

Attention is all you need

1. Introduction

LSTM and RNNs factor computation along the symbol positions of the input and the output sequence.

$$\text{generator: } h_t = \text{fct}(h_{t-1})$$

↳ includes parallelization (becomes critical with long sequences) due to the sequential computation.

2. Background

Models such as ENGPU, ByteNet and ConvS2S use CNN as basic building block and aim at reducing sequential computation.

- difficult to learn long range dependences (linearly for ConvS2S and log for ByteNet)

↳ Transformers reduces it to constant number of operations.

3. Model Architecture

Most competitive neural sequence models have an encoder-decoder

structure. The encoder maps an input sequence of symbol representations (z_1, \dots, z_n) to continuous representations $z = (z_1, \dots, z_n)$. Given z , the decoder generates (y_1, \dots, y_n) . At each step the model is auto-sequence.

3.1. Encoder Decoder Stacks

Encoder: $N=6$ identical layers

- each layer has 2 sub-layers:

- Multi-Head Self Attention
- Position-wise FFN.

$$\text{layerNorm}(x - \text{SubLayer}(x))$$

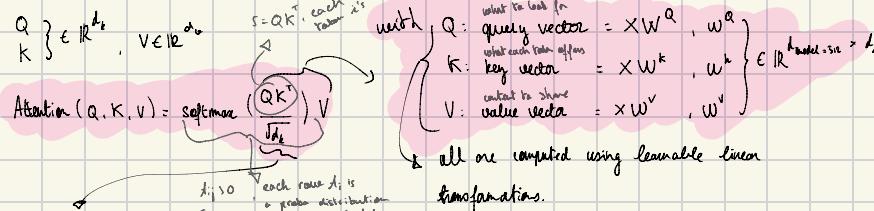
$$4n \text{ outputs } \in \mathbb{R}^{4 \times d_k}$$

NN

3.2. Attention

3.2.1. Scaled Dot-Product Attention

$$Q \in \mathbb{R}^{d_k}, V \in \mathbb{R}^{d_v}$$



why scaling?
if d_k is too large, $q_i^T k_j$ tends to have a large variance → softmax saturates

$$\text{If } q_i, k_j \sim \mathcal{U}(0, 1), \text{ then } \mathbb{E}[q_i k_j] = 0$$

$$\text{but } \text{Var}(q_i k_j) = d_k$$

Deriving gradients of Attention

$$\text{We need: } \frac{\partial L}{\partial Q}, \frac{\partial L}{\partial K}, \frac{\partial L}{\partial V}$$

$$\text{L} = \frac{\partial L}{\partial X} = \frac{\partial L}{\partial Z} \cdot \frac{\partial Z}{\partial A} \cdot \frac{\partial A}{\partial Q, K} \cdot \frac{\partial Q, K}{\partial X}$$

$$\frac{\partial L}{\partial V} = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)^T \cdot \frac{\partial L}{\partial \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V} \quad \text{with } L = -\frac{1}{n} \sum_{i=1}^n y_i \log p_i$$

$$\frac{\partial L}{\partial A} = \left(\frac{\partial L}{\partial Z} \otimes V^T \right) \circ \text{softmax}'(A) \quad \frac{\partial L}{\partial W^Q} = X^T \otimes \frac{\partial L}{\partial V}$$

$$\frac{\partial L}{\partial Q} = \frac{1}{d_k} \left(\frac{\partial L}{\partial A} \right) K \quad \rightarrow \quad \frac{\partial L}{\partial W^Q} = X^T \otimes \frac{\partial L}{\partial Q}$$

$$\frac{\partial L}{\partial K} = \frac{1}{d_k} \left(\frac{\partial L}{\partial A} \right)^T Q \quad \rightarrow \quad \frac{\partial L}{\partial W^K} = X^T \otimes \frac{\partial L}{\partial K}$$

What it does?

- weigh the importance of different parts of a sequence

Decoder: $n=6$ layers

- 3 sub-layers: Multi-Head Attention

- position-wise FFN
- MH Self Attention (with mask)

B.1.1. Layer Normalization (19.11.07.013)

- normalize the distribution of intermediate layers → reducing vanishing/exploding gradients

$$\text{PROOF: } y_i = \gamma_i \left[\frac{x_i - \bar{x}_i}{\sqrt{\sigma_x^2 + \epsilon}} \right] + \beta_i, \text{ with } \gamma, \beta \in \mathbb{R}^d \text{ learnable.}$$

⇒ y_i has a bounded mean and variance

$\text{Var}[y_i]$ is more consistent

- enables a faster training and convergence

PROOF

In convex fcts, GD satisfies:

$$f(x^{(t)}) - f(x^*) \leq \frac{C}{t} \quad \text{with Training speed depending on C.}$$

$$\leq \frac{L}{2t} \|x^{(t)} - x^*\|^2$$

with L : gradient Lipschitz const
- how rapidly the gradient changes
 $\|x^{(0)} - x^*\|$: the initial distance from the optimum.

Longer train

given $X \in \mathbb{R}^d$, layerNorm re-centers and re-scales X as:

$$\hat{X} = g @ N(X) + b, \quad N(X) = \frac{X - \mu}{\sigma}, \quad \mu = \frac{1}{d} \sum_{i=1}^d x_i$$

$$\sigma = \sqrt{\frac{1}{d} \sum_{i=1}^d (x_i - \mu)^2}$$

layerNorm(x) = $\gamma \hat{X} + \beta$, where $\gamma, \beta \in \mathbb{R}^d$ are learnable params

which were present to cause overfitting, bc they are learned from the training set and they ignore the testing data distribution. 19.11.09.013

shows that layerNorm offers lower training loss but higher validation loss than w/o γ, β .

The solution: AdamNorm

def $g = d(x) = \frac{x - \mu}{\sigma}$ be the normalized vector.

def $\phi(g)$ be a function replacing the γ and β :

$z = \phi(g) @ g = \phi(d(x)) @ d(x)$ where

z is the output of AdamNorm.

3.3 - Position Wise Feed-Forward Networks

Let's remember that Attention is a linear process: Q, K, V are linear projections of X.

→ FNN introduces non-linearity letting the model transform each token representation independently, enables deep hierarchical representations and makes the model capable of learning complex mappings.

$$FFN(x) = \sigma(xW_1 + b_1)W_2 + b_2$$

↳ (Gelu, ReLU, max, sigmoid, etc...)

Let $z_1 = xW_1 + b_1$, $z_2 = \sigma(z_1)W_2 + b_2$

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial z_1} \cdot \mathbf{1}^T \in \mathbb{R}^{d_{model} \times d_H}$$

with $L = -\sum_i y_i \log(\hat{y}_i)$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_1} \in \mathbb{R}^{d_H}$$

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial z_2} \cdot \mathbf{1}^T \in \mathbb{R}^{d_H \times d_H}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2} \in \mathbb{R}^{d_H}$$

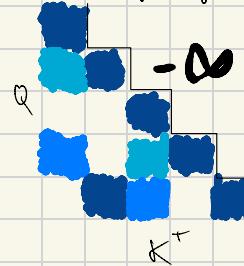
Masked Multi-head Attention

- ensures that the model can attend

to tokens o_{i-1} when predicting token i .

to not leak any info.

How are you doing?



How

are

you

doing

?

→ -0 becomes 0 after softmax,
so the model won't give any
importance to those tokens.

Cross-Attention

- integrates and aligns information from different sequences.

- takes as an input the src and target and capture their dependencies.

- Q from target, K and V from S

- example with translation: aligns source and target words. Generates the translation word by word.

$$Q \in \mathbb{R}^{n \times d_k} = X_{dec} W^Q$$

$$V, K \in \mathbb{R}^{m \times d_h}$$

$$V = X_{enc} W^K$$

$$K = X_{enc} W^V$$

$$\text{CrossAttention}(X_{enc}, X_{dec}) = \text{softmax}\left(\frac{(X_{dec} W^Q @ X_{enc} W^k^T)}{\sqrt{d_k}}\right) X_{enc} W^V$$