

Attention Is All You Need.

Abstract

1. Introduction

2. Background

3. Model Architecture

3.1 Encoder & Decoder Stacks

3.2 Attention

3.2.1 Scaled Dot-Product Attention

3.2.2 Multi-Head Attention

3.2.3 Applications of Attention in our Model

3.3 Position-wise Feed-Forward Networks

3.4 Embeddings and Softmax

3.5 Positional Encoding

4. Why Self-Attention

5. Training

5.1 Training Data and Batching

5.2 Hardware and Schedule

5.3 Optimizer

5.4 Regularization

6. Results

6.1 Machine Translation

6.2 Model Variations

6.3 English Constituency Parsing

7. Conclusion

Abstract

- Simple network architecture ... Transformer
- based solely on attention mechanisms!
- Experiments (machine translation):
 - superior in quality
 - more parallelizable
 - require significantly less time to train
- WMT 2014, best BLEU
- generalizes well!
 - ^{*}! Constituency parsing

1. Introduction

Recurrent models

- typically factor computation along the symbol position of the input and output sequences.
- Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t , as
 - a function of previous hidden state h_{t-1}
 - input for position t .



preclude parallelization within training examples,
critical at longer sequence lengths!

(improvements: factorization tricks & conditional computation)

- Attention mechanism
 - : allow modeling of dependencies without regard to their distance in the input or output sentences.

Transformer

- rely entirely on an attention mechanism to draw global dependencies between input & output
- more parallelization
- SOTA in translation quality
- 12 hrs training with 8 P100 GPUs.

2. Background

- to reduce sequential computation ; CNN

- Extended Neural GPU

- ByteNet

- ConvS2S



the number of operations required to relate signals

from two arbitrary input or output positions grows in the distance between positions.

- linearly for ConvS2S

- logarithmically for ByteNet



* Transformer *

- reduced to a constant number of operations,

- reduced effective resolution due to averaging attention-weighted positions



- Counteract with Multi-Head Attention

- Self-Attention (= intra-attention)

: attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.

- 1st transduction model relying entirely on self-attention

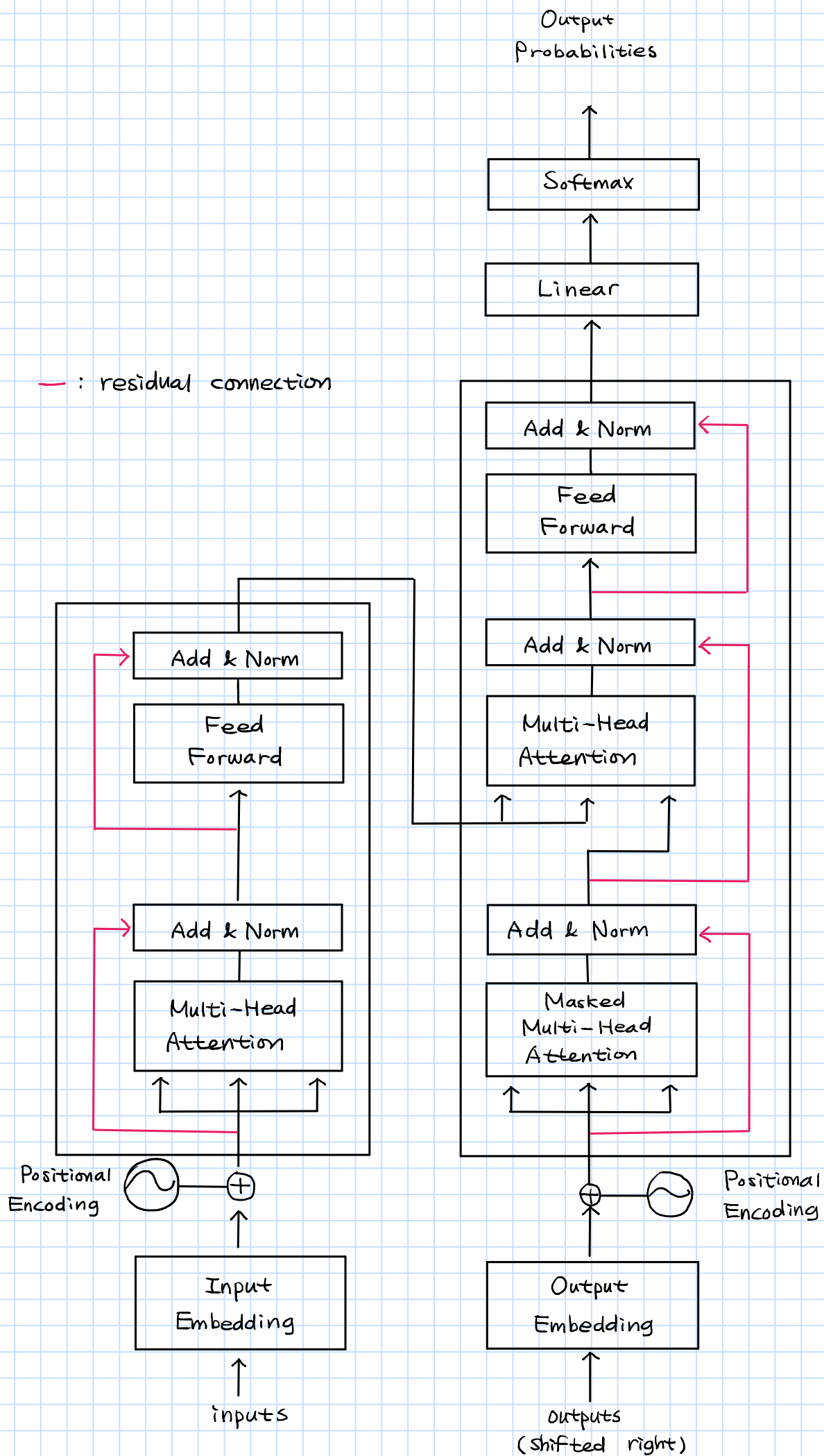
to compute representations of its input & output without using sequence-aligned RNN or convolution.

3. Model Architecture

- Encoder - decoder structure
- Encoder: maps an input sequence of symbol representations (x_1, \dots, x_n) to a sequence of continuous representations $z = (z_1, \dots, z_n)$

↓

- Decoder: generates an output sequence (y_1, \dots, y_m) of symbols one element at a time.
- auto-regressive at each step.
(consuming previously generated symbols as additional input when generating the next)
- stacked self-attention
- point-wise, fully connected layers for both encoder & decoder



3.1 Encoder & Decoder Stacks

- Encoder

- stack of $N=6$ identical layers
- 1st: multi-head self-attention mechanism
- 2nd: position-wise fully connected feed-forward network
- residual connection around each of the two sub-layers, followed by layer normalization.
- the output of each sub-layer is $\text{LayerNorm}(x + \text{Sublayer}(x))$
function implemented by the sub-layer itself.
- $d_{\text{model}} = 512$

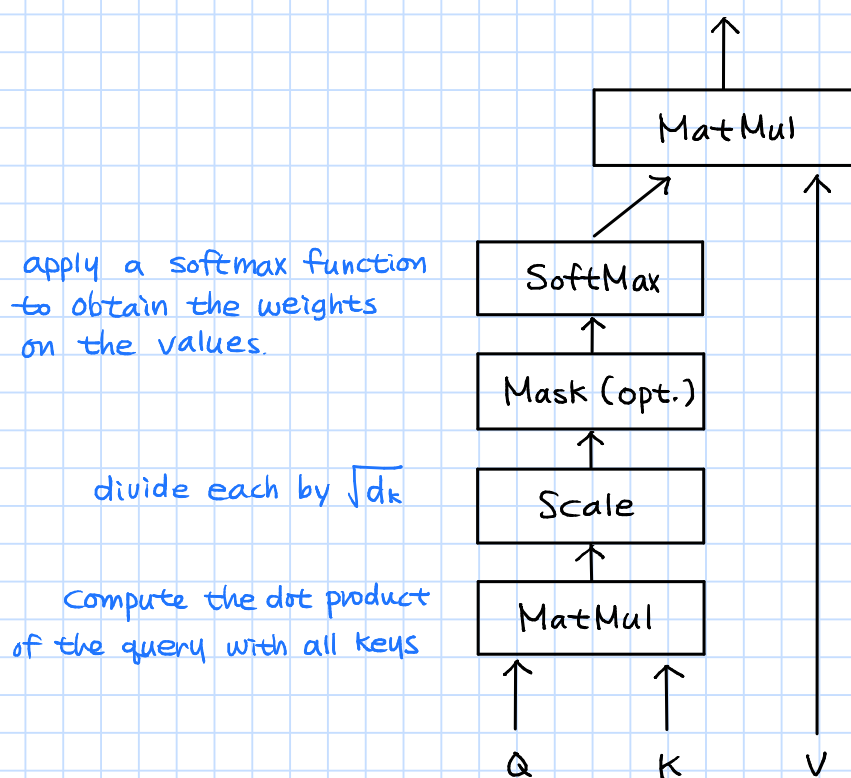
- Decoder

- stack of $N=6$ identical layers
- masking: prevent positions from attending to subsequent positions.

3.2 Attention

- mapping a query & a set of key-value pairs to an output
(Q, K, V , output are all vectors)
- output: weighted sum of the values
- the weight assigned to each value is computed by
a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention



- input: queries, keys of dimension d_k , values of dimension d_v
- In practice, compute attention function simultaneously,
packed together into a matrix $Q, K(\text{keys}), V(\text{Values})$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

2 most commonly used attention functions:

- ① additive attention
- ② dot-product (multiplicative) attention

① additive attention

- computes compatibility function using a feed-forward network with a single hidden layer.

② dot-product attention

- much faster, space-efficient in practice
- for large values of d_k , the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients.

↓

scale the dot product by $\frac{1}{\sqrt{d_k}}$

*1. Constituency parsing

: 문장이 구 단위를 묶어가면서 구조를 이루는 방법
어순이 고정적인 영어에서 쓰임.