ADL hw1 report

B05705017 資管四 黃亮穎

Q1: Data processing

Using sample preprocessing code:

1. Tokenize the data :
   a. Documents are text sentences in all of train, validation, test data plus summary sentences.
   b. Use spacy's en_core_web_sm to tokenize sentences.
   c. number of tokens in total are 96808 and 97513 in extractive summary and abstractive summary respectively.
2. Truncation:
   max length of text is set to 300 while max length of summary is set to 80.
3. pre-trained embedding : glove 840B 300d

Q2: extractive summarization

1. **model :**

```
LSTM(
  (embedding): Embedding(96808, 300)
  (dropout): Dropout(p=0.2, inplace=False)
  (lstm): LSTM(300, 128, num_layers=2, bidirectional=True)
  (fc): Linear(in_features=256, out_features=1, bias=True)
)
```

description:
Two layer, bi-directional LSTM with hidden dimension = 128 and dropout 0.2.

$$output = fc(O_1, O_2, ..., O_t, ...O_T),\ where\ O_t\ is\ LSTM's\ output\ in\ time\ stamp\ t$$

stacks output in every time stamp and then pass through a linear layer to reduce

dimension to 1. This one dimension tensor will then be use in binary cross entropy with log softmax loss function to calculate loss.

2. **performance : validation rouge score**


```
{
    "mean": {
        "rouge-1": 0.18960789894719451,
        "rouge-2": 0.028931178865085144,
        "rouge-l": 0.12987530388902072
    },
    "std": {
        "rouge-1": 0.07881957951072957,
        "rouge-2": 0.04079902990983379,
        "rouge-l": 0.055646249139425064
    }
}
```


3. **optimization algorithm** : AdamW
4. **loss function** : binary cross entropy loss w/ log softmax
   learning rate : 1e-3
   batch size : 16
   epoch : 5
5. **training time per epoch** : about 1 minutes
6. **post-processing strategy** :
   a. for each sentence in text, count tokens labeled 1 (selected), then divide by number of tokens in that sentence to acquire ratio of the selected tokens and total tokens in the sentence.
   b. Based on experiment, choose top 2 sentences with highest ratio (tokens in them were selected many more than other sentences)
   c. sort 2 sentences' index.

Q3: Seq2Seq+Attention

1. **model :**

```
Seq2Seq(
  (encoder): EncoderBiRnn(
    (embedding): Embedding(97513, 300, padding_idx=0)
    (dropout): Dropout(p=0.3, inplace=False)
    (gru): GRU(300, 300, bidirectional=True)
    (fc): Linear(in_features=600, out_features=300, bias=True)
  )
  (decoder): DecoderRnn(
    (attention): Attention(
      (attn): Linear(in_features=900, out_features=300, bias=True)
      (v): Linear(in_features=300, out_features=1, bias=False)
    )
    (embedding): Embedding(97513, 300, padding_idx=0)
    (dropout): Dropout(p=0.3, inplace=False)
    (gru): GRU(900, 300)
    (fc): Linear(in_features=900, out_features=97513, bias=True)
  )
)
```

description:

Encoder :

Based on the above model structure, encoder's GRU takes embedded inputs *([seq_len, batch_size, embed_size])* and then returns outputs *([seq_len, batch_size, hidden_size * 2])* as well as hidden *([1, batch_size, hidden_size]).*

$o = tanh(g(h_T^{\rightarrow}, h_T^{\leftarrow})) = s_0,\ h_T\ denoted\ encoders\ last\ hidden\ state$
Since it is one layer bidirectional GRU, the forward and backward hidden then be concatenated and pass through tanh activation function. The output will be decoder's first hidden state.

Attention:

Attention takes in the previous state of the decoder, s_{t-1} and all hidden states from the encoder, H to produce a vector a_t. Energy will be calculate as $E_t = tanh(attention(s_{t-1}, H))$. Then, it can simply multiply the energy by v *([1, hidden_size])* in order to put attention over all token.($\hat{a}_t = vE_t$). Finally apply softmax to ensure all numbers are between 0, 1. ($a_t = softmax(\hat{a}_t)$)

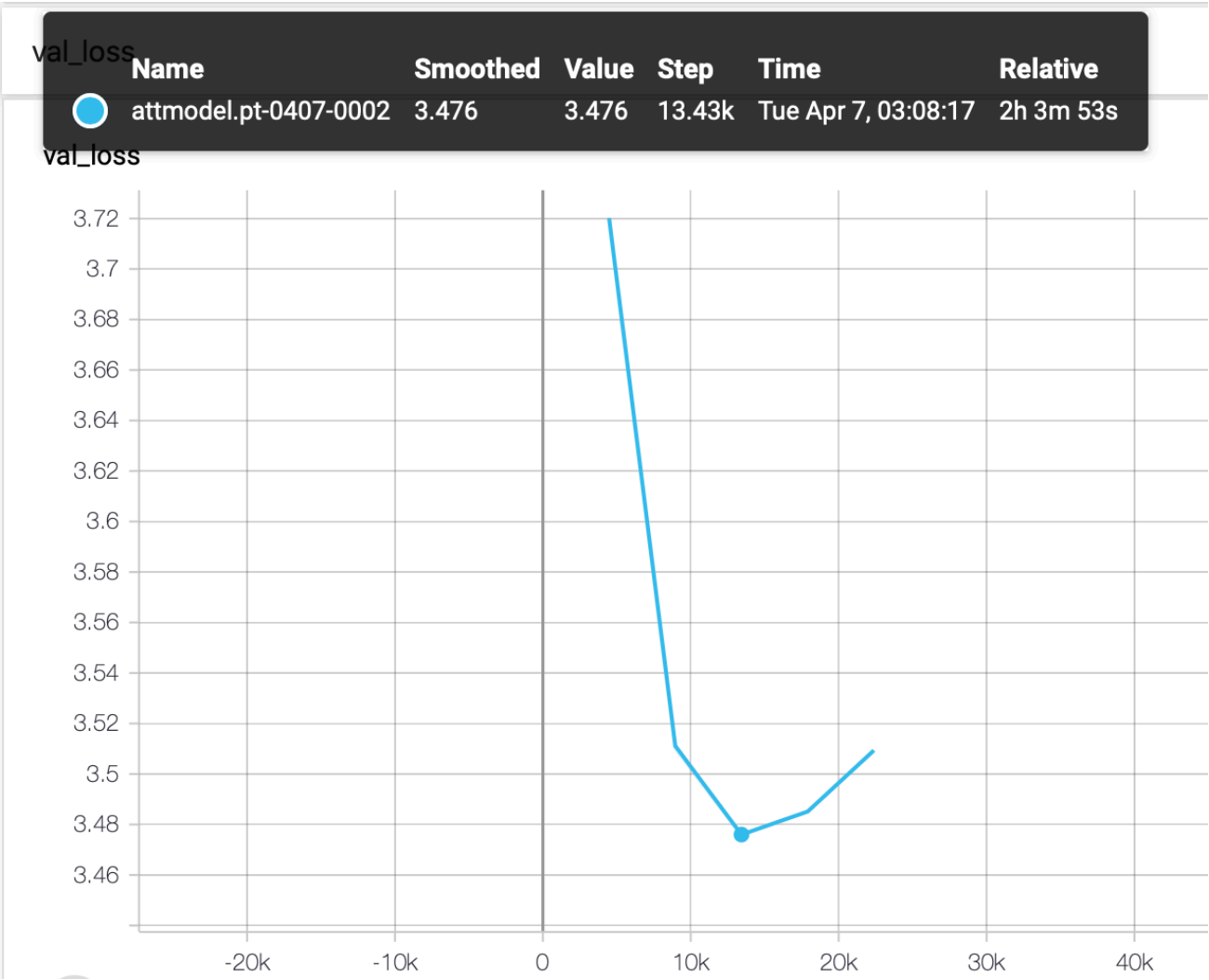
Decoder:

The weighted source vector $w_t = a_t H$, embed vector e_t and s_{t-1} are the inputs of decoder's GRU to returns new time stamp s_t. Then, both s_t and w_t are concatenated and pass through linear layer to produce output.

Seq2Seq:

Wrap up encoder, attention, decoder to be a sequence to sequence model with attention which structure is shown above.

2. **performance :**
   a. validation loss : best validation loss is 3.476 in the third epoch, however, best training loss is 2.635 in the last epoch, overfitting happened.



validation loss

   b. validation rouge score :

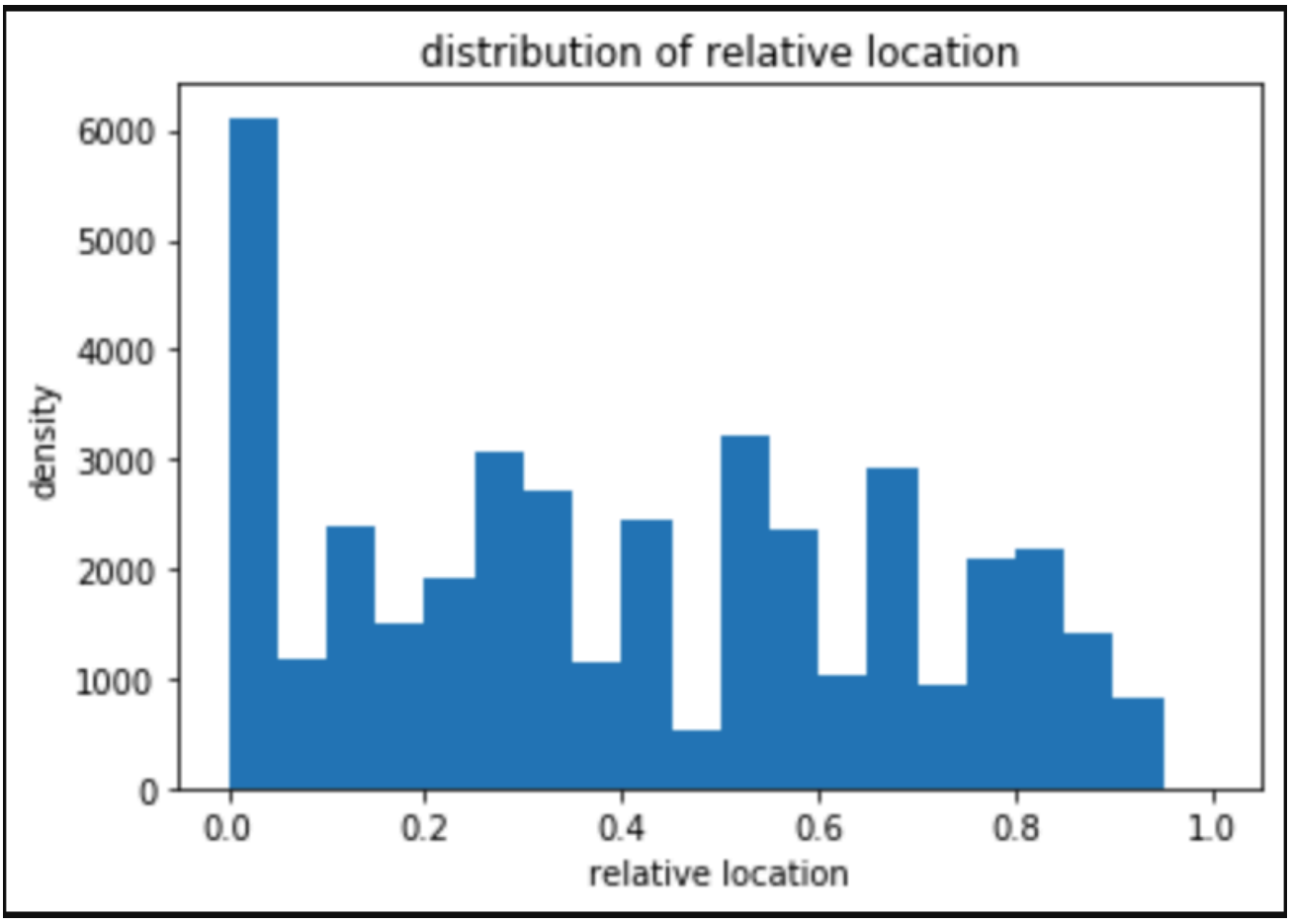
```
{
    "mean": {
        "rouge-1": 0.25492154442638587,
        "rouge-2": 0.07197218093261795,
        "rouge-l": 0.2101066360345319
    },
    "std": {
        "rouge-1": 0.1222585074987723,
        "rouge-2": 0.09409217399864296,
```

```
        "rouge-l": 0.11340053956866959
    }
}
```

3. **optimization algorithm** : AdamW
4. **loss function** : Cross Entropy Loss
   learning rate : 1e-3
   batch size : 16 (Due to limited VRAM (8G), I can not increase batch size anymore)
   epoch : 5
5. training time per epoch : about 60 minutes

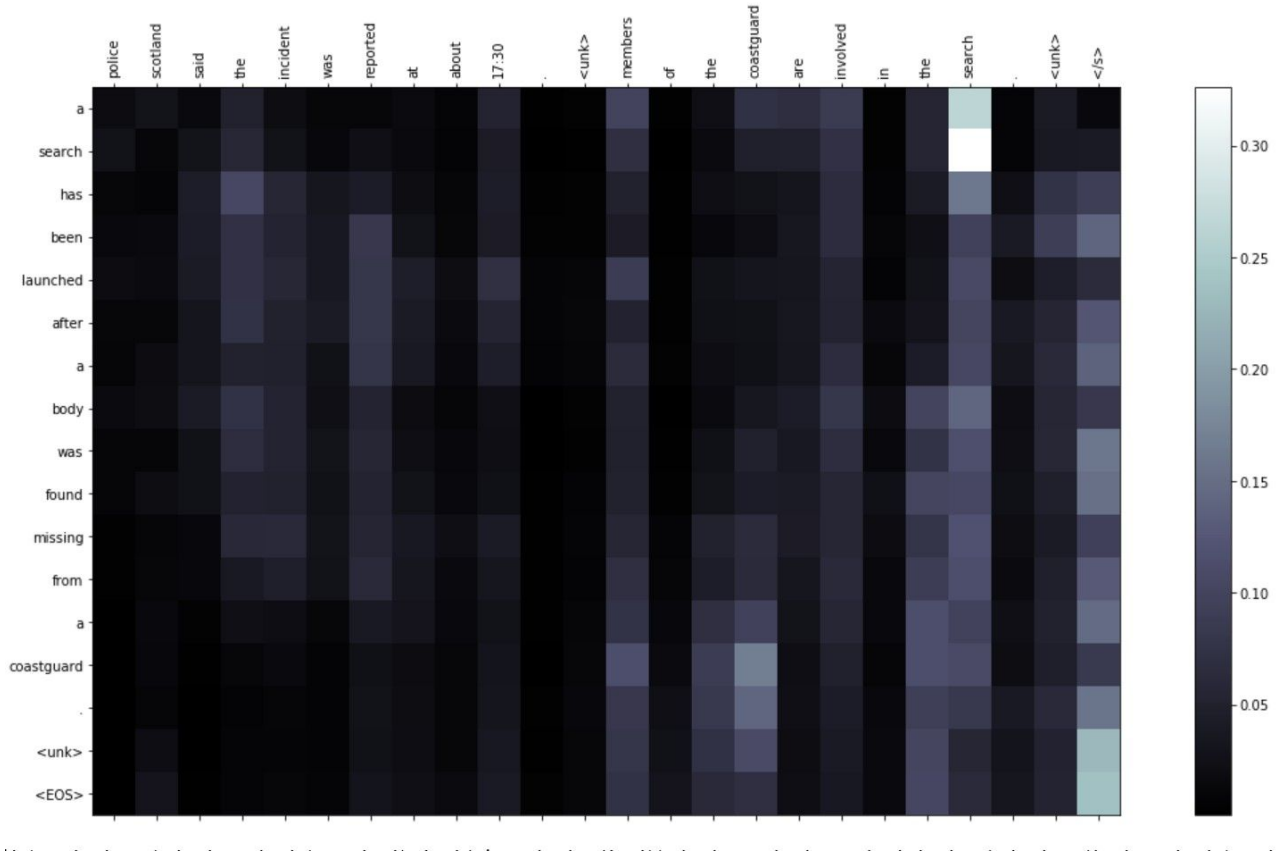
Q4: Plot the distribution of relative locations

I divide x axis into 20 bins. From the above histogram, there is no obvious pattern where extractive summary sentences are in my prediction, however, the most relative locations is 0, which means my model often use the first sentence to be as summary.



Q5: visualize the attention weights

Since I did not return attention weights in my training model and I did not have enough time to modify codes once I discovered the misery, I could not produce an attention weights graph by my own. Therefore, I borrowed a graph from my classmate to finish this problem.



A vertical bar in the right of the graph indicates scale of weights. The lighter the color is, the bigger the weight is. There is an obvious white block in top right of the graph, whose input token and output token are both “search”. It means that the attention weight of input token “search” is a lot bigger when decoder predict “search” as output. The same effect is also shown in the block (coast guard, coast guard). To sum up, sequence to sequence model with attention really let decoder focus on different part of input sentence and thus have a better performance in text summarization.

Q6: Explain Rouge-L

L is the first word of LCS (Longest common subsequence)
LCS: given two sequences X and Y, $Z = LCS(X, Y)$ where Z is the subsequence of X and also Y and Z is longest one among all subsequences.

Scores of LCS is shown below,

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

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

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

where X, Y, denoted the target summary and the predict summary respectively.
where m, n denoted the length of the target summary and the length of the predict summary respectively.