Loss function
使用BCEWithLogitsLoss，參數reduction=none, pos_weight=7，避免label分佈不平均的問題。在將predict和target餵進loss function之前，我還有用mask select將padding的部分去掉，如此一來padding的部分將不會影響到loss的值。

Optimizer
使用Adam，learning rate設為0.001，batch size設為16。

Post-processing
將model預測出來的output再經過sigmoid處理後，此時將每個句子的token的output加總起來並除以該句子的token數，得出該句子整體的平均機率，再選出該筆資料中機率最高的句子當作最後預測的結果。

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

Model結構

Encoder:
- $w_t = embedding(idx_t)$，idx_t是第t個token的index
- $out_t, h_t, c_t = LSTM(w_t)$，$w_t$是第t個token的word embedding vector
- $l_t = linear(h_t)$，h_t是第t個token embedding vector經過LSTM後的雙向的hidden layer，經過linear後維度從hidden layer * 2變成hidden layer
- $hid_t = tanh(l_t)$，l_t是第t個hidden layer經過linear後的vector

Decoder:
- $w_t = embedding(idx_t)$，idx_t是第t個token的index
- $out_t, h_t, c_t = LSTM(w_t)$，$w_t$是第t個token的word embedding vector
- $l_t = linear(out_t)$，out_t是第t個token embedding vector經過LSTM後的output

Attention:
- Key = Encoder的out_t
- Query = Decoder的out_t
- Attention weight = softmax(Key * Query)
- Context vector = Attention weight * Key
- $predict = linear(concatenate(context, out_t))$，$context$是context vector，$out_t$是decoder的LSTM的output

將句子中的每個token index傳進encoder，encoder透過embedding會利用index轉成對應的embedding vector，再將vector傳進LSTM，LSTM的hidden layer和cell再用linear將維度從hidden layer * 2變成hidden layer。

Decoder的LSTM的初始hidden和cell是encoder最後產生出的hidden和cell，再將Key, Query, Attention weight, Context vector算出來後，和Decoder的LSTM的output連接在一起，經過linear後變成9萬多維，9萬多維是embedding vector的總數。Linear後再經過softmax，選出最大的token當作預測的結果。

Performance

Seq2Seq


```
{
    "mean": {
        "rouge-1": 0.19355755088872828,
        "rouge-2": 0.03857256129500186,
        "rouge-l": 0.1603494494708174
    },
    "std": {
        "rouge-1": 0.105476024076513,
        "rouge-2": 0.060724726178081896,
        "rouge-l": 0.09321336698398165
    }
}
```


Attention


```
{
    "mean": {
        "rouge-1": 0.25427142428407284,
        "rouge-2": 0.07167133807836375,
        "rouge-l": 0.2064440643734931
    },
    "std": {
        "rouge-1": 0.1230634344465886,
        "rouge-2": 0.0950147884324542,
        "rouge-l": 0.11352580540752097
    }
}
```


Loss function

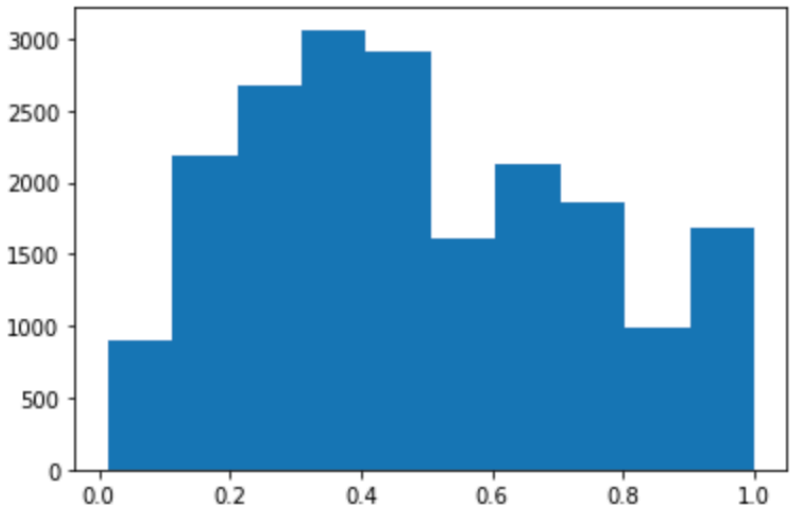
使用CrossEntropyLoss，參數ignore_index=0，此參數的用意是讓padding的部分不要被計算到，如此一來padding的部分將不會影響到loss的值。

Optimizer

Encoder和Decoder都是使用Adam，learning rate設為0.001，batch size設為16。

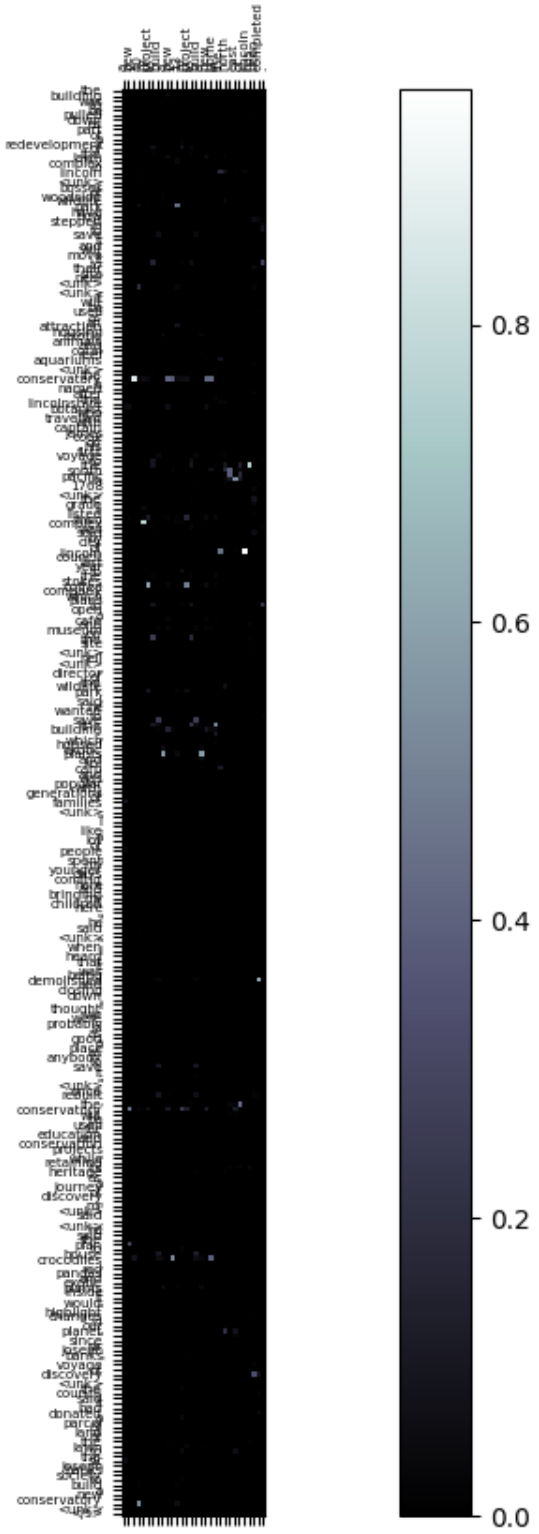
Q4: Plot the distribution of relative locations (1%)

透過這張圖可以發現，我的預測比較常是在比較前面的句子，會有這樣子的情況我推測可能是因為ground truth可能也比較常在前面，所以會造成model認為前面的句子機率比較大。



Q5: Visualize the attention weights (2%)

透過這張圖可以發現，Attention看的地方蠻分散的，沒有規律性可言，看的順序也和原文的前後順序無關，我認為可能是一個好的summary是需要綜合整體的文意去產生出來的，因此model在預測的時候是看的是比較全面的。



Q6: Explain Rouge-L (1%)

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

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

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

F_{lcs}即為Rouge-L，其中$LCS(X, Y)$是X和Y的LCS長度，m和n分別是X和Y的長度，而β是一個很大的數，因此F_{lcs}幾乎只考慮R_{lcs}。