# DL HW3

Mehdi Jamalkhah

August 2024

## 1 LSTM

### 1.1

Based on this article the models with LSTM and GRU methods have the same accuracy results in sentiment analysis but LSTM is slightly superior in precision, recall and F1-score. From both models, it can be concluded that both are superior in analyzing negative sentiments.

However, it should also be considered that GRU is more efficient, and in this particular situation, we can overlook this minor superiority and opt for GRU instead.

### 1.2

No, LSTM and GRU are not parallelizable very well. They actually have an O(T) serial component, where T is the length of the sequence. As a result, GPUs cannot make significant progress on these models. Instead, we should choose a more parallelizable alternative, which is the Transformer architecture.

### 1.3

$$c^{<t>} = c^{<t-1>} * \sigma(W_f \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} + b_f) + tanh(W_c \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} + b_c) * \sigma(W_u \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} + b_u)$$

$$a^{<t>} = tanh(c^{<t>}) * \sigma(W_o \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} + b_o)$$

$$\hat{y}^{<t>} = softmax(W_y(tanh(C^{<t>}) * \sigma(W_o \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} + b_o)) + b_y)$$

### 1.4

$$\frac{\partial L^{<t>}}{\partial a^{<t>}} = \frac{\partial L^{<t>}}{\partial z^{<t>}} \frac{\partial z^{<t>}}{\partial a^{<t>}} = (\hat{y}^{<t>} - y^{<t>})^T W_y^T$$

$$\frac{\partial L}{\partial a^{<t>}} = \frac{\partial L^{<t>}}{\partial a^{<t>}} + da_{next} = (\hat{y}^{<t>} - y^{<t>})^T W_y^T + da_{next}$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{W_y}} = \frac{\partial L^{<t>}}{\partial W_y} = \frac{\partial L^{<t>}}{\partial z^{<t>}} \frac{\partial z^{<t>}}{\partial W_y} = (\hat{y}^{<t>} - y^{<t>}) a^{<t>T}$$

$$\frac{\partial \mathbf{L}}{\partial \mathbf{b_y}} = \frac{\partial L^{<t>}}{\partial b_y} = \frac{\partial L^{<t>}}{\partial z^{<t>}} \frac{\partial z^{<t>}}{\partial b_y} = (\hat{y}^{<t>} - y^{<t>})$$

$$\frac{\partial L}{\partial \Gamma_o^{<t>}} = \frac{\partial L}{\partial a^{<t>}} \frac{\partial a^{<t>}}{\partial \Gamma_o^{<t>}} = \frac{\partial L}{\partial a^{<t>}} * tanh(c^{<t>})$$
$$\frac{\partial}{\partial L}\left(\gamma_o^{<t>}\right) = \frac{\partial L}{\partial \Gamma_o^{<t>}} \frac{\partial \Gamma_o^{<t>}}{\partial \gamma_o^{<t>}} = \frac{\partial L}{\partial \Gamma_o^{<t>}} * \Gamma_o^{<t>} * (1 - \Gamma_o^{<t>})$$
$$\frac{\partial \mathbf{L}}{\partial \mathbf{W_o}} = \frac{\partial L}{\partial \gamma_o^{<t>}} \frac{\partial \gamma_o^{<t>}}{\partial W_o} = \frac{\partial L}{\partial \gamma_o^{<t>}} \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}^T$$
$$\frac{\partial \mathbf{L}}{\partial \mathbf{b_o}} = \frac{\partial L}{\partial \gamma_o^{<t>}} \frac{\partial \gamma_o^{<t>}}{\partial b_o} = \frac{\partial L}{\partial \gamma_o^{<t>}}$$

$$\frac{\partial L}{\partial c^{<t>}} = \frac{\partial L}{\partial a^{<t>}} \frac{\partial a^{<t>}}{\partial c^{<t>}} + dc_{next} = \frac{\partial L}{\partial a^{<t>}} * \Gamma_o^{<t>} * (1 - tanh^2(c^{<t>})) + dc_{next}$$
$$\frac{\partial L}{\partial \hat{c}^{<t>}} = \frac{\partial L}{\partial c^{<t>}} \frac{\partial c^{<t>}}{\partial \hat{c}^{<t>}} = \frac{\partial L}{\partial c^{<t>}} * \Gamma_u^{<t>}$$
$$\frac{\partial L}{\partial p\hat{c}^{<t>}} = \frac{\partial L}{\partial \hat{c}^{<t>}} \frac{\partial \hat{c}^{<t>}}{\partial p\hat{c}^{<t>}} = \frac{\partial L}{\partial \hat{c}^{<t>}} * (1 - p\hat{c}^{<t>2})$$
$$\frac{\partial \mathbf{L}}{\partial \mathbf{W_c}} = \frac{\partial L}{\partial p\hat{c}^{<t>}} \frac{\partial p\hat{c}^{<t>}}{\partial W_c} = \frac{\partial L}{\partial p\hat{c}^{<t>}} \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}^T$$
$$\frac{\partial \mathbf{L}}{\partial \mathbf{b_c}} = \frac{\partial L}{\partial p\hat{c}^{<t>}}$$

$$\frac{\partial L}{\partial \Gamma_u^{<t>}} = \frac{\partial L}{\partial c^{<t>}} \frac{\partial c^{<t>}}{\partial \Gamma_u^{<t>}} = \frac{\partial L}{\partial c^{<t>}} * \hat{c}^{<t>}$$
$$\frac{\partial L}{\partial \gamma_u^{<t>}} = \frac{\partial L}{\partial \Gamma_u^{<t>}} \frac{\partial \Gamma_u^{<t>}}{\partial \gamma_u^{<t>}} = \frac{\partial L}{\partial \Gamma_u^{<t>}} * \Gamma_u^{<t>} * (1 - \Gamma_u^{<t>})$$
$$\frac{\partial \mathbf{L}}{\partial \mathbf{W_u}} = \frac{\partial L}{\partial \gamma_u^{<t>}} \frac{\partial \gamma_u^{<t>}}{\partial W_u} = \frac{\partial L}{\partial \gamma_u^{<t>}} \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}^T$$
$$\frac{\partial \mathbf{L}}{\partial \mathbf{b_u}} = \frac{\partial L}{\partial \gamma_u^{<t>}}$$

$$\frac{\partial L}{\partial \Gamma_f^{<t>}} = \frac{\partial L}{\partial c^{<t>}} \frac{\partial c^{<t>}}{\partial \Gamma_f^{<t>}} = \frac{\partial L}{\partial c^{<t>}} * c^{<t-1>}$$
$$\frac{\partial L}{\partial \gamma_f^{<t>}} = \frac{\partial L}{\partial \Gamma_f^{<t>}} \frac{\partial \Gamma_f^{<t>}}{\partial \gamma_f^{<t>}} = \frac{\partial L}{\partial \Gamma_f^{<t>}} * \Gamma_f^{<t>} * (1 - \Gamma_f^{<t>})$$
$$\frac{\partial \mathbf{L}}{\partial \mathbf{W_f}} = \frac{\partial L}{\partial \gamma_f^{<t>}} \frac{\partial \gamma^{<t>}}{\partial W_f} = \frac{\partial L}{\partial \gamma_f^{<t>}} \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}^T$$
$$\frac{\partial \mathbf{L}}{\partial \mathbf{b_f}} = \frac{\partial L}{\partial \gamma_f^{<t>}}$$

# 2 Transformer Parameters

## 2.1

Let's define all hyper-parameters:

$$\begin{aligned}
&\text{vocab_size} = 30000\\
&\text{n_encoder} = 12\\
&\text{n_decoder} = 8\\
&d = 1024\\
&T = 2048\\
&d_h = 768\\
&d_{ff} = 512\\
&B = 32\\
&h = 4
\end{aligned}$$

Dimention of all signals:

$$\begin{aligned}
&\text{Source, target: } B \times T \times \text{vocab_size}\\
&\text{others: } B \times T \times d
\end{aligned}$$

## 2.2

$$\begin{aligned}
&\text{Embeding: vocab_size} \times d = 30,720,000\\
&\text{Multi_head: } d \times \frac{d_h}{h} \times 3 \times h + d_h \times 3 + d_h \times d + d = 3,149,056\\
&\text{Norm: } 2 \times d = 2048\\
&\text{FFN: } 2 \times d \times dff + d + d_{ff} = 1,050,112
\end{aligned}$$

$$\begin{aligned}
\#\text{ params} &= \text{n_encoder} \times (\text{Multi_head} + 2\text{Norm} + \text{FFN})\\
&\quad + \text{n_decoder} \times (2\text{Multi_head} + 3\text{Norm} + \text{FFN})\\
&\quad + \text{Embeding}\\
&= 50,439,168 + 58,834,944 + 30,720,000\\
&= 139,994,112
\end{aligned}$$

# 3 Self-Attention

## 3.1

$$a_{nm} = \frac{exp(x_n^T x_m)}{\sum_{m'=1}^{N} x_n^T x_{m'}} = \begin{cases} \frac{0}{\sum_{m'=1}^{N} x_n^T x_{m'}} = 0 & n \neq m \\ \frac{exp(x_n^T x_m)}{exp(x_n^T x_m)} = 1 & n = m \end{cases}$$

$$\Rightarrow y_n = \sum_{m=1}^{N} a_{nm} x_m = x_n$$

### 3.2

$$\begin{aligned}
\mathbb{E}[(a^T b)^2] &= \mathbb{E}[(a^T b)(a^T b)] \\
&= \mathbb{E}[(a_1 b_1 + \cdots + a_d b_d)(a_1 b_1 + \cdots + a_d b_d)] \\
&= \mathbb{E}[\sum_{i=1}^{D}\sum_{j=1}^{D} a_i b_i a_j b_j] \\
&= \sum_{i=1}^{D}\sum_{j=1}^{D} \mathbb{E}[a_i b_i a_j b_j] \\
&= \sum_{i=1}^{D}\sum_{j=1}^{D} \mathbb{E}[a_i a_j]\mathbb{E}[b_i b_j] \\
&= \sum_{i=1}^{D}\sum_{j=1,j\neq i}^{D} \mathbb{E}[a_i a_j]\mathbb{E}[b_i b_j] + \sum_{i=1}^{D} \mathbb{E}[a_i^2]\mathbb{E}[b_i^2] \\
&= 0 + D(1) = D
\end{aligned}$$

### 3.3

$$\begin{aligned}
Y(X) &= [H_1, \ldots, H_H] W^{(o)} \\
&= [H_1, \ldots, H_H] \begin{bmatrix} W_1^{(0)} \\ \vdots \\ W_H^{(o)} \end{bmatrix} \\
&= \sum_{h=1}^{H} H_h W_h^{(o)} \\
&= \sum_{h=1}^{H} softmax[\frac{Q_h K_h^T}{\sqrt{D_{k_h}}}] V_h W_h^{(o)} \\
&= \sum_{h=1}^{H} softmax[\frac{Q_h K_h^T}{\sqrt{D_{k_h}}}] X \underbrace{W_h^{(v)} W_h^{(o)}}_{W^{(h)}} \\
&= \sum_{h=1}^{H} softmax[\frac{Q_h K_h^T}{\sqrt{D_{k_h}}}] X W^{(h)}
\end{aligned}$$

# 4 Rotary and ALiBi PE

## 4.1

Learned approaches can optimize position encoding, while unlearned PE cannot. and this feature makes the learned PE more powerful. on the other hand, we have absolute and relative PE, the first one is not robust; for example, adding an adverb to the beginning of a sentence can change the result of the model while the meaning is the same as before. relative PE solves this problem and for calculating PE only use the relative distances between tokens.

**Disadvantages:**

1. Learned: Extra memory and computation
2. Unlearned: Inductive bias that may not be proper for our task
3. Absolute: Unrobust to little changes in the absolute position of words, discard the fact that relative position is important in natural languages
4. Relative: Computationally inefficient. During inference, researchers like to use a method called KV cache which helps to reduce the inference speed. but in this method the embeddings for each token change for each new time step.

## 4.2

Rotary PE is a relatively unlearned PE. So we talk about the disadvantages of these categories. while it is a relative PE but does not Decrease the inference speed, because this method finds a solution to multiply each token by a constant that results in attention is only dependent on relative distance. so PE added at first only one time, not for calculating the attention of each token. The second improvment about inductive bias will be explained in next subsection.

## 4.3

We can interpret the length of the vector as the embedding of the word and its angle as the embedding of position. In this way make this information vertical to each other.

## 4.4

Suppose the dimension of word embeddings is 4.

**Tokens:**

| "sharif" | "university" | "of" | "technology" |
|---|---|---|---|

**Embeding:**

| "sharif" | "university" | "of" | "technology" |
|---|---|---|---|
| $\begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}$ | $\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$ | $\begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix}$ | $\begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{bmatrix}$ |

**Sinusoidal:**

| | | | |
|---|---|---|---|
| $\begin{bmatrix} s_1 + sin(\theta_1) \\ s_2 + cos(\theta_1) \\ s_3 + sin(\theta_2) \\ s_4 + cos(\theta_2) \end{bmatrix}$ | $\begin{bmatrix} u_1 + sin(2\theta_1) \\ u_2 + cos(2\theta_1) \\ u_3 + sin(2\theta_2) \\ u_4 + cos(2\theta_2) \end{bmatrix}$ | $\begin{bmatrix} o_1 + sin(3\theta_1) \\ o_2 + cos(3\theta_1) \\ o_3 + sin(3\theta_2) \\ o_4 + cos(3\theta_2) \end{bmatrix}$ | $\begin{bmatrix} t_1 + sin(4\theta_1) \\ t_2 + cos(4\theta_1) \\ t_3 + sin(4\theta_2) \\ t_4 + cos(4\theta_2) \end{bmatrix}$ |

**Rotary:**

| | | | |
|---|---|---|---|
| $\begin{bmatrix} s_1cos(\theta_1) - s_2sin(\theta_1) \\ s_1sin(\theta_1) + s_2cos(\theta_1) \\ s_3cos(\theta_2) - s_4sin(\theta_2) \\ s_3sin(\theta_2) + s_4cos(\theta_2) \end{bmatrix}$ | $\begin{bmatrix} u_1cos(2\theta_1) - u_2sin(2\theta_1) \\ u_1sin(2\theta_1) + u_2cos(2\theta_1) \\ u_3cos(2\theta_2) - u_4sin(2\theta_2) \\ u_3sin(2\theta_2) + u_4cos(2\theta_2) \end{bmatrix}$ | $\begin{bmatrix} o_1cos(3\theta_1) - o_2sin(3\theta_1) \\ o_1sin(3\theta_1) + o_2cos(3\theta_1) \\ o_3cos(3\theta_2) - o_4sin(3\theta_2) \\ o_3sin(3\theta_2) + o_4cos(3\theta_2) \end{bmatrix}$ | $\begin{bmatrix} t_1cos(4\theta_1) - t_2sin(4\theta_1) \\ t_1sin(4\theta_1) + t_2cos(4\theta_1) \\ t_3cos(4\theta_2) - t_4sin(4\theta_2) \\ t_3sin(4\theta_2) + t_4cos(4\theta_2) \end{bmatrix}$ |

In Sinusoidal PE each value of word embedding will be added by a sin/cos part, but in Rotary they will be multiplied by each and combine to make new elements of embedding vectors.

### 4.5

The position embedding will be incorporated into the attention matrix. Additionally, the value of the biases is related to the relative position of the tokens. This is because, as evident in the bias matrix, the values decrease as the distance from the diagonal of the matrix increases. Furthermore, the diagonal of the matrix is zero, representing the distance of each element from itself.

Scalar m is a head-specific slope fixed before training. More accuratly, ALiBi penalizes attention scores between distant query-key pairs, with the penalty increasing as the distance between a key and a query grows. The different heads increase their penalties at different rates, depending on the slope magnitude.

## 5

### 5.1

Following an example makes the proof easier:

$$\frac{QK^T}{\sqrt{D_k}} = \frac{1}{\sqrt{D_k}} XW_q W_k^T X^T = XAX^T$$
$$= \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{bmatrix}$$

$$= \begin{bmatrix} ax_{11}x_{11} + bx_{12}x_{11} + cx_{11}x_{12} + dx_{12}x_{12} & ax_{21}x_{11} + bx_{12}x_{21} + cx_{21}x_{12} + dx_{22}x_{12} \\ ax_{11}x_{21} + bx_{12}x_{21} + cx_{11}x_{21} + dx_{12}x_{22} & ax_{21}x_{21} + bx_{22}x_{21} + cx_{21}x_{22} + dx_{22}x_{22} \end{bmatrix}$$

$$= \begin{bmatrix} x_{11}x_{11} & x_{11}x_{12} & x_{11}x_{21} & x_{11}x_{22} \\ x_{12}x_{11} & x_{12}x_{12} & x_{12}x_{21} & x_{12}x_{22} \\ x_{21}x_{11} & x_{21}x_{12} & x_{21}x_{21} & x_{21}x_{22} \\ x_{22}x_{11} & x_{22}x_{12} & x_{22}x_{21} & x_{22}x_{22} \end{bmatrix} M_{ND,ND,N,N}$$

### 5.2

The $M$ matrix as can be seen in above has $O(N^4D^2)$ parameters

### 5.3

$$M_{:,:,1,1} = \begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad M_{:,:,1,2} = \begin{bmatrix} 0 & 0 & a & b \\ 0 & 0 & c & d \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_{:,:,2,1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ a & b & 0 & 0 \\ c & d & 0 & 0 \end{bmatrix} \qquad M_{:,:,2,2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix}$$

### 5.4

The attention process involves multiplying matrices and adding values based on those multiplications. Permuting the input will also permute the added values in the same way, resulting in the same final output, since adding is equivarient.