LIGN 167 Problem Set 5

Gaotong Wu

TOTAL POINTS

6.5 / 8

QUESTION 1

1 Problem 1 **1 / 1**

✓ **- 0 pts** **Correct: matrix multiplication between word embedding and attention matrices**

- 1 pts Incorrect description

QUESTION 2

2 Problem 2 **1 / 1**

✓ **- 0 pts** **Correct: outlines all steps to calc z_i**

- 0.2 pts One step missing/incorrect

- 0 pts Incorrect

QUESTION 3

3 Problem 3 **0.5 / 1**

- 0 pts Correct: Q,K, and V corresponds to the Query, Key, and Value representations for each of the words; columns represents the activation of each of the the word w.r.t the columns in the weight matrices.

✓ **- 0.5 pts** **One of the descriptions missing/incorrect/incomplete.**

- 1 pts Incorrect.

QUESTION 4

4 Problem 4 **0.5 / 1**

- 0 pts Correct

- 0.5 pts doesn't mention that query matrix is multiplied by *transpose* of key matrix

✓ **- 0.5 pts** **doesn't mention that softmax is applied row-by-row, not to columns or the entire matrix**

QUESTION 5

5 Problem 5 **1 / 1**

✓ **- 0 pts** **Correct**

- 1 pts doesn't substitute previously defined variables for x, e.g. f(z) = ReLU(Mz)

QUESTION 6

6 Problem 6 **1 / 1**

✓ **- 0 pts** **Correct**

- 1 pts doesn't point out that MLP expects single attention head

QUESTION 7

7 Problem 7 **0.5 / 1**

- 0 pts Correct

- 0.25 pts Minor mistake in proof

✓ **- 0.5 pts** **Major mistake in proof**

- 0.75 pts Insufficient rigor in explanation

- 1 pts Proof with position vectors

- 1 pts Wrong/Missing

Output vectors are more nuanced than a single linear matrix operation - there is a softmax non-linearity and MLP applied as well.

QUESTION 8

8 Problem 8 **1 / 1**

✓ **- 0 pts** **Correct**

- 0.5 pts Missing/Insufficient explanation of position vector encoding

- 0.5 pts Missing explanation of why RNNs do not face the issue

- 1 pts Wrong/Missing

QUESTION 9

9 Additional Comments **0 / 0**

✓ **- 0 pts** **None**

Problem 1

$$\vec{x} = (x_1, x_2, x_3, \ldots, x_n)$$

where the dimension is 1*n

$$W^Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} & q_{15} \ldots q_{1n} \\ q_{21} & q_{22} & q_{23} & q_{24} & q_{25} \ldots q_{2n} \\ q_{31} & q_{32} & q_{33} & q_{34} & q_{35} \ldots q_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ q_{m1} & q_{m2} & q_{m3} & q_{m4} & q_{m5} \ldots q_{mn} \end{bmatrix}$$

$$W^k = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} \ldots k_{1n} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} \ldots k_{2n} \\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} \ldots k_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ k_{m1} & k_{m2} & k_{m3} & k_{m4} & k_{m5} \ldots k_{mn} \end{bmatrix}$$

$$W^v = \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \ldots v_{1n} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \ldots v_{2n} \\ v_{31} & v_{32} & v_{33} & v_{34} & v_{35} \ldots v_{3n} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ v_{m1} & v_{m2} & v_{m3} & v_{m4} & v_{m5} \ldots v_{mn} \end{bmatrix}$$

where the dimension is m*n

$$\vec{q} = \vec{x} * W^Q$$

$$\vec{k} = \vec{x} * W^K$$

$$\vec{v} = \vec{x} * W^V$$

$$\vec{q} = (q_1, q_2, q_3, \ldots, q_m)$$

$$\vec{k} = (k_1, k_2, k_3, \ldots, k_m)$$

$$\vec{v} = (v_1, v_2, v_3, \ldots, v_m)$$

1 Problem 1 **1 / 1**

✓ **- 0 pts** Correct: matrix multiplication between word embedding and attention matrices

- 1 pts Incorrect description

Problem 2

$$\text{step 2: } \overrightarrow{q_1} * \overrightarrow{k_1} = q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}$$

$$\text{step 3: } \frac{\overrightarrow{q_1} * \overrightarrow{k_1}}{\sqrt{64}} = \frac{\overrightarrow{q_1} * \overrightarrow{k_1}}{8} = \frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}}{8}$$

$$\text{step 4: } softmax(\frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}}{8})$$

$$\text{step 5: } \overrightarrow{v_1} * softmax(\frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}}{8})$$

$$= (v_{11}, v_{12}, v_{13}, \ldots, v_{1n}) * softmax(\frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_n * k_n}{8})$$

$$= (v^s_{11}, v^s_{12}, v^s_{13}, \ldots, v^s_{1n})$$

$$\text{other } \vec{v} \text{ are computed in the same way}$$

$$\vec{z} = \overrightarrow{v^s_1} + \overrightarrow{v^s_2} + \overrightarrow{v^s_3} + \ldots + \overrightarrow{v^s_m} = (v^s_{11} + v^s_{21} + v^s_{31} + \ldots + v^s_{m1}, v^s_{12} + v^s_{22} + v^s_{32} + \ldots + v^s_{m2}, \ldots, v^s_{1n} + v^s_{2n} + v^s_{3n} + \ldots + v^s_{mn})$$

$$= (z_1, z_2, z_3, \ldots, z_n)$$

Problem 3

The matrix Q contains information about the query vectors of all the words in the input sentence. Each column is the query vector corresponding to the word in that position. The matrix K contains information about the key vectors of all the words in the input sentence. Each column is the key vector corresponding to the word in that position. The matrix V contains information about the value vectors of all the words in the input sentence. Each column is the value vector corresponding to the word in that position.

Problem 4

In this example we are given, the Q matrix is 2*3 and K^T i is 3*2, the result of their multiplication is a matrix with size 2*2; this matrix is divide by 8, as we did in previous part; then the 2*2 matrix is put into a softmax function; last, the softmax output, which is still a 2*2 matrix, multiply with the 2*3 V matrix. The final result is a 2*3 Z matrix. Each row represents the z vector of each word in a sentence.

2 Problem 2 **1 / 1**

✓ **- 0 pts** **Correct: outlines all steps to calc z_i**

- 0.2 pts One step missing/incorrect

- 0 pts Incorrect

Problem 2

$$\text{step 2: } \overrightarrow{q_1} * \overrightarrow{k_1} = q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}$$

$$\text{step 3: } \frac{\overrightarrow{q_1} * \overrightarrow{k_1}}{\sqrt{64}} = \frac{\overrightarrow{q_1} * \overrightarrow{k_1}}{8} = \frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}}{8}$$

$$\text{step 4: } softmax(\frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}}{8})$$

$$\text{step 5: } \overrightarrow{v_1} * softmax(\frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}}{8})$$

$$= (v_{11}, v_{12}, v_{13}, \ldots, v_{1n}) * softmax(\frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_n * k_n}{8})$$

$$= (v^s_{11}, v^s_{12}, v^s_{13}, \ldots, v^s_{1n})$$

$$\text{other } \vec{v} \text{ are computed in the same way}$$

$$\vec{z} = \overrightarrow{v^s_1} + \overrightarrow{v^s_2} + \overrightarrow{v^s_3} + \ldots + \overrightarrow{v^s_m} = (v^s_{11} + v^s_{21} + v^s_{31} + \ldots + v^s_{m1}, v^s_{12} + v^s_{22} + v^s_{32} + \ldots + v^s_{m2}, \ldots, v^s_{1n} + v^s_{2n} + v^s_{3n} + \ldots + v^s_{mn})$$

$$= (z_1, z_2, z_3, \ldots, z_n)$$

Problem 3

The matrix Q contains information about the query vectors of all the words in the input sentence. Each column is the query vector corresponding to the word in that position. The matrix K contains information about the key vectors of all the words in the input sentence. Each column is the key vector corresponding to the word in that position. The matrix V contains information about the value vectors of all the words in the input sentence. Each column is the value vector corresponding to the word in that position.

Problem 4

In this example we are given, the Q matrix is 2*3 and K^T i is 3*2, the result of their multiplication is a matrix with size 2*2; this matrix is divide by 8, as we did in previous part; then the 2*2 matrix is put into a softmax function; last, the softmax output, which is still a 2*2 matrix, multiply with the 2*3 V matrix. The final result is a 2*3 Z matrix. Each row represents the z vector of each word in a sentence.

3 Problem 3 **0.5 / 1**

- **- 0 pts** Correct: Q,K, and V corresponds to the Query, Key, and Value representations for each of the words; columns represents the activation of each of the the word w.r.t the columns in the weight matrices.
- ✓ **- 0.5 pts One of the descriptions missing/incorrect/incomplete.**
- **- 1 pts** Incorrect.

Problem 2

$$\text{step 2: } \vec{q_1} * \vec{k_1} = q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}$$

$$\text{step 3: } \frac{\vec{q_1} * \vec{k_1}}{\sqrt{64}} = \frac{\vec{q_1} * \vec{k_1}}{8} = \frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}}{8}$$

$$\text{step 4: } softmax(\frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}}{8})$$

$$\text{step 5: } \vec{v_1} * softmax(\frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_{1n} * k_{1n}}{8})$$

$$= (v_{11}, v_{12}, v_{13}, \ldots, v_{1n}) * softmax(\frac{q_{11} * k_{11} + q_{12} * k_{12} + q_{13} * k_{13} + \ldots + q_n * k_n}{8})$$

$$= (v^s_{11}, v^s_{12}, v^s_{13}, \ldots, v^s_{1n})$$

other $\vec{v}$ are computed in the same way

$$\vec{z} = \vec{v^s_1} + \vec{v^s_2} + \vec{v^s_3} + \ldots + \vec{v^s_m} = (v^s_{11} + v^s_{21} + v^s_{31} + \ldots + v^s_{m1}, v^s_{12} + v^s_{22} + v^s_{32} + \ldots + v^s_{m2}, \ldots, v^s_{1n} + v^s_{2n} + v^s_{3n} + \ldots + v^s_{mn})$$

$$= (z_1, z_2, z_3, \ldots, z_n)$$

Problem 3

The matrix Q contains information about the query vectors of all the words in the input sentence. Each column is the query vector corresponding to the word in that position. The matrix K contains information about the key vectors of all the words in the input sentence. Each column is the key vector corresponding to the word in that position. The matrix V contains information about the value vectors of all the words in the input sentence. Each column is the value vector corresponding to the word in that position.

Problem 4

In this example we are given, the Q matrix is 2*3 and K^T i is 3*2, the result of their multiplication is a matrix with size 2*2; this matrix is divide by 8, as we did in previous part; then the 2*2 matrix is put into a softmax function; last, the softmax output, which is still a 2*2 matrix, multiply with the 2*3 V matrix. The final result is a 2*3 Z matrix. Each row represents the z vector of each word in a sentence.

4 Problem 4 **0.5 / 1**

- **- 0 pts** Correct
- **- 0.5 pts** doesn't mention that query matrix is multiplied by *transpose* of key matrix
- ✓ **- 0.5 pts doesn't mention that softmax is applied row-by-row, not to columns or the entire matrix**

Problem 5

$$ReLU = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

When x is a vector input to the MLP and M is a matrix whose column dimensions are consistent with the input x, the multiplication of M and x results in a vector having the same size as x. In this case, the input vector is the resulted z vector above.

$$\vec{z} = (z_1, z_2, z_3, \ldots, z_n)$$

let the resulting vector of M*z be $\vec{m}$

$$MLP = (ReLU(m_1), ReLU(m_2), ReLU(m_3), \ldots, ReLU(m_n))$$

where each element of m is the dot product of $\vec{z}$ and the corresponding column, ReLU function is defined above

This MLP is a ReLU function, which applies non-linearity to the model. As we can see, the previous steps in the self-attention layer are basically linear computations. If this is the model all about, it will be simply a linear regression model, which is not powerful enough to deal with complex texts. Therefore, non-linearity is able to increase the ability of the model to learn something complex. The usage of ReLU function is to avoid vanishing gradient and its gradient is easy to compute.

Problem 6

This process results in 8 sets of Z matrices. However, the feed-forward layer is not able to deal with 8 Z matrices. It can only deal with one Z matrix with each row representing a vector for each word. The problem is that the MLP layer only takes one input vector x instead of 8 vectors, as explained in problem 5. The architecture solves this problem by concatenating 8 Z matrices together and multiplying it with an additional weights matrix WO. The result turns into a single Z matrix and we can feed it into the feed-forward layer.

5 Problem 5 **1 / 1**

✓ **- 0 pts** Correct

- 1 pts doesn't substitute previously defined variables for x, e.g. f(z) = ReLU(Mz)

Problem 5

$$ReLU = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

When x is a vector input to the MLP and M is a matrix whose column dimensions are consistent with the input x, the multiplication of M and x results in a vector having the same size as x. In this case, the input vector is the resulted z vector above.

$$\vec{z} = (z_1, z_2, z_3, \ldots, z_n)$$

let the resulting vector of M*z be $\vec{m}$

$$MLP = (ReLU(m_1), ReLU(m_2), ReLU(m_3), \ldots, ReLU(m_n))$$

where each element of m is the dot product of $\vec{z}$ and the corresponding column, ReLU function is defined above

This MLP is a ReLU function, which applies non-linearity to the model. As we can see, the previous steps in the self-attention layer are basically linear computations. If this is the model all about, it will be simply a linear regression model, which is not powerful enough to deal with complex texts. Therefore, non-linearity is able to increase the ability of the model to learn something complex. The usage of ReLU function is to avoid vanishing gradient and its gradient is easy to compute.

Problem 6

This process results in 8 sets of Z matrices. However, the feed-forward layer is not able to deal with 8 Z matrices. It can only deal with one Z matrix with each row representing a vector for each word. The problem is that the MLP layer only takes one input vector x instead of 8 vectors, as explained in problem 5. The architecture solves this problem by concatenating 8 Z matrices together and multiplying it with an additional weights matrix WO. The result turns into a single Z matrix and we can feed it into the feed-forward layer.

6 Problem 6 **1 / 1**

✓ **- 0 pts** Correct

- 1 pts doesn't point out that MLP expects single attention head

Problem 7

The encoder consists of two layers: self-attention and feed-forward. All the steps in the self-attention are linear computation, as indicated in problems 1 and 2; if it is always the two same vectors: x1 and x2, despite the order of them, being sent into the self-attention layer, the output vectors z1 and z2 are always the same set. Two feed-forward layers are identical and parallel, thus the outputs r1 and r2 are the same set despite the order when the same set of vectors z1 and z2 are sent as input. The whole procedure is not affected by the order of input x1 and x2.

Problem 8

The transformer solves this problem by introducing positional encoding vectors. These vectors follow specific patterns learned from the model and contain information about the position of these words or the distance between these words. Adding the embedding vectors to the positional encoding vectors gives the model a sense of orders of words. This is not a problem for RNN because RNN, unlike Transformer which is parallel computation, takes the word-by-word order as input, which means that the next output is based on the previous output and the current input. Therefore, the architecture itself already takes care of the problem of orders.

7 Problem 7 **0.5 / 1**

- **0 pts** Correct
- **0.25 pts** Minor mistake in proof

✓ **- 0.5 pts Major mistake in proof**

- **0.75 pts** Insufficient rigor in explanation
- **1 pts** Proof with position vectors
- **1 pts** Wrong/Missing

Output vectors are more nuanced than a single linear matrix operation - there is a softmax non-linearity and MLP applied as well.

Problem 7

The encoder consists of two layers: self-attention and feed-forward. All the steps in the self-attention are linear computation, as indicated in problems 1 and 2; if it is always the two same vectors: x1 and x2, despite the order of them, being sent into the self-attention layer, the output vectors z1 and z2 are always the same set. Two feed-forward layers are identical and parallel, thus the outputs r1 and r2 are the same set despite the order when the same set of vectors z1 and z2 are sent as input. The whole procedure is not affected by the order of input x1 and x2.

Problem 8

The transformer solves this problem by introducing positional encoding vectors. These vectors follow specific patterns learned from the model and contain information about the position of these words or the distance between these words. Adding the embedding vectors to the positional encoding vectors gives the model a sense of orders of words. This is not a problem for RNN because RNN, unlike Transformer which is parallel computation, takes the word-by-word order as input, which means that the next output is based on the previous output and the current input. Therefore, the architecture itself already takes care of the problem of orders.

8 Problem 8 1 / 1

✓ **- 0 pts** Correct

- 0.5 pts Missing/Insufficient explanation of position vector encoding

- 0.5 pts Missing explanation of why RNNs do not face the issue

- 1 pts Wrong/Missing

9 Additional Comments 0 / 0

✓ **- 0 pts** None