CS 533: Natural Language Processing **(Due: 03/10/20)**

Assignment 3

Name: Twisha Naik, *NetID:* tn268 *Students discussed with:* phj15, kd706

Problem 1: Backpropagation

(((1 + 1 + 3 + 1 + 1) + (1 + 1 + 4) = 13 points))

1. (Scalar-Valued Variables):

   (a) Computation Graph:

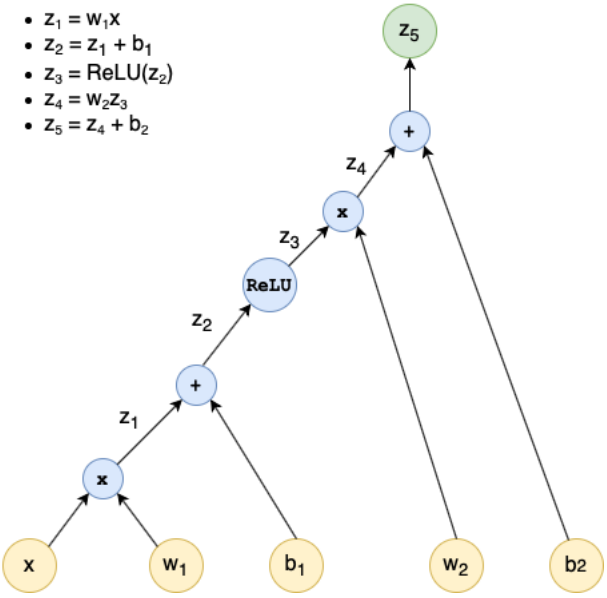

Figure 1: Computation Graph

   (b) Forward pass

$$x = 1, w_1 = \frac{1}{4}, b_1 = 0, w_2 = \frac{1}{3}, b_2 = 0$$

$$z_1 = w_1 x = (\frac{1}{4})(1) = \frac{1}{4}$$

$$z_2 = z_1 + b_1 = \frac{1}{4} + 0 = \frac{1}{4}$$

$$z_3 = ReLU(z_2) = ReLU(\frac{1}{4}) = \frac{1}{4}$$

$$z_4 = w_2 z_3 = (\frac{1}{3})(\frac{1}{4}) = \frac{1}{12}$$

$$z_5 = z_4 + b_2 = \frac{1}{12} + 0 = \frac{1}{12}$$

(c) Backpropagation

$$z_5 = z_4 + b_2$$
$$\implies \frac{\partial z_5}{\partial z_4} = \frac{\partial z_4}{\partial z_4} + \frac{\partial b_2}{\partial z_4} = 1 + 0 = 1$$
$$\implies \boxed{\frac{\partial z_5}{\partial b_2} = \frac{\partial z_4}{\partial b_2} + \frac{\partial b_2}{\partial b_2} = 0 + 1 = 1}$$

$$z_4 = w_2 z_3$$
$$\implies \frac{\partial z_4}{\partial z_3} = w_2 \frac{\partial z_3}{\partial z_3} + z_3 \frac{\partial w_2}{\partial z_3}$$
$$= w_2(1) + z_3(0) = w_2 = \frac{1}{3}$$
$$\implies \frac{\partial z_5}{\partial z_3} = \frac{\partial z_4}{\partial z_3}\frac{\partial z_5}{\partial z_4} = (\frac{1}{3})(1) = \frac{1}{3}$$

$$\implies \frac{\partial z_4}{\partial w_2} = w_2 \frac{\partial z_3}{\partial w_2} + z_3 \frac{\partial w_2}{\partial w_2}$$
$$= w_2(0) + z_3(1) = z_3 = \frac{1}{4}$$
$$\implies \boxed{\frac{\partial z_5}{\partial w_2} = \frac{\partial z_4}{\partial w_2}\frac{\partial z_5}{\partial z_4} = (\frac{1}{4})(1) = \frac{1}{4}}$$

$$z_3 = ReLU(z_2)$$
$$\implies \frac{\partial z_3}{\partial z_2} = \frac{\partial ReLU(z_2)}{\partial z_2}$$
$$= 1(since z_2 > 0)$$
$$\implies \frac{\partial z_5}{\partial z_2} = \frac{\partial z_3}{\partial z_2}\frac{\partial z_5}{\partial z_3} = (1)(\frac{1}{3}) = \frac{1}{3}$$

$$z_2 = z_1 + b_1$$
$$\implies \frac{\partial z_2}{\partial z_1} = \frac{\partial z_1}{\partial z_1} + \frac{\partial b_1}{\partial z_1}$$
$$= 1 + 0 = 1$$
$$\implies \frac{\partial z_5}{\partial z_1} = \frac{\partial z_2}{\partial z_1}\frac{\partial z_5}{\partial z_2} = (1)(\frac{1}{3}) = \frac{1}{3}$$

$$\implies \frac{\partial z_2}{\partial b_1} = \frac{\partial z_1}{\partial b_1} + \frac{\partial b_1}{\partial b_1}$$
$$= 0 + 1 = 1$$
$$\implies \boxed{\frac{\partial z_5}{\partial b_1} = \frac{\partial z_2}{\partial b_1}\frac{\partial z_5}{\partial z_2} = (1)(\frac{1}{3}) = \frac{1}{3}}$$

$$z_1 = w_1 x$$
$$\implies \frac{\partial z_1}{\partial x} = w_1 \frac{\partial x}{\partial x} + x \frac{\partial w_1}{\partial x}$$
$$= w_1(1) + z_3(0) = w_1 = \frac{1}{4}$$
$$\implies \boxed{\frac{\partial z_5}{\partial x} = \frac{\partial z_1}{\partial x}\frac{\partial z_5}{\partial z_1} = (\frac{1}{4})(\frac{1}{3}) = \frac{1}{12}}$$

$$\implies \frac{\partial z_1}{\partial w_1} = w_1 \frac{\partial x}{\partial w_1} + x \frac{\partial w_1}{\partial w_1}$$
$$= w_1(0) + x(1) = x = 1$$
$$\implies \boxed{\frac{\partial z_5}{\partial w_1} = \frac{\partial z_1}{\partial w_1}\frac{\partial z_5}{\partial z_1} = (1)(\frac{1}{3}) = \frac{1}{3}}$$

(d) Addition of the skip connection

i. Computation Graph

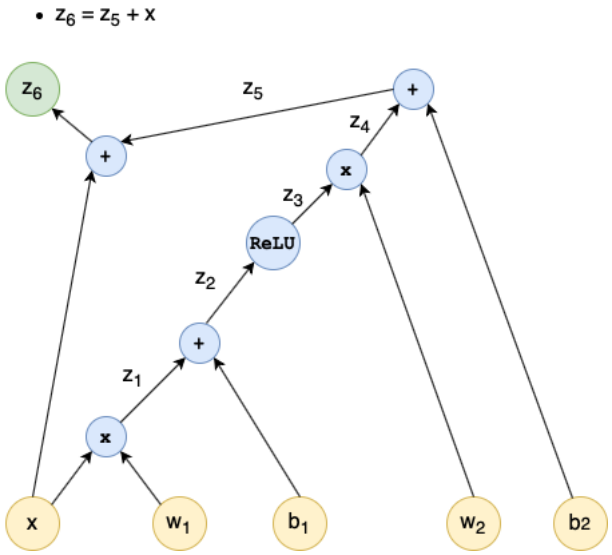

Figure 2: Computation Graph

ii. Forward Pass

$$\boxed{z_6 = z_5 + x = \frac{1}{12} + 1 = \frac{13}{12}}$$

iii. Backward Pass

$$z_6 = z_5 + x$$

$$\implies \boxed{\frac{\partial z_6}{\partial x} = \frac{\partial x}{\partial x} + \frac{\partial z_5}{\partial x} = 1 + \frac{1}{12} = \frac{13}{12}}$$

All the other derivatives will remain the same as the derivatives with respect to z5.

$$z_6 = z_5 + x$$

$$\implies \frac{\partial z_6}{\partial b_2} = \frac{\partial z_6}{\partial z_5}\frac{\partial z_5}{\partial b_2} = (1)(1) = 1$$
$$\implies \frac{\partial z_6}{\partial w_2} = \frac{\partial z_6}{\partial z_5}\frac{\partial z_5}{\partial w_2} = (1)(\frac{1}{4}) = \frac{1}{4}$$
$$\implies \frac{\partial z_6}{\partial b_1} = \frac{\partial z_6}{\partial z_5}\frac{\partial z_5}{\partial b_1} = (1)(\frac{1}{3}) = \frac{1}{3}$$
$$\implies \frac{\partial z_6}{\partial w_1} = \frac{\partial z_6}{\partial z_5}\frac{\partial z_5}{\partial w_1} = (1)(\frac{1}{3}) = \frac{1}{3}$$

(e) Sensitivity of the function with x

Due to the direct skip connection added to the network, the value of derivative with respect to x increased. This means that with the same amount of change in x, the value of function changes more. Therefore, the sensitivity of the function with respect to x increases.

2. (Vector-Valued Variables):

   (a) Computation Graph:

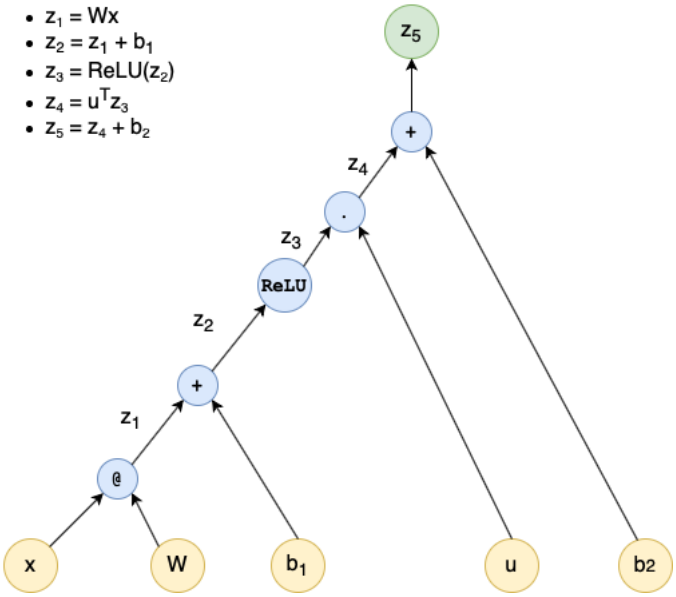

Figure 3: Computation Graph

   (b) Forward pass

$$x = \begin{bmatrix}1\\1\end{bmatrix}, W = \begin{bmatrix}-1 & -1\\1 & 1\end{bmatrix}, b_1 = \begin{bmatrix}0\\0\end{bmatrix}, u = \begin{bmatrix}1\\1\end{bmatrix}, b_2 = 0$$

$$z_1 = Wx = \begin{bmatrix}-1 & -1\\1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}-2\\2\end{bmatrix}$$

$$z_2 = z_1 + b_1 = \begin{bmatrix}-2\\2\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}-2\\2\end{bmatrix}$$

$$z_3 = ReLU(z_2) = ReLU(\begin{bmatrix}-2\\2\end{bmatrix}) = \begin{bmatrix}0\\2\end{bmatrix}$$

$$z_4 = u^T z_3 = \begin{bmatrix}1 & 1\end{bmatrix}\begin{bmatrix}0\\2\end{bmatrix} = 2$$

$$z_5 = z_4 + b_2 = 2 + 0 = 2$$

   (c) Backpropagation

$$\begin{aligned} & z_5 = z_4 + b_2 \\ \implies & \frac{\partial z_5}{\partial z_4} = \frac{\partial z_4}{\partial z_4} + \frac{\partial b_2}{\partial z_4} = 1 + 0 = 1 \\ \implies & \boxed{\frac{\partial z_5}{\partial b_2} = \frac{\partial z_4}{\partial b_2} + \frac{\partial b_2}{\partial b_2} = 0 + 1 = 1} \end{aligned} \tag{1}$$

$$z_4 = u^T z_3$$
$$\implies \frac{\partial z_5}{\partial z_3} = u\frac{\partial z_5}{\partial z_4} = \begin{bmatrix}1\\1\end{bmatrix} (\because Lemma1)$$

$$z_4 = u^T z_3 = z_3^T u$$
$$\implies \boxed{\frac{\partial z_5}{\partial u} = \frac{\partial z_5}{\partial z_4} z3 = \begin{bmatrix}0\\2\end{bmatrix}}$$

$$z3 = ReLU(z_2)$$
$$\implies \frac{\partial z_5}{\partial z_2} = \frac{\partial z_5}{\partial z_3}\frac{\partial z_3}{\partial z_2}$$
$$= \begin{bmatrix}1\\1\end{bmatrix} \odot \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$$

$$z_2 = z_1 + b_1$$
$$\implies \frac{\partial z_2}{\partial z_1} = \frac{\partial z_1}{\partial z_1} + \frac{\partial b_1}{\partial z_1}$$
$$= \begin{bmatrix}1\\1\end{bmatrix} + \begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}$$

$$\implies \frac{\partial z_5}{\partial z_1} = \frac{\partial z_5}{\partial z_2}\frac{\partial z_2}{\partial z_1} = \begin{bmatrix}0\\1\end{bmatrix} \odot \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$$

$$\implies \boxed{\frac{\partial z_5}{\partial b_1} = \frac{\partial z_5}{\partial z_2}\frac{\partial z_2}{\partial b_1} = \begin{bmatrix}0\\1\end{bmatrix} \odot \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}}$$

$$z_1 = Wx$$
$$\implies \frac{\partial z_5}{\partial x} = W^T\frac{\partial z_5}{\partial z_1} = \begin{bmatrix}-1 & 1\\-1 & 1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}$$
$$\boxed{\frac{\partial z_5}{\partial x} = \begin{bmatrix}1\\1\end{bmatrix}}$$
$$\implies \frac{\partial z_5}{\partial W} = \frac{\partial z_5}{\partial z_1}x^T = \begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix}$$
$$\frac{\partial z_5}{\partial W} = \begin{bmatrix}0 & 0\\1 & 1\end{bmatrix}$$

Problem 2: Self-Attention ((4 + 4 = 8 points))

(a) Computation Graph:

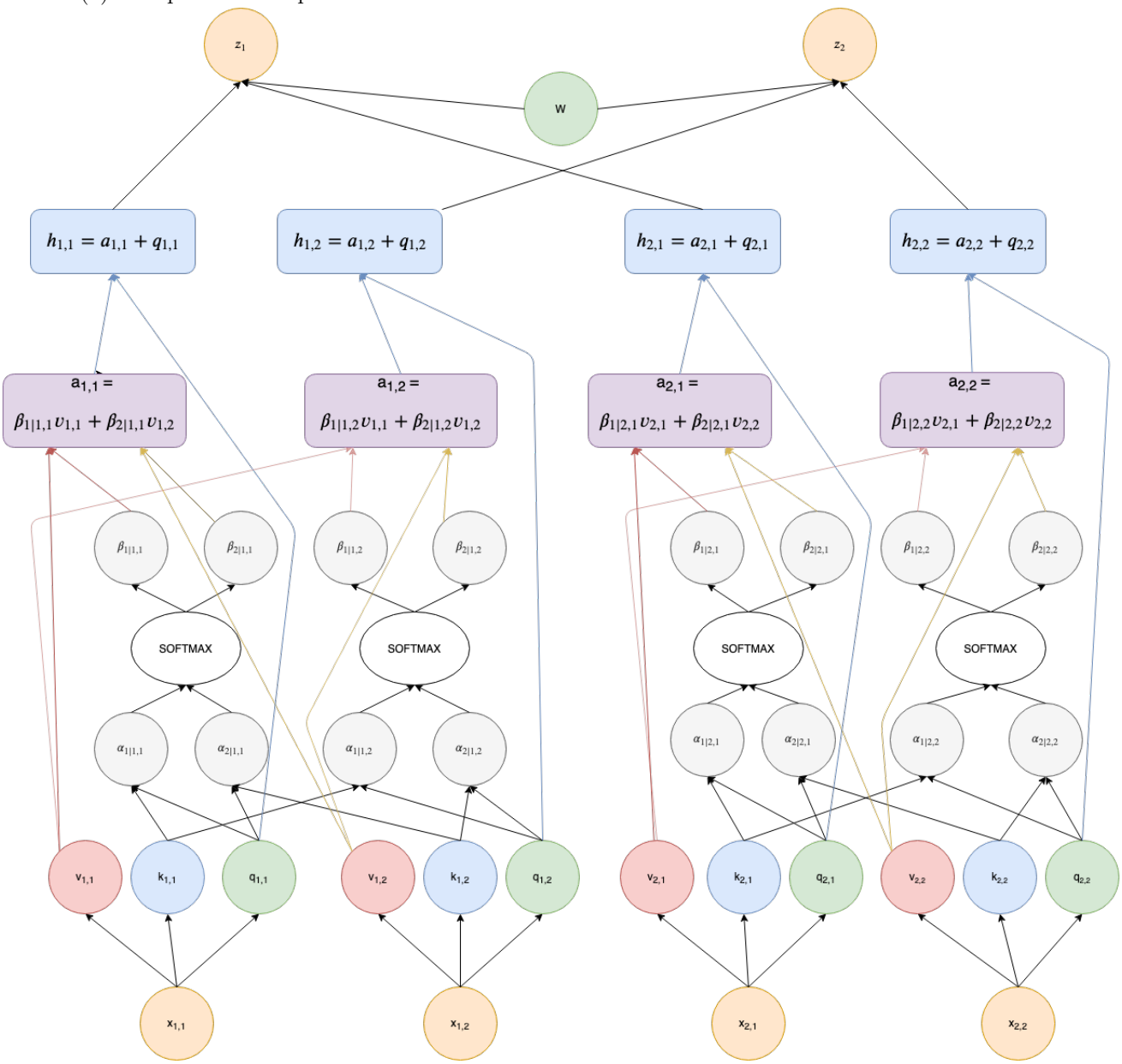

Figure 4: Computation Graph

(b) Forward pass

$$q_{h,t} = (W_h)^Q x_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 2 & 3 \end{bmatrix}$$

$$k_{h,t} = (W_h)^K x_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 2 & 3 \end{bmatrix}$$

$$v_{h,t} = (W_h)^V x_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 2 & 3 \end{bmatrix}$$

Once we have the three encodings (key, queries and values) for each word, we define α.

$\alpha_{t^{`}|h,t} = k_{h,t^{`}} q_{h,t}$ for t = 1...T and $t^{`} = 1....T$,

$$\alpha_{1|h,t} = \begin{bmatrix} 1 & -1 \\ 4 & 6 \end{bmatrix}$$
$$\alpha_{2|h,t} = \begin{bmatrix} -1 & 1 \\ 6 & 9 \end{bmatrix}$$

By applying the softmax function on α, we calculate β. For t = 1....T,

$$(\beta_{1|h,t}....\beta_{T|h,t}) = softmax(\alpha_{1|h,t}....\alpha_{T|h,t})$$
$$\implies (\beta_{1|h,t}, \beta_{2|h,t}) = softmax(\alpha_{1|h,t}, \alpha_{2|h,t})$$

For $t^{(\prime)} = 1$:

$$\beta_{1|h,t} = \begin{bmatrix} 0.8808 & 0.1192 \\ 0.1192 & 0.8808 \end{bmatrix}$$

Similarly, for $t^{(\prime)} = 2$:

$$\beta_{2|h,t} = \begin{bmatrix} 0.1192 & 0.8808 \\ 0.4742 & 0.9525 \end{bmatrix}$$

For t = 1....T,

$$a_{h,t} = \sum_{t'=1}^{T} \beta_{t'|h,t} v_{h,t'}$$

For t = 1:

$$\text{Attention Head 1} : a_{11} = \beta_{1|1,1} v_{1,1} + \beta_{2|1,1} v_{1,2} = (0.8808)(1) + (0.1192)(-1) = 0.7616$$
$$\text{Attention Head 2} : a_{21} = \beta_{1|2,1} v_{2,1} + \beta_{2|2,1} v_{2,2} = (0.1192)(2) + (0.8808)(3) = 2.8808$$

For t = 2:

$$\text{Attention Head 1} : a_{12} = \beta_{1|1,2} v_{1,1} + \beta_{2|1,2} v_{1,2} = (0.1192)(1) + (0.8808)(-1) = -0.7616$$
$$\text{Attention Head 2} : a_{22} = \beta_{1|2,2} v_{2,1} + \beta_{2|2,2} v_{2,2} = (0.4742)(2) + (0.9525)(3) = 2.9525$$

$$a_{h,t} = \begin{bmatrix} 0.7616 & -0.7616 \\ 2.8808 & 2.9525 \end{bmatrix}$$

Now, for t = 1...T,

$$h_{h,t} = a_{h,t} + q_{h,t}$$
$$h_{h,t} = \begin{bmatrix} 0.7616 & -0.7616 \\ 2.8808 & 2.9525 \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ 2 & 3 \end{bmatrix}$$
$$h_{h,t} = \begin{bmatrix} 1.7616 & -1.7616 \\ 4.8808 & 5.9525 \end{bmatrix}$$

Finally,

$$z_t = W.(h_{1,t} + ... + h_{T,t})$$
$$z_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1.7616 \\ 4.8808 \end{bmatrix} = \begin{bmatrix} 1.7616 \\ 4.8808 \end{bmatrix}$$
$$z_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -1.7616 \\ 5.9525 \end{bmatrix} = \begin{bmatrix} -1.7616 \\ 5.9525 \end{bmatrix}$$

Problem 3: Programming ((1 + 2 + (1 + 1 + 1) + (1 + 1 + 1) + 5 + 4 + 1 + 1 + 3 = 23 points))

1. Implement get_ngram_counts

```
    def get_ngram_counts(refs, hyp, n):
        hyp_ngrams = [tuple(hyp[i:i + n]) for i in range(len(hyp) - n + 1)]
        num_hyp_ngrams = max(1, len(hyp_ngrams))  # Avoid empty

        hyp_ngrams_count = Counter(hyp_ngrams)

        refs_ngrams_count = []
        for ref in refs:
          ref_ngrams = [tuple(ref[i:i + n]) for i in range(len(ref) - n + 1)]
          refs_ngrams_count.append(Counter(ref_ngrams))

        # TODO: Implement
        num_hyp_ngrams_in_refs_clipped = 0
        for n_gram in hyp_ngrams_count:
          c = hyp_ngrams_count[n_gram]
          max_ref_count = 0
          for i, ref in enumerate(refs):
            max_ref_count = max(max_ref_count, refs_ngrams_count[i][n_gram])

          num_hyp_ngrams_in_refs_clipped += min(c, max_ref_count)

        return num_hyp_ngrams_in_refs_clipped, num_hyp_ngrams

```

2. Implement compute_bleu

```
    def compute_bleu(reflists, hyps, n_max=4, use_shortest_ref=False):
        assert len(reflists) == len(hyps)

        N = len(hyps)

        prec_mean = 1.0  # TODO: Implement
        for n in range(1, 5):
          pn_num = 0
          pn_deno = 0
          for i in range(N):
            an, bn = get_ngram_counts(reflists[i], hyps[i], n)
            pn_num += an
            pn_deno += bn
          prec_mean *= (float(pn_num)/pn_deno)
        prec_mean = prec_mean ** 0.25

        brevity_penalty = 0  # TODO:Implement
        H = 0, R = 0
        for i in range(N):
          h = len(hyps[i])
          H += h

          r_max = sys.maxsize
          r = 0
          refs = reflists[i]
          for ref in refs:
            if abs(len(ref) - h) < r_max:
              r = len(ref)
          R += r
        brevity_penalty = min(1, math.exp(1 - (float(R)/H)))

        bleu = brevity_penalty * prec_mean
        return bleu

```

3. The default training command "python main.py –train" yields a perplexity of **196.77**.
   Explain:

   (a) What does bptt (stands for "backpropagation through time") do?

   - The parameter bptt is used to fix the sequence length in the continuous training model. A batch size is fixed and each of the example in the batch is of the length bptt.

- We process data batch wise. Here, the batch size is 20 which means that there are 20 examples in one batch each of size 35 (=bptt). Loss and gradients are calculated for each example of the batch using "step_on_batch" function. Thus, the backpropogation happens only for bptt length as that is the sequence length (or length of the example).

(b) In which function does the model compute the loss and calculate gradients?
The loss and gradients are calculated in the "**step_on_batch**" function. The following lines in the function does the needful.

```
    # Calculates the Cross entropy loss by comparing:
    # 1. output (model prediction) and 2. golds (actual word)
    loss = self.avgCE(output, golds)

    # Calculates the gradients for each of the model parameter
    loss.backward()

```

(c) How does the model "carry over" the hidden state from the previous batch?

- For continuous data, the parameter "**start**" is **False** in the "step_on_batch". The forward function of model is called with **start=False**.

```
        if start:
            self.dec.init_state(batch_size=batch_size, encoder_final=final)
        else:
            self.dec.detach_state()

```

- Whenever the start parameter is False, the **deatch_state** function of the decoder is called which does not reinitialise the hidden state but continues using the one already set using the previous example in the batch or the previous batch.
- Thus, by avoiding reinitalization of weights for every batch, the model maintains the hidden state from the previous batch. This also makes the hidden state is deatched which means gradients are no longer propagated further.

4. Explain:

(a) In this case bptt is no longer used. Why?

- In this case we have a mapping of sentence in source language mapped to the sentence in target language. Thus, we do not want to fix a sequence length. We take entire sentences as input in form of a batch and process them.
- Each sentence can be of different length. To make sure the longer sentences are learned properly, the size of each example in the batch is set to the maximum length of sentence in the batch.
- Thus, instead of using a fixed length of bptt it uses the size of longest sequence in the batch. Hence, bptt is not used.

(b) In which function does the model encode the source sentence?

- When the model is continuous there is no encoder associated but for translation model an encoder is present.
- The source sentence is passed into the encoder. Using the forward function of the encoder, the embeddings are generated and then using an LSTM model we get the encoded sentence.

(c) In which function does the model condition on the final encoding of the source sentence?

- In the **forward** function of the model, for a translation model, the **is_conditional** parameter is True. Hence the **encoder_final** parameter is not None.
- The **init_state** function in the decoder is called with the final state of encoder. This final state is used to set the initial state of the decoder.
- In this way the decoder model is conditioned on the output of the encoder model.

5. Implement score in attention.py

```
    def score(self, Q, K):
        """
               (BxT'xd) (BxTxd)  ------->  (BxT'xT)
        """
        L = torch.bmm(self.linear_in(Q), K.transpose(1,2))
        return L  # TODO: Implement using self.linear_in.

```

6. Implement forward in attention.py

```
    # TODO: attn_h is the LHS of Eq. (5) in Luong et al. (2015).
        # Implement it using c, queries, and self.linear_out.
        attn_h = torch.tanh(self.linear_out(torch.cat((c, queries), dim=2)))

```

7. Tests in test_attention.py

```
(venv_assignment) Twishas-MacBook-Pro:code twishanaik$ python3 test_attention.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.078s

OK
```

Figure 5: Passing test_attention.py

8. Train an LSTM language model

```
| epoch   1 |    20/   29 batches | lr 20.00 | ms/batch 146.84 | loss  7.62 | ppl  2037.78
-----------------------------------------------------------------------------------------
| end of epoch   1 | time:  4.93s | valid loss  5.94 | valid ppl   380.12 | valid sqxent   126.35
-----------------------------------------------------------------------------------------
| epoch   2 |    20/   29 batches | lr 20.00 | ms/batch 140.08 | loss  5.90 | ppl   364.17
-----------------------------------------------------------------------------------------
| end of epoch   2 | time:  4.58s | valid loss  5.73 | valid ppl   307.24 | valid sqxent   121.83
-----------------------------------------------------------------------------------------
| epoch   3 |    20/   29 batches | lr 20.00 | ms/batch 122.92 | loss  5.07 | ppl   159.89
-----------------------------------------------------------------------------------------
| end of epoch   3 | time:  4.23s | valid loss  5.69 | valid ppl   295.38 | valid sqxent   120.99
-----------------------------------------------------------------------------------------
| epoch   4 |    20/   29 batches | lr 20.00 | ms/batch 132.94 | loss  4.62 | ppl   101.09
-----------------------------------------------------------------------------------------
| end of epoch   4 | time:  4.33s | valid loss  5.48 | valid ppl   240.42 | valid sqxent   116.61
-----------------------------------------------------------------------------------------
| epoch   5 |    20/   29 batches | lr 20.00 | ms/batch 119.71 | loss  4.57 | ppl    96.85
-----------------------------------------------------------------------------------------
| end of epoch   5 | time:  4.03s | valid loss  5.45 | valid ppl   232.43 | valid sqxent   115.89
-----------------------------------------------------------------------------------------
| epoch   6 |    20/   29 batches | lr 20.00 | ms/batch 118.36 | loss  4.33 | ppl    75.82
-----------------------------------------------------------------------------------------
| end of epoch   6 | time:  4.08s | valid loss  5.36 | valid ppl   213.30 | valid sqxent   114.06
-----------------------------------------------------------------------------------------
| epoch   7 |    20/   29 batches | lr 20.00 | ms/batch 124.48 | loss  4.15 | ppl    63.75
-----------------------------------------------------------------------------------------
| end of epoch   7 | time:  4.21s | valid loss  5.31 | valid ppl   203.00 | valid sqxent   113.01
-----------------------------------------------------------------------------------------
| epoch   8 |    20/   29 batches | lr 20.00 | ms/batch 124.90 | loss  4.10 | ppl    60.55
-----------------------------------------------------------------------------------------
| end of epoch   8 | time:  4.15s | valid loss  5.19 | valid ppl   180.01 | valid sqxent   110.46
-----------------------------------------------------------------------------------------
| epoch   9 |    20/   29 batches | lr 20.00 | ms/batch 116.67 | loss  4.15 | ppl    63.66
-----------------------------------------------------------------------------------------
| end of epoch   9 | time:  4.00s | valid loss  5.11 | valid ppl   166.15 | valid sqxent   108.75
-----------------------------------------------------------------------------------------
| epoch  10 |    20/   29 batches | lr 20.00 | ms/batch 117.22 | loss  3.95 | ppl    51.72
-----------------------------------------------------------------------------------------
| end of epoch  10 | time:  3.98s | valid loss  5.07 | valid ppl   159.12 | valid sqxent   107.83
-----------------------------------------------------------------------------------------
=========================================================================================
| End of training | final loss  5.07 | final ppl   159.12 | final sqxent   107.83
=========================================================================================
```

Figure 6: Training LSTM language model

9. Train the attention model

```
-----------------------------------------------------------------------------------------
| epoch 205 |    20/   29 batches | lr 0.08 | ms/batch 114.59 | loss  1.86 | ppl     6.40
-----------------------------------------------------------------------------------------
| end of epoch 205 | time:  3.86s | valid loss  2.93 | valid ppl    18.77 | valid sqxent    62.37
-----------------------------------------------------------------------------------------
| epoch 206 |    20/   29 batches | lr 0.02 | ms/batch 115.04 | loss  1.87 | ppl     6.49
-----------------------------------------------------------------------------------------
| end of epoch 206 | time:  3.94s | valid loss  2.93 | valid ppl    18.77 | valid sqxent    62.37
-----------------------------------------------------------------------------------------
| epoch 207 |    20/   29 batches | lr 0.00 | ms/batch 124.66 | loss  1.83 | ppl     6.25
-----------------------------------------------------------------------------------------
| end of epoch 207 | time:  4.24s | valid loss  2.93 | valid ppl    18.77 | valid sqxent    62.36
-----------------------------------------------------------------------------------------
```

Figure 7: Training attention model

As observed from the screenshot, after 207 epochs the learning rate becomes zero and thus the model stops learning.
Following was the result after convergence:
— End of training — final loss: 2.93 — **final ppl: 18.71** — final sqxent: 62.31—
Better values can be obtained by changing the random seed.

- The attention weights of the decoder does make sense. They are supposed to show the importance of the words in input while predicting the output sequence.
- A source sentence was chosen from the train_src.txt file.
  Source sentence = "Over a third of children between 9 and 13 are *unk* while 9% of *unk* are overweight , 5% are obese ."
- The trained model was fetched and this source sentence was passed as an input to the model. The encoder encodes the model and while prediction, the decoder returns the attention value at each step.
- The attention was seen to be linearly shifting over the input as we progress. This is because the input and the output sentences in our case are exactly the same.
- Thus, if we have (a,b,c) as source, and current output predicted is (a,b), ideally all our attention should be focusing on the word c as that is the next word we want to predict.