Applied Deep Learning: Assignment 1 - Summarization

b07902047 羅啓帆

April 2020

Q1: Data processing

For extractive and seq2seq model, I used conceptnet-numberbatch as my pretrained embedding. For attention model, I used glove.840B.300d. The following steps are the my text preprocessing pipeline:

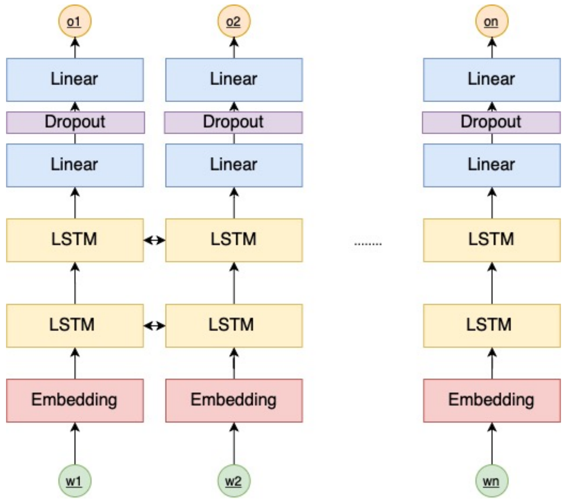
1. Turned all alphabets to lowercase
2. Replace contractions with expaned words e.g. ain't → am not. The full list I use is given here
3. This step is different for text and summary. I used functions in gensim.parsing.preprocessing. The following listed functions are applied sequentially.
   - text(X):
     (a) strip_tags()
     (b) strip_punctuation()
     (c) remove_stopwords() (only in extractive and seq2seq)
     (d) strip_numeric()
     (e) strip_non_alphanum()
     (f) strip_multiple_whitespaces()
   - summary(Y):
     (a) strip_tags()
     (b) strip_punctuation()
     (c) strip_multiple_whitespaces()
4. Split strings into list of words.
5. Insert $<EOS>$(End Of Sentence) tokens. This step is different for extractive model and abstractive model.
   - Extractive: Insert $<EOS>$ token after the end of **every** sentence
   - Abstractive: Insert $<EOS>$ token after the end of the whole article
6. Make the length of all article same by padding $<PAD>$ token or truncate it. The specfic length is different, which is given below:
   - Extractive text: 301
   - Abstractive text: 251
   - Abstractive summary: 30 in seq2seq and 40 in attention
7. Load pretrained word embeddings pre, and count the frequency of words in train, valid, test data.
8. Create word embedding emb based on the result above:
   - s is in pre: put s in emb
   - frequency of s is greater then a given number(5 for seq2seq, ∞ for attention, 2 for extractive): put s in emb and create a random vector for it

   Note that four special words are added in emb: $<SOS>, <EOS>, <PAD>, <UNK>$.
9. Transform words into index using emb. If a word is not in emb, replace it with the id of $<UNK>$.

Q2: Describe your extractive summarization model

1. Model structure

    - Forward:



    - Some details:

```
Model(
  (embedding): Embedding(107785, 300)
  (rnn): LSTM(300, 300, num_layers=2, bidirectional=True)
  (out): Sequential(
    (0): Linear(in_features=600, out_features=300, bias=True)
    (1): Dropout(p=0.5, inplace=False)
    (2): Linear(in_features=300, out_features=1, bias=True)
  )
)
```

Listing 1: Model structure

2. Performance on validation set

    - mean
        - Rouge-1: 0.193957665188440427
        - Rouge-2: 0.030904724823523365
        - Rouge-L: 0.132143998509014 98
    - std
        - Rouge-1: 0.07646853018548 51
        - Rouge-2: 0.041372657658442495
        - Rouge-L: 0.054225718444142 59

3. Loss function

    - $nn.BCEWithLogitsLoss(pos\_weight = [301])$, where 301 is the length of text

4. Optimization

    - optim.Adadelta
    - used gradient clipping with GRAD_MAX = 1 i.e clip gradient into $[-1, 1]$
    - learning rate = 1.0
    - batch size = 256

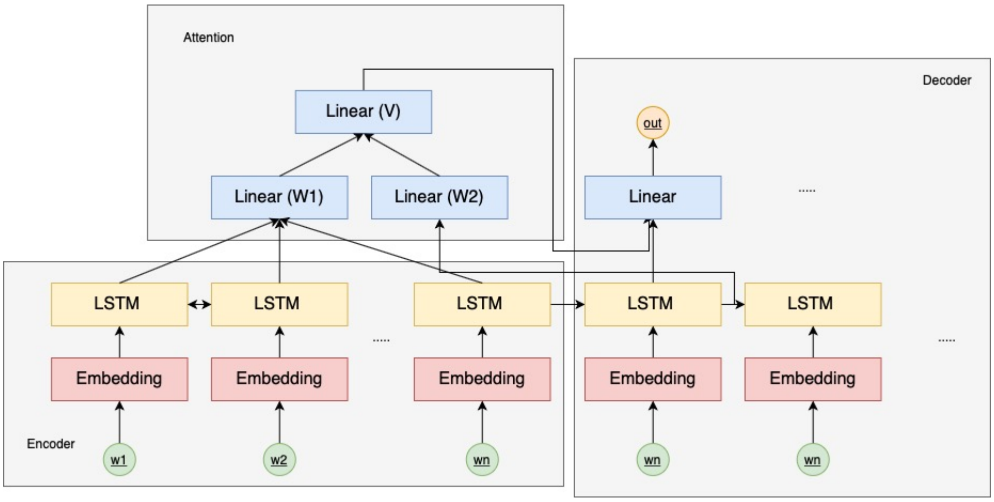
5. Post-processing

    - Collect all probabilities that corresponding to $< EOS >$ tokens.
    - Sort them by probabilities from large to small
    - Pick the top 2 of them as prediction

Q3: Describe your Seq2Seq + Attention model

1. Model structure
   I excluded softmax layers and dropout layers.

   - Forward:



   - Some details:

```
Seq2seq(
  (encoder): EncoderRNN(
    (rnn): GRU(300, 256, bidirectional=True)
  )
  (decoder): DecoderRNN(
    (rnn): GRU(300, 512)
    (out): Linear(in_features=1024, out_features=83674, bias=True)
    (softmax): LogSoftmax()
  )
  (attention): Attention(
    (w1): Linear(in_features=512, out_features=512, bias=True)
    (w2): Linear(in_features=512, out_features=512, bias=True)
    (v): Linear(in_features=512, out_features=1, bias=True)
  )
  (embedding): Embedding(83674, 300)
  (dropout): Dropout(p=0.5, inplace=False)
)
```

Listing 2: Model structure

2. Performance on validation set

   - mean
     - Rouge-1: 0.263477745546085
     - Rouge-2: 0.07386982156504307
     - Rouge-L: 0.21579091132387265
   - std
     - Rouge-1: 0.1266203577567892
     - Rouge-2: 0.0983715038018696
     - Rouge-L: 0.11795962757388617

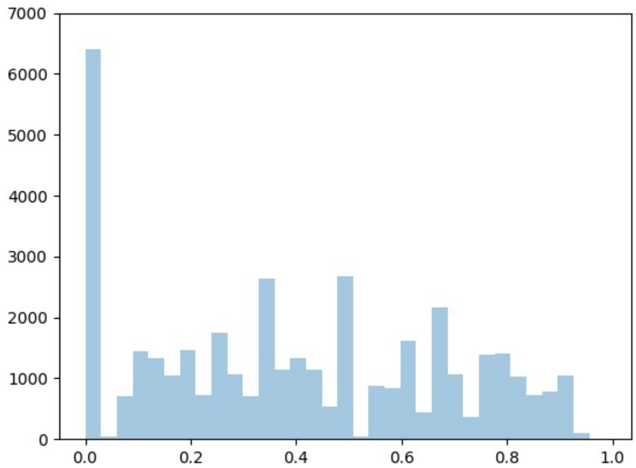
3. Loss function

   - $nn.NLLLoss(ignore\_index = PAD\_token)$, note that I excluded $< PAD >$ from loss calculation.

4. Optimization

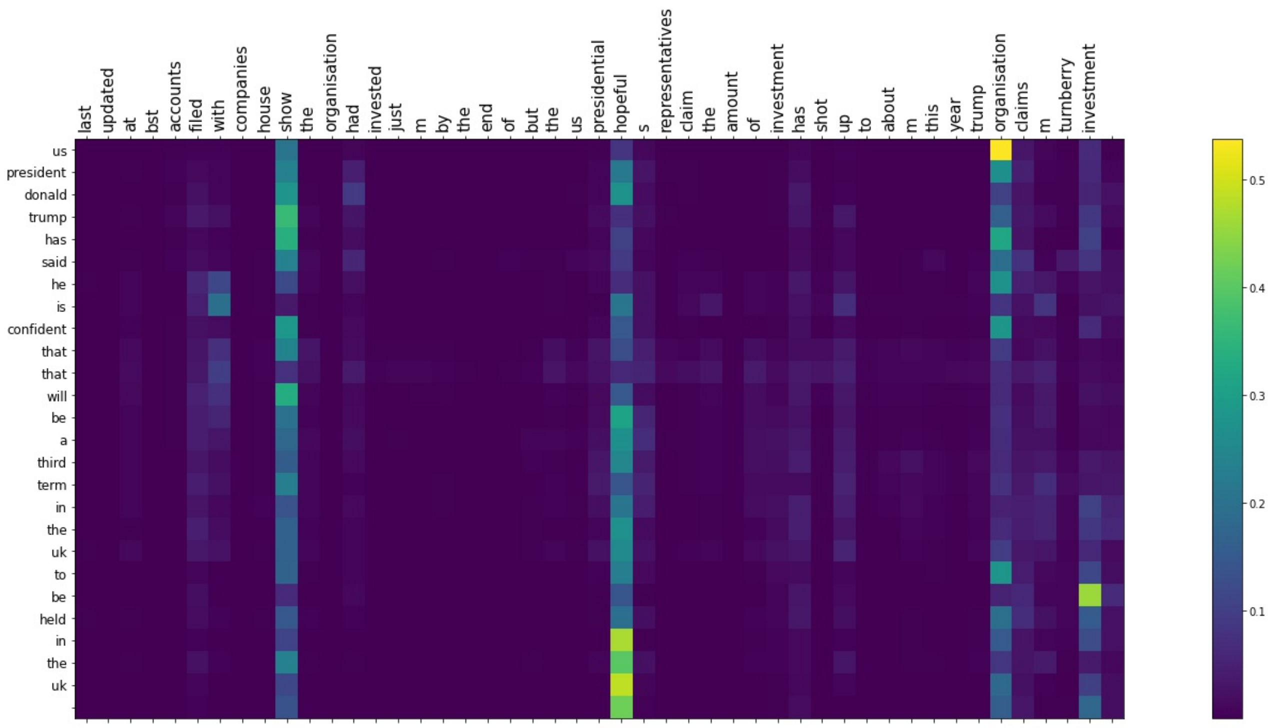
- optim.Adam(amsgrad=True)
- learning rate = 0.001
- batch size = 32

Q4: Plot the distribution of relative locations



We can observe that lots of sentence chosen by our model have small relative location, which means that the model usually think that the first few senteces can summarize the whole paragraph.

Q5: Visualize the attention weights



We can observe that the model focus on only a few words. In this example, the model mainly focused on *show, hopeful, organisation, investment.* This might be reasonable as usually only a few words are important for the summary.

Q6: Explain Rouge-L

Let x, y be two strings and β be a given constant. Then,

$$\text{Rouge-L(x, y)}=\begin{cases} P_{lcs} = \frac{LCS(x,y)}{len(x)} & , precision \\ R_{lcs} = \frac{LCS(x,y)}{len(y)} & , recall \\ F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}} & , f-measure \end{cases}$$

where $LCS(x, y)$ represent the Longest Common Subsequence between x and y and $len(x)$ represent the length of x.

Q7: Beam Search

1. Pseudocode

```
def beam_search(x, beam_size): # single data
    encoder_outputs, encoder_hidden = encoder(x)

    # nodes are compared by there average score i.e. score / length
    start_node = (encoder_hidden, '<SOS>', None, 0, 1) # (hidden, idx, prev_node
    , score, length) # Start Of Sentence token

    all_nodes = [start_node]
    now_nodes = [start_node]
    end_pq = PriorityQueue() # min heap (maintain max value)

    for j in 1, 2, ..., MAX_TARGET_LEN:
        if now_node is empty:
            break
        pq = PriorityQueue() # min heap

        for node in now_nodes:
            input, hidden = node.idx, node.hidden
            output, hidden = decoder(input, hidden)
            topv, topi = top beam_size highest probability in output
            for (score, idx) in zip(topv, topi):
                nxt_node = (hidden, idx, node, node.score + score, idx)
                pq.put(nxt_node)

        now_nodes = []
        for _ in 1, 2, ..., beam_size:
            node = pq.top() # node with maximum average score
            all_nodes.append(node)
            if node.idx == '<EOS>' or j == TARGET_LEN:
                end_pq.put(node)
            else:
                now_nodes.append(node)

    best_node = end_pq.top() # node with maximum average score

    predict = [best_node]
    while best_node.prev_node is not None:
        best_node = best_node.prev
        predict.append(best_node)
    remove '<SOS>' from predict

    return predict
```

Listing 3: Model structure

2. Performance compare
   I sampled 3200 datas from valid dataset to generate the following table.

R stands for Rouge	R-1 mean	R-1 std	R-2 mean	R-2 std	R-L mean	R-L std
w/ beam & w/ attention	0.2654	0.1335	0.0807	0.1051	0.2201	0.1249
w/o beam & w/ attention	0.2615	0.1260	0.0739	0.0976	0.2150	0.1175
w/ beam & w/o attention	0.2099	0.1175	0.0466	0.0806	0.1737	0.1056
w/o beam & w/o attention	0.2115	0.1150	0.0440	0.0765	0.1761	0.1042

3. Samples where beam search performs better

   - text (input): the three day event opened on friday evening with an aerial display fireworks live music and a proposal singer ste johnson was performing on stage at the event when he asked his partner mags foster to marry him sunderland city council said saw one of the show s best days ever on the saturday and a total of one million spectators overall attractions over the weekend included parachutists and a battle of britain memorial flight councillor john kelly said we are very proud of our fantastic sunderland international airshow we have some teams coming here for the very first time and i m sure the crowds will give them a huge warm welcome they will remember for a long time to come
   - summary (target): thousands of people gathered at and roker seafront for the sunderland international airshow
   - without beam: a memorial service in sunderland has been cancelled after a series of concerts goers
     - Rouge-1: 0.14814
     - Rouge-2: 0.0
     - Rouge-L: 0.07407
   - with beam(beam_size=3): thousands of people have gathered at the top of the world s largest venues in sunderland
     - Rouge-1: 0.48275
     - Rouge-2: 0.22222
     - Rouge-L: 0.48275

   The score above are f-measures, and both of them uses attention mechanism.

4. Why beam search may perform better

   Beam search may performs better because in the beginning, the top words might have similar probability (score), and one word might have a small advantage over another. Thus, if we chooes it greedily i.e. choose the one with the highest probability, we might miss a more possible sentence. For the example above, *a* is given a small advantage over *thousands* in the beginning, causing the model to choose a sentence with lower probability when decoding greedily.