ADL HW1 Report

Describe how do you use the data:

- **How do you tokenize the data.**
  - Using spaCy - 'en_core_web_sm' dictionary for tokenization
  - Not filtering out the puntuation or other stemming.
- **The pre-trained embedding you used.**
  - Using Glove.840B.300d.txt
  - 840B tokens, 2.2M vocab, cased, 300d vectors
  - Using only the tokens appear in train, valid and test dataset(97513 words)
- **Describe your extractive summarization model.**

```
Encoder(
  (embedding): Embedding(97513, 300)
  (rnn): LSTM(300, 128, bidirectional=True)
  (proj): Linear(in_features=256, out_features=1, bias=True)
  (dropout): Dropout(p=0.5)
)
```

- **performance of your model.(on the validation set)**

"mean": {
"rouge-1": 0.1932604508019379,
"rouge-2": 0.029456076543035902,
"rouge-l": 0.1323552340890648
},
"std": {
"rouge-1": 0.07821801211071075,
"rouge-2": 0.04093055017990801,
"rouge-l": 0.05548583287978239
}

- **the loss function you used.**

```
criterion = nn.BCEWithLogitsLoss(reduction='none',pos_weight= 10))
# pos_weight: a weight of positive examples.
```

For pos weight, I set it to be nearly 10 for a better performance of the model. (accurate ratio of 2 class is 6.8)

- **The optimization algorithm (e.g. Adam), learning rate and batch size.**
  - optimizer: Adam
  - Parameter:

```
{
    'learning_rate': 0.0001
    'embed_size': 300,
    'batch_size': 64,
    'pos_weight': 10,
    'rnn_hidden_size': 128,
}

```

- **Post-processing strategy**
  - for post-processing, I use top-k technic to choose the top k most labelling sentence for output

- $P_n = \sigma(M(x_n)) > 0,\ \sigma = sigmoid,\ M = encoder$
  $Ans = Argmax_k(\sum P_n)$

Describe your Seq2Seq + Attention summarization model.

```
Seq2Seq(
  (encoder): Encoder(
    (embed): Embedding(97513, 300)
    (gru): GRU(300, 300, num_layers=2, dropout=0.2, bidirectional=True)
  )
  (decoder): Decoder(
    (embed): Embedding(97513, 300)
    (dropout): Dropout(p=0.2, inplace)
    (attention): Attention(
      (attn): Linear(in_features=600, out_features=300, bias=True)
    )
    (gru): GRU(600, 300, dropout=0.2)
    (out): Linear(in_features=600, out_features=97513, bias=True)
  )
)
```

- **performance of your model.(on the validation set)**
  - 10 epoches - 4.5 hours

"mean": {
"rouge-1": 0.2559601574299985,
"rouge-2": 0.07379624637401633,
"rouge-l": 0.2150483739650911
},
"std": {
"rouge-1": 0.12518714935054517,
"rouge-2": 0.09629784492243731,
"rouge-l": 0.11587473409967955
}

- **the loss function of seq2seq attention.**

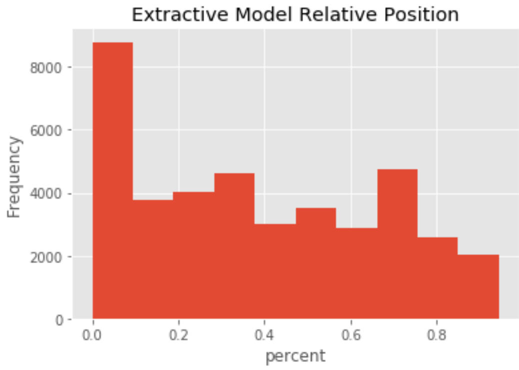
```
F.nll_loss(ignore_index=PAD_token)
# PAD_token is set to 0 when preprocessing.
```

$Nlloss : l(y) = -log(y)$
NLL loss has a better performance on multi-class task

- **The optimization algorithm (e.g. Adam), learning rate and batch size.**
  - optimizer: Adam
  - Parameter:
    - learning_rate: 0.0001
    - embed_size: 300,
    - batch_size: 16,
    - rnn_hidden_size: 300

Plot the distribution of relative locations

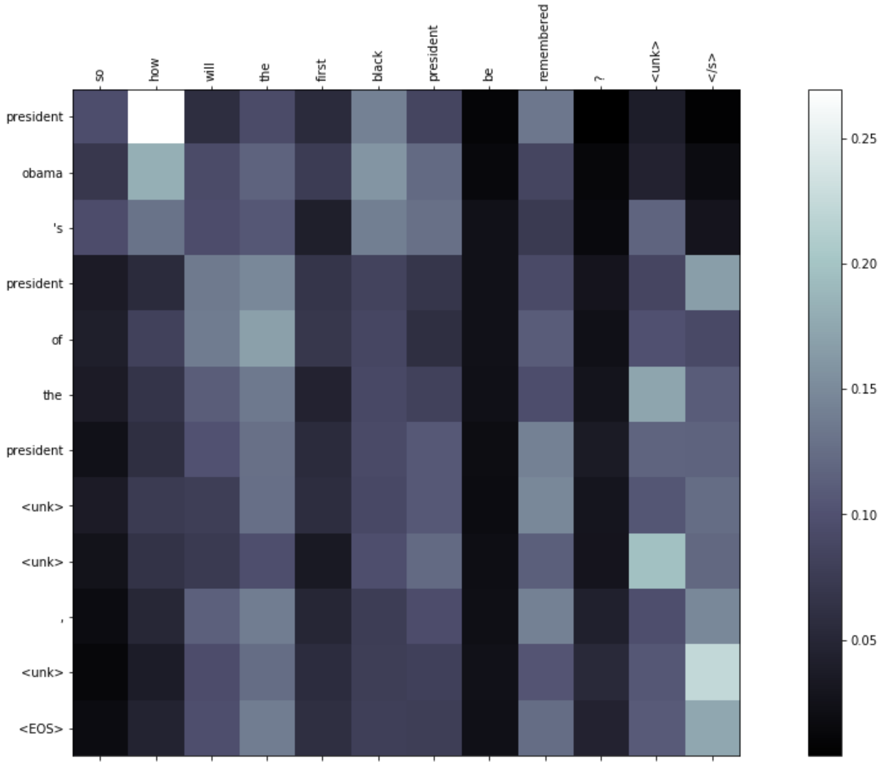
- **Plot the distribution of relative locations of predicted sentences by extractive model**

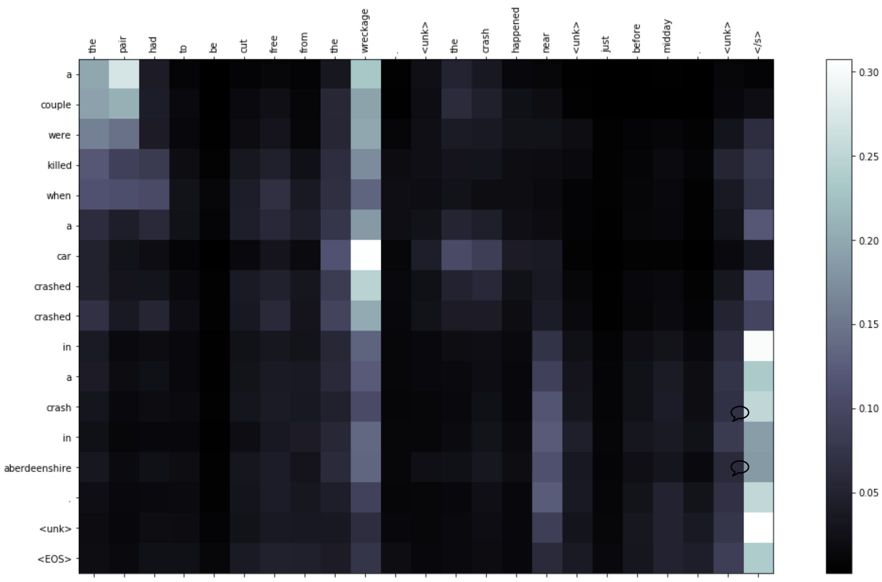


  - The statistic shows that the highest point of relative positiion is at the very first part of the article.
  - The second highest point is nearing the middle of the end.

Visualize the attention weights

- **Take one example in the validation set and visualize the attention weights (after softmax)**





  - The output show that if the input phrase is Noun, the attention weight is averagely higher than the others.

        - hypothesis is that the word next to noun is hard to defined.
    - Some digit or timestamp in input might have higher weight for the output as well.
    - If the input is article(like a, the), output weight might put attention on the noun.

Explain Rouge-L

- Explain the way Rouge-L is calculated.
    - Suppose X is the candidate of summary, Y is the ground truth
    - define $LCS(X,Y)$ is the Longest Common Subsequence without considering order
    - Then we can calculate:
        - $Precision_{lcs} = \frac{LCS(X,Y)}{n}$, where n is the length of Y
        - $Recall_{lcs} = \frac{LCS(X,Y)}{m}$, where m is the length of X
    - Finally, we useP_{lcs}(precision) R_{lcs}(Recall) to calculate F_{lcs}(the score we want)
    - $F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}}$where β is usually set to $\inf$