[redacted] Transformers: Part 1

[redacted]

[redacted]

Pre-requisites

- Familiar with linear algebra
- Familiar with basic DL

You'll leave these [redacted] understanding transformers

Getting Started with Google BERT

Build and train state-of-the-art natural language processing models using BERT

Sudharsan Ravichandiran

Overview

1. Context
2. The intuition + architecture
3. Encoder block
4. Self-attention
5. Multi-head attention
6. Positional encoding
7. Feedforward layer
8. Add and norm component
9. Encoder step-by-step

RNN-LSTM
ARE GREAT FOR
SEQUENTIAL DATA


RNN-LSTM
ARE GREAT FOR
SEQUENTIAL DATA
THEY STRUGGLE
WITH LONG-TERM
DEPENDENCIES
imgflip.com

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after

Transformers' applications

- NLP
- Image processing
- Large language models
- Generative AI
- Generative music
- …

Transformers' applications

ChatGPT

Transformers’ applications

The basics

- Sequential data

The basics

- Sequential data
- Capture long-term dependencies

The basics

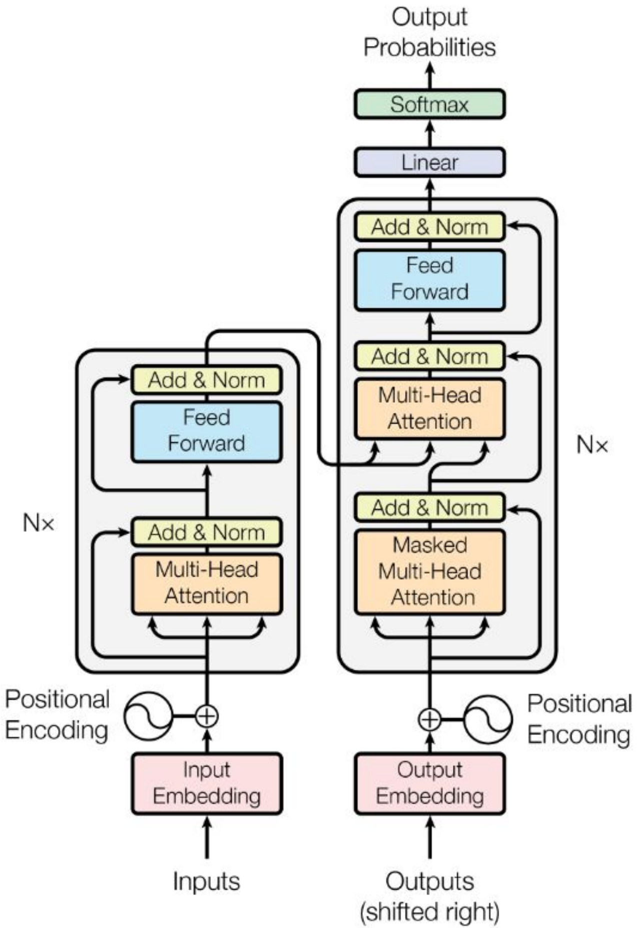
- Sequential data
- Capture long-term dependencies
- Get rid of recurrence

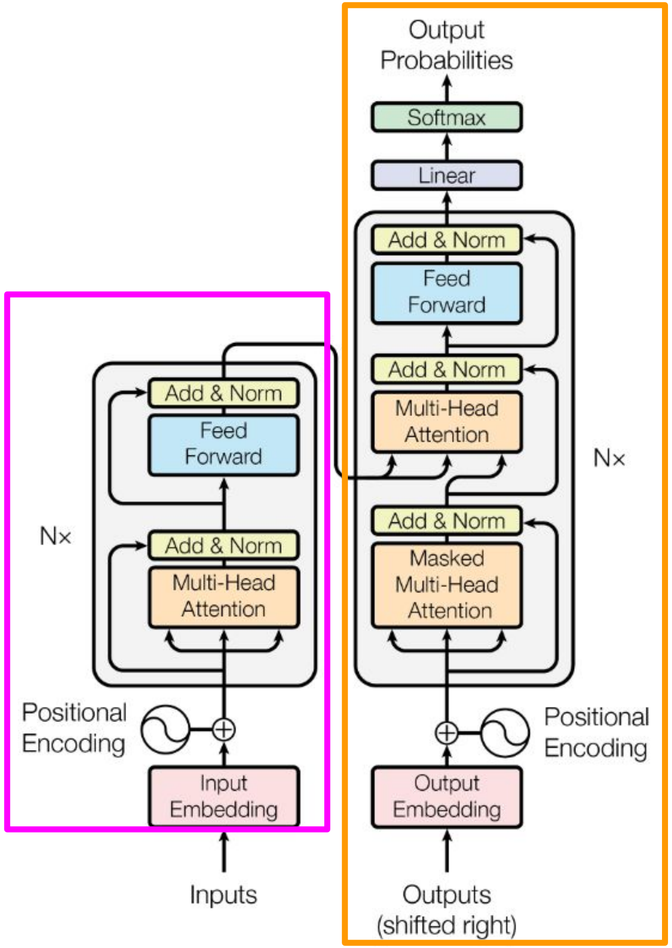
The basics

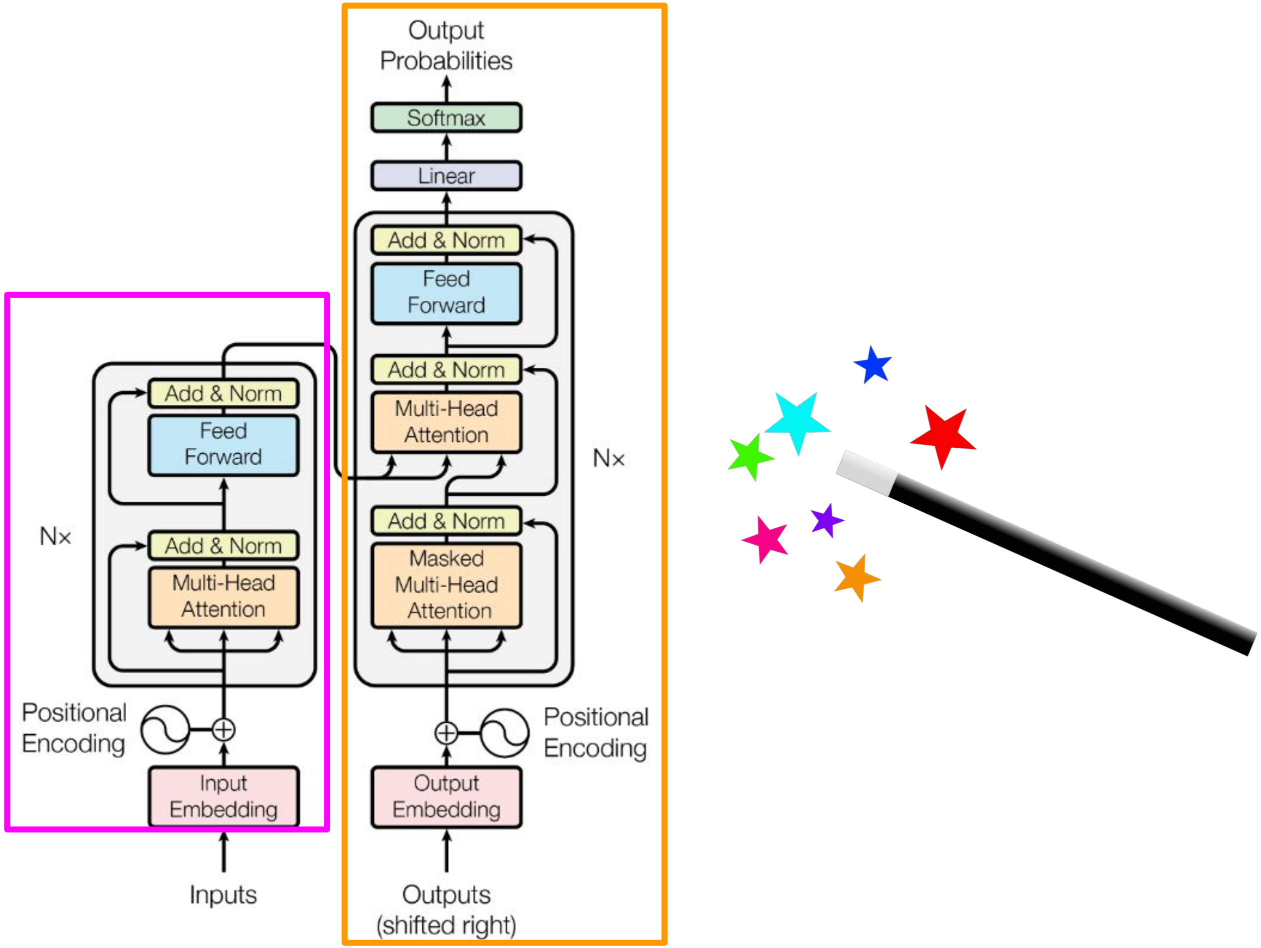
- Sequential data
- Capture long-term dependencies
- Get rid of recurrence
- Self-attention mechanism

The basics

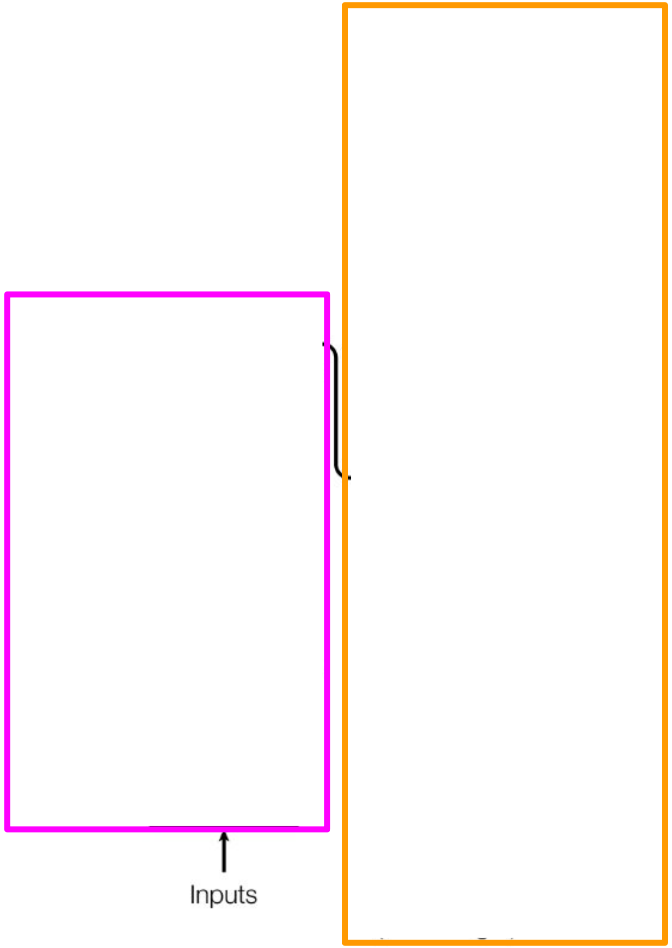
- Sequential data
- Capture long-term dependencies
- Get rid of recurrence
- Self-attention mechanism

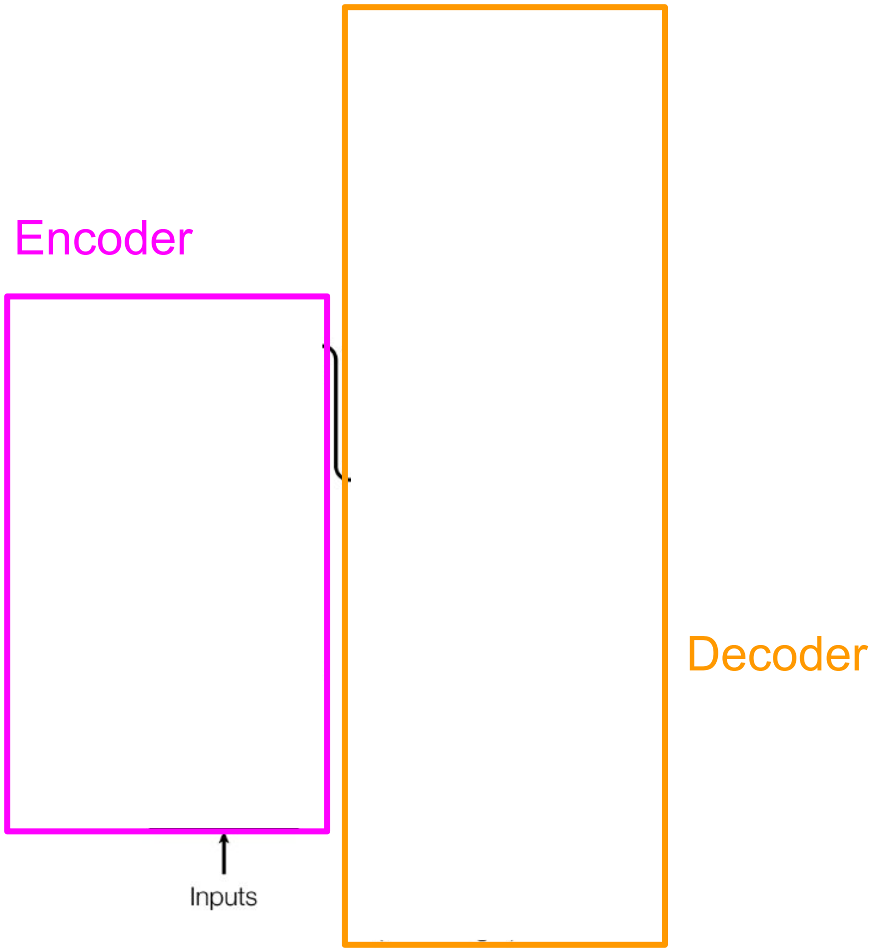

Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Output
Embedding
Positional
Encoding
Inputs
Outputs
(shifted right)



Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Positional
Encoding
Output
Embedding
Outputs
(shifted right)
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs


Inputs



Encoder
Decoder
Inputs

The intuition

1. Feed sentence to encoder

The intuition

1. Feed sentence to encoder
2. Encoder outputs representation of sentence

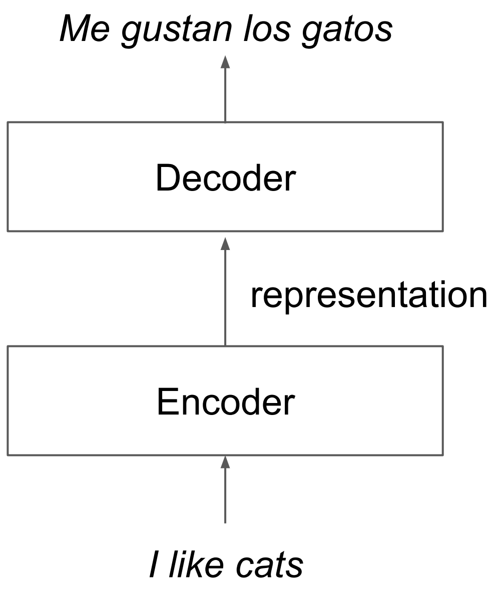
The intuition

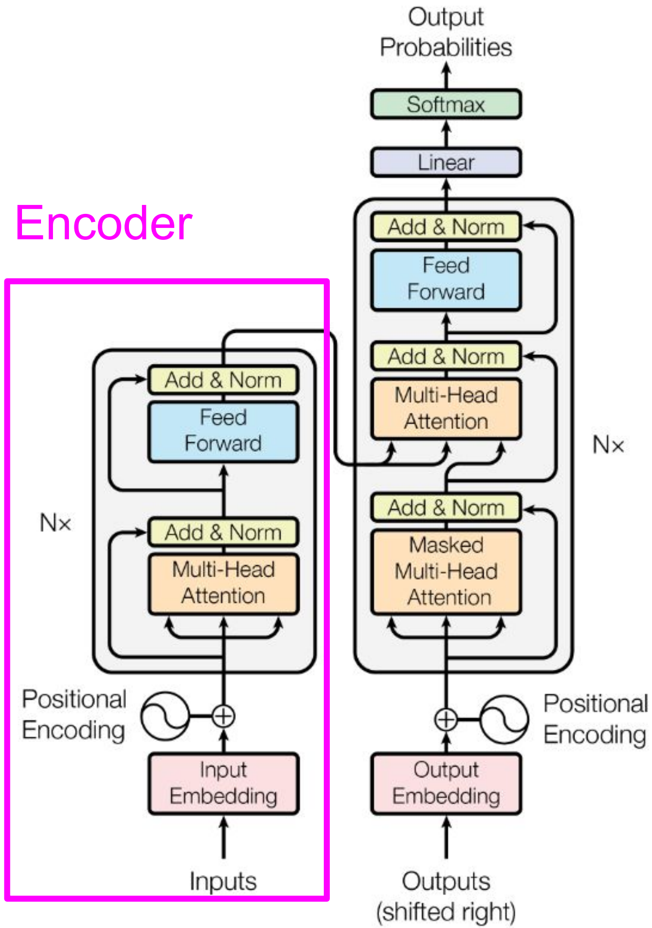
1. Feed sentence to encoder
2. Encoder outputs representation of sentence
3. Representation is fed to decoder

The intuition

1. Feed sentence to encoder
2. Encoder outputs representation of sentence
3. Representation is fed to decoder
4. Decoder generates translation

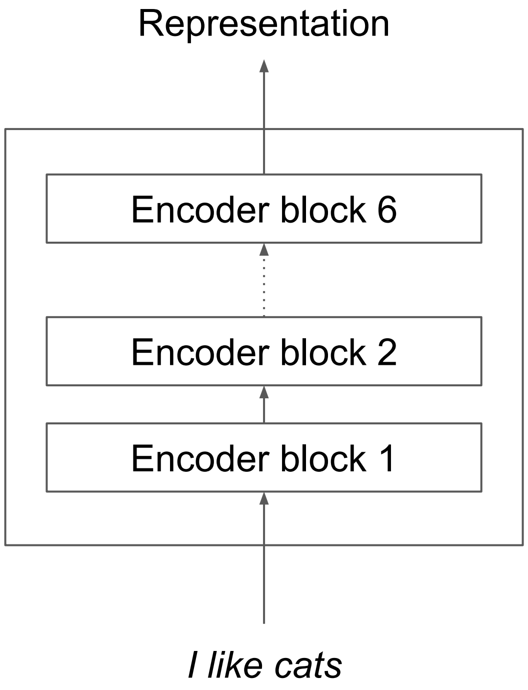
The intuition




Encoder
Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Output
Embedding
Positional
Encoding
Outputs
(shifted right)

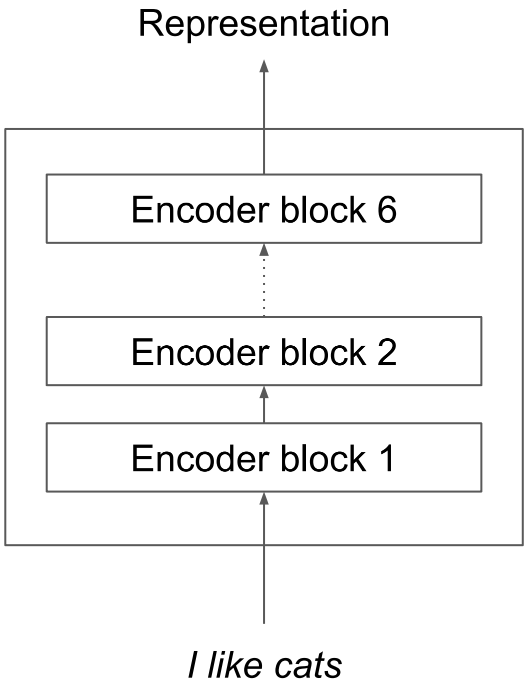
Encoder

- Stack of *N* encoder blocks



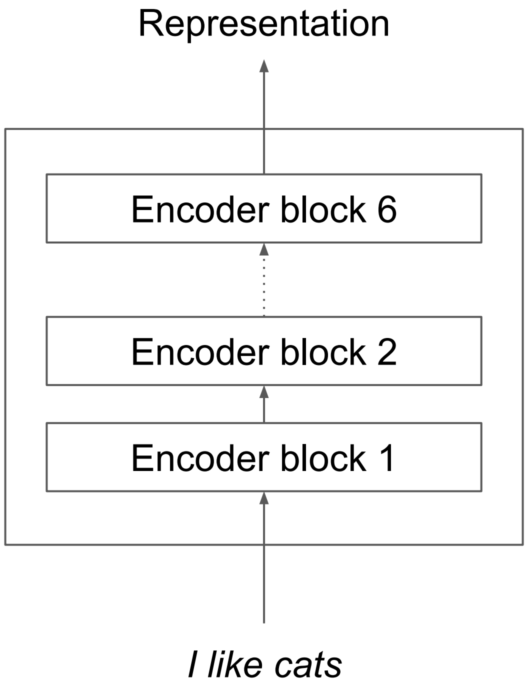
Encoder

- Stack of *N* encoder blocks
- Output of one block is input for next



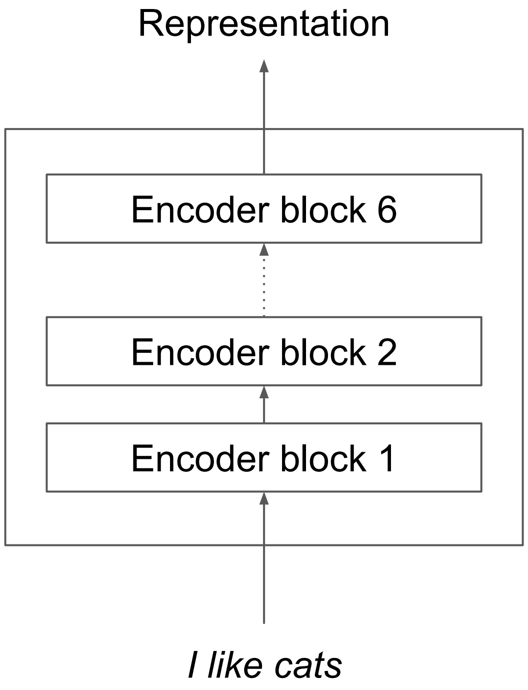
Encoder

- Stack of *N* encoder blocks
- Output of one block is input for next
- Final encoder returns representation



Encoder

- Stack of *N* encoder blocks
- Output of one block is input for next
- Final encoder returns representation
- *Attention Is All You Need:* 6 encoders



Why stacking encoder blocks?

- Complex representations

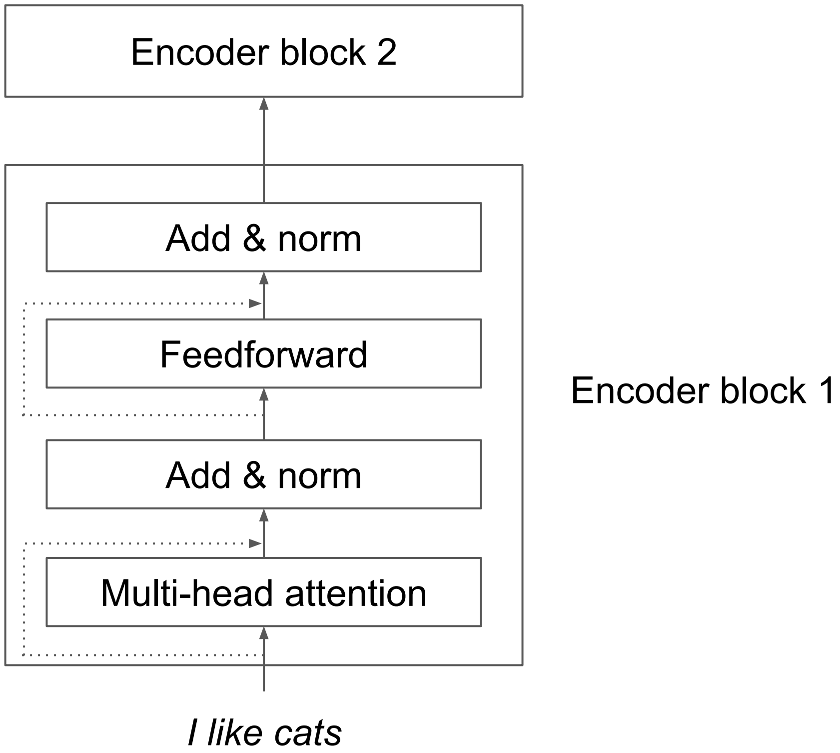
Why stacking encoder blocks?

- Complex representations
- Learning different aspects at each layer

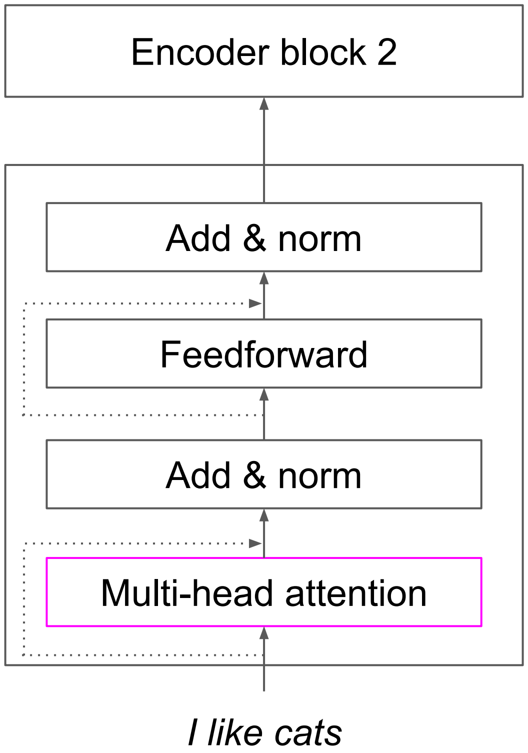
Why stacking encoder blocks?

- Complex representations
- Learning different aspects at each layer
- Improved contextualization

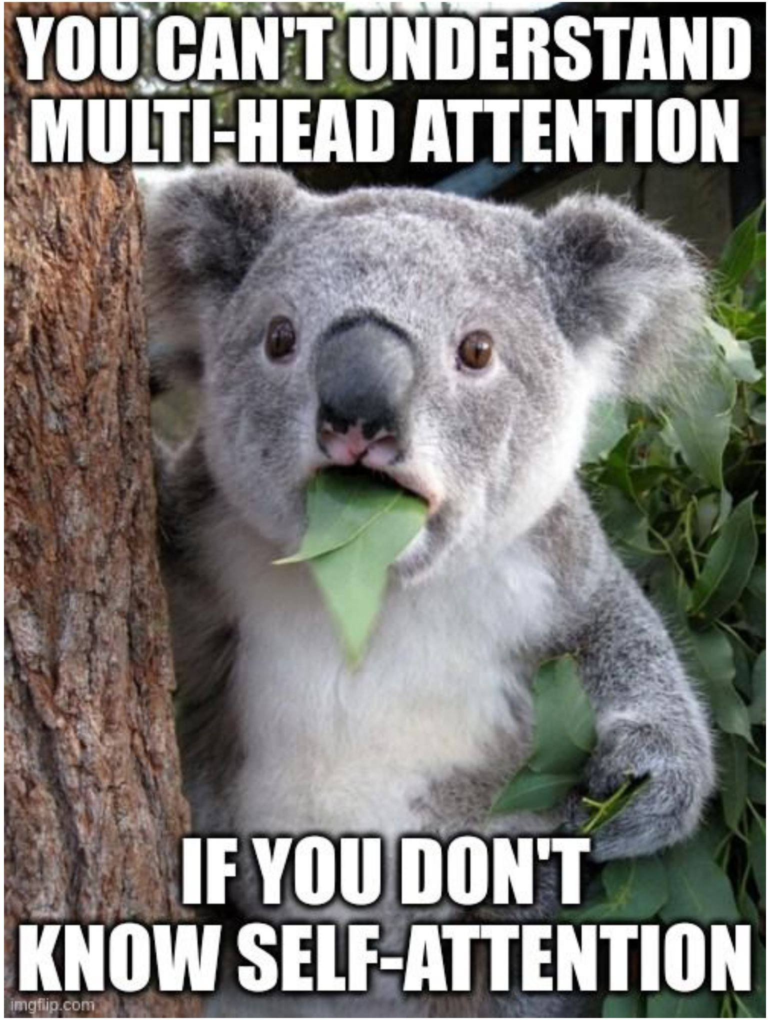
Encoder block



Encoder block



Encoder block 1


YOU CAN'T UNDERSTAND MULTI-HEAD ATTENTION
IF YOU DON'T KNOW SELF-ATTENTION
imgflip.com

A reference problem

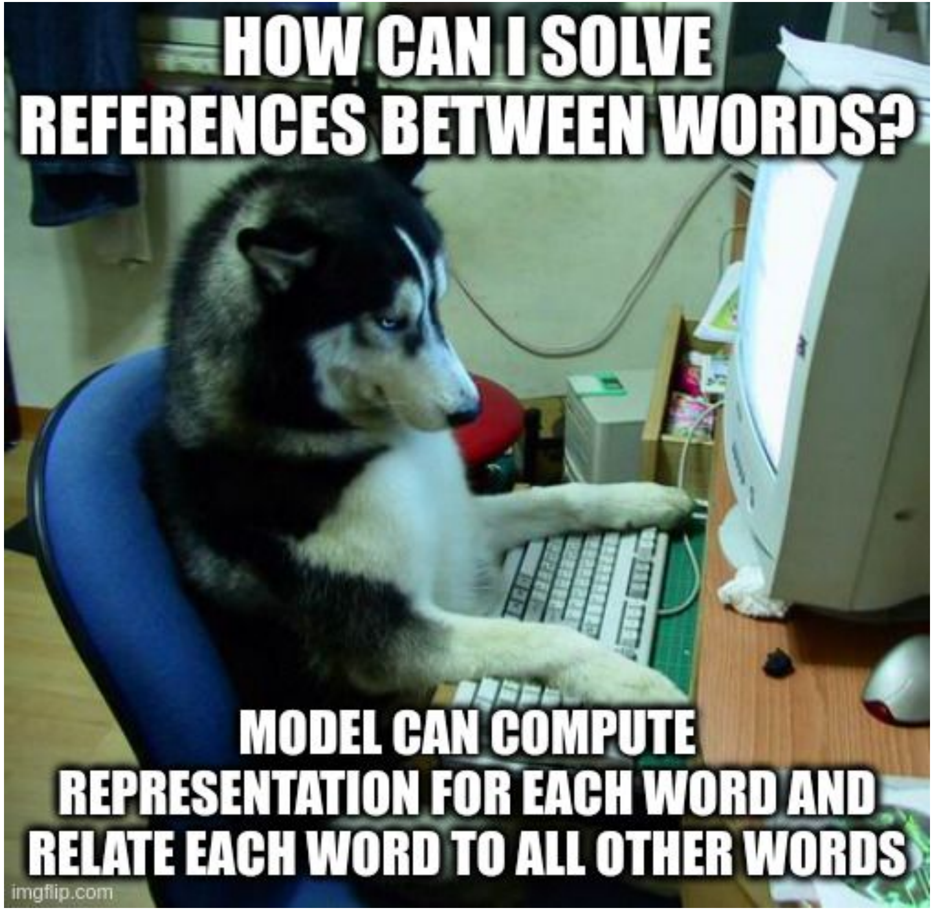
My friends like tomatoes because they are tasty

A reference problem

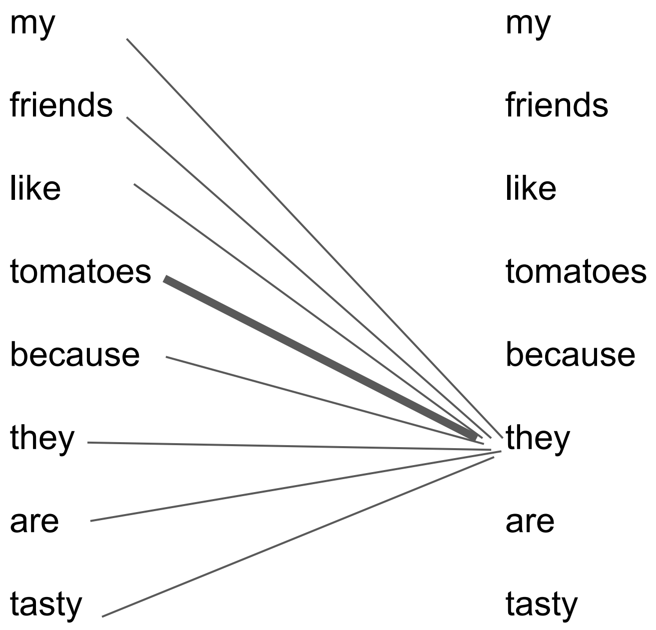
My friends like tomatoes because they are tasty

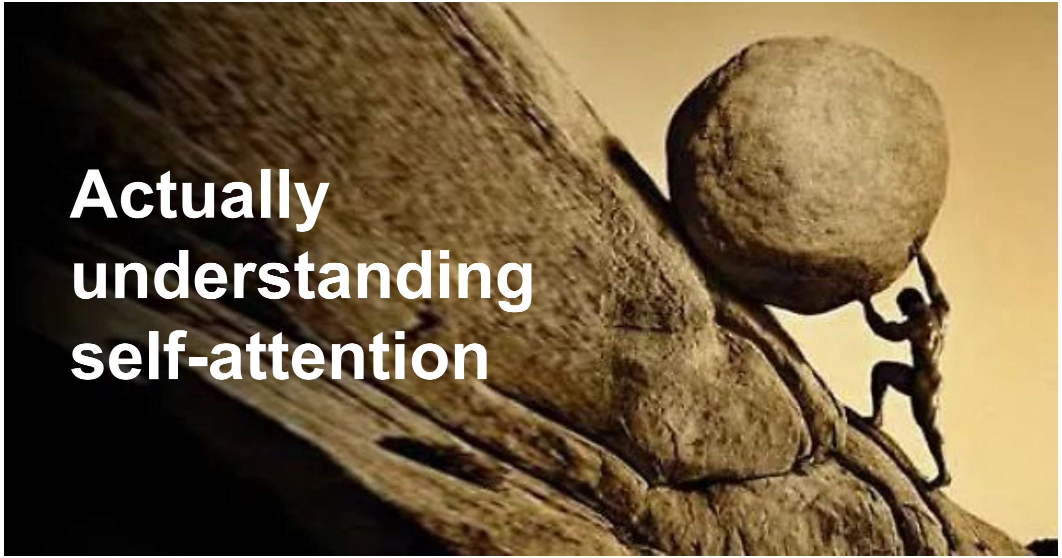
A reference problem

My friends like tomatoes because they are tasty


HOW CAN I SOLVE
REFERENCES BETWEEN WORDS?
MODEL CAN COMPUTE
REPRESENTATION FOR EACH WORD AND
RELATE EACH WORD TO ALL OTHER WORDS
imgflip.com

Self-attention: Intuition




Actually
understanding
self-attention

Self-attention: Protagonists

- Input embedding matrix (I)
- Query matrix (Q)
- Key matrix (K)
- Value matrix (V)

Input embedding matrix

- # of words x embedding dimension

Input embedding matrix

- # of words x embedding dimension
- Each row is a word vector

Input embedding matrix

- # of words x embedding dimension
- Each row is a word vector

I like cats $\longrightarrow$ $\begin{array}{c c} \text{I} & \\ \text{like} & \\ \text{cats} & \end{array} \begin{bmatrix} 0.2 & 1.2 \\ 0.5 & 4.1 \\ 2.1 & 0.4 \end{bmatrix}$

Query, key, value matrices

Query (Q)

$$\begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \\ 1.9 & 0.1 \end{bmatrix}$$

Key (K)

$$\begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.6 & 2.4 \\ 0.8 & 1.7 \\ 2.5 & 0.3 \end{bmatrix}$$

Value (V)

$$\begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.4 & 1.0 \\ 1.2 & 2.8 \\ 1.7 & 0.2 \end{bmatrix}$$

How do we derive Q, K, V?

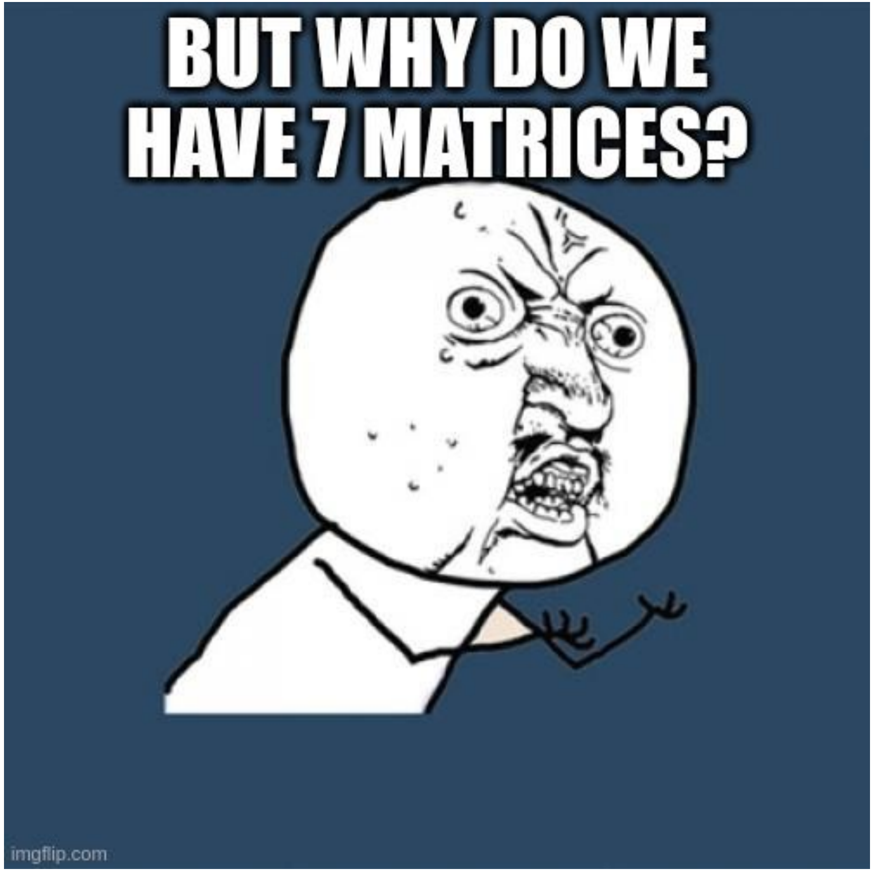
- Multiply input matrix by 3 weight matrices

$$IW_Q = Q$$
$$IW_K = K$$
$$IW_V = V$$

How do we derive Q, K, V?

- Multiply input matrix by 3 weight matrices
- Learn weights during training

$$IW_Q = Q$$
$$IW_K = K$$
$$IW_V = V$$


BUT WHY DO WE HAVE 7 MATRICES?
imgflip.com

Relation of a word
with all other words
is computed
through Q, K, V

Self-attention: Formalisation

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

Self-attention: Step 1

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

Self-attention: Step 1

Key (K)

$$\begin{array}{r} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.6 & 2.4 \\ 0.8 & 1.7 \\ 2.5 & 0.3 \end{bmatrix}$$

Self-attention: Step 1

Key (K)

$$\begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.6 & 2.4 \\ 0.8 & 1.7 \\ 2.5 & 0.3 \end{bmatrix}$$

$$K^T = \begin{bmatrix} 0.6 & 0.8 & 2.5 \\ 2.4 & 1.7 & 0.3 \end{bmatrix}$$

Self-attention: Step 1

$$\begin{array}{cc} & \text{Key (K)} \\ \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} & \begin{bmatrix} \boxed{\begin{matrix} 0.6 \\ 0.8 \\ 2.5 \end{matrix}} & \begin{matrix} 2.4 \\ 1.7 \\ 0.3 \end{matrix} \end{bmatrix} \end{array}$$

$$K^T = \begin{bmatrix} \boxed{0.6 \quad 0.8 \quad 2.5} \\ 2.4 \quad 1.7 \quad 0.3 \end{bmatrix}$$

Self-attention: Step 1

Key (K)

$$\begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.6 & \boxed{2.4} \\ 0.8 & \boxed{1.7} \\ 2.5 & \boxed{0.3} \end{bmatrix}$$

$$K^T = \begin{bmatrix} 0.6 & 0.8 & 2.5 \\ \boxed{2.4} & \boxed{1.7} & \boxed{0.3} \end{bmatrix}$$

Self-attention: Step 1

$$QK^T = \begin{array}{c c} & \\ \begin{array}{r} \text{I} \\ \text{like} \\ \text{cats} \end{array} & \underbrace{\begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \\ 1.9 & 0.1 \end{bmatrix}}_{Q} \end{array} \quad \begin{array}{c} \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \underbrace{\begin{bmatrix} 0.6 & 0.8 & 2.5 \\ 2.4 & 1.7 & 0.3 \end{bmatrix}}_{K^T} \end{array}$$

Self-attention: Step 1

$$QK^T = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \underbrace{\begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \\ 1.9 & 0.1 \end{bmatrix}}_{Q} \begin{array}{c} q_1 \\ \\ \\ \end{array} \underbrace{\begin{array}{ccc} \text{I} & \text{like} & \text{cats} \\ \left[\begin{matrix} 0.6 \\ 2.4 \end{matrix}\right. & \begin{matrix} 0.8 \\ 1.7 \end{matrix} & \left.\begin{matrix} 2.5 \\ 0.3 \end{matrix}\right] \\ k_1 & & \end{array}}_{K^T}$$

Self-attention: Step 1

$$QK^T = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \underbrace{\begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \\ 1.9 & 0.1 \end{bmatrix}}_{Q} \begin{array}{c} q_1 \\ q_2 \\ q_3 \end{array} \underbrace{\begin{array}{ccc} \text{I} & \text{like} & \text{cats} \\ \multicolumn{3}{c}{\begin{bmatrix} 0.6 & 0.8 & 2.5 \\ 2.4 & 1.7 & 0.3 \end{bmatrix}} \\ k_1 & k_2 & k_3 \end{array}}_{K^T}$$

Self-attention: Step 1

$$QK^T = \underset{Q}{\begin{array}{r} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \\ 1.9 & 0.1 \end{bmatrix} \begin{array}{l} q_1 \\ q_2 \\ q_3 \end{array}} \underset{K^T}{\begin{array}{c} \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \begin{bmatrix} 0.6 & 0.8 & 2.5 \\ 2.4 & 1.7 & 0.3 \end{bmatrix} \\ \begin{array}{ccc} k_1 & k_2 & k_3 \end{array} \end{array}} = \begin{bmatrix} q_1k_1 & q_1k_2 & q_1k_3 \\ q_2k_1 & q_2k_2 & q_2k_3 \\ q_3k_1 & q_3k_2 & q_3k_3 \end{bmatrix}$$

Self-attention: Step 1

$$QK^T = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \underset{Q}{\begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \\ 1.9 & 0.1 \end{bmatrix}} \begin{array}{c} q_1 \\ q_2 \\ q_3 \end{array} \underset{K^T}{\begin{array}{ccc} \text{I} & \text{like} & \text{cats} \\ \multicolumn{3}{c}{\begin{bmatrix} 0.6 & 0.8 & 2.5 \\ 2.4 & 1.7 & 0.3 \end{bmatrix}} \\ k_1 & k_2 & k_3 \end{array}} = \begin{bmatrix} \boxed{q_1 k_1} & q_1 k_2 & q_1 k_3 \\ q_2 k_1 & q_2 k_2 & q_2 k_3 \\ q_3 k_1 & q_3 k_2 & q_3 k_3 \end{bmatrix}$$

Self-attention: Step 1

$$QK^T = \begin{array}{r} \text{I} \\ \text{like} \\ \text{cats} \end{array} \underset{Q}{\begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \\ 1.9 & 0.1 \end{bmatrix}} \begin{array}{l} q_1 \\ q_2 \\ q_3 \end{array} \begin{array}{c} \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \begin{bmatrix} 0.6 & 0.8 & 2.5 \\ 2.4 & 1.7 & 0.3 \end{bmatrix} \\ \begin{array}{ccc} k_1 & k_2 & k_3 \end{array} \\ K^T \end{array} = \begin{bmatrix} q_1k_1 & \boxed{q_1k_2} & q_1k_3 \\ q_2k_1 & q_2k_2 & q_2k_3 \\ q_3k_1 & q_3k_2 & q_3k_3 \end{bmatrix}$$

Self-attention: Step 1

$$
QK^T = \underbrace{\begin{array}{r}\text{I}\\ \text{like}\\ \text{cats}\end{array}\begin{bmatrix} 1.3 & 0.8 \\ \boxed{0.7 \quad 3.5} \\ 1.9 & 0.1 \end{bmatrix}\begin{array}{l} q_1 \\ q_2 \\ q_3 \end{array}}_{Q}
\underbrace{\begin{array}{ccc} \text{I} & \text{like} & \text{cats} \\ \left[\begin{array}{cc} 0.6 & 0.8 \\ 2.4 & 1.7 \end{array}\right. & & \left.\boxed{\begin{array}{c} 2.5 \\ 0.3 \end{array}}\right] \\ k_1 & k_2 & k_3 \end{array}}_{K^T}
= \begin{bmatrix} q_1 k_1 & q_1 k_2 & q_1 k_3 \\ q_2 k_1 & q_2 k_2 & \boxed{q_2 k_3} \\ q_3 k_1 & q_3 k_2 & q_3 k_3 \end{bmatrix}
$$

Self-attention: Step 1

$$QK^T = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \underbrace{\begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \\ 1.9 & 0.1 \end{bmatrix}}_{Q} \begin{array}{c} q_1 \\ q_2 \\ q_3 \end{array} \underbrace{\begin{array}{ccc} \text{I} & \text{like} & \text{cats} \\ \begin{bmatrix} 0.6 & 0.8 & 2.5 \\ 2.4 & 1.7 & 0.3 \end{bmatrix} \\ k_1 & k_2 & k_3 \end{array}}_{K^T} = \begin{bmatrix} q_1k_1 & q_1k_2 & q_1k_3 \\ q_2k_1 & q_2k_2 & q_2k_3 \\ q_3k_1 & q_3k_2 & q_3k_3 \end{bmatrix} = \begin{array}{c|ccc} & \text{I} & \text{like} & \text{cats} \\ \hline \text{I} & 2.7 & 2.4 & 3.49 \\ \text{like} & 8.82 & 6.51 & 2.8 \\ \text{cats} & 1.38 & 1.69 & 4.78 \end{array}$$

Self-attention: Step 1

$$QK^T = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 1.3 & 0.8 \\ 0.7 & 3.5 \\ 1.9 & 0.1 \end{bmatrix} \begin{array}{c} q_1 \\ q_2 \\ q_3 \end{array} \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \\ \begin{bmatrix} 0.6 \\ 2.4 \end{bmatrix} & \begin{matrix} 0.8 \\ 1.7 \end{matrix} & \begin{bmatrix} 2.5 \\ 0.3 \end{bmatrix} \\ k_1 & k_2 & k_3 \end{array} = \begin{bmatrix} q_1k_1 & q_1k_2 & q_1k_3 \\ q_2k_1 & q_2k_2 & q_2k_3 \\ q_3k_1 & q_3k_2 & q_3k_3 \end{bmatrix} = \begin{array}{cc} & \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} & \begin{bmatrix} 2.7 & 2.4 & 3.49 \\ 8.82 & 6.51 & 2.8 \\ 1.38 & \boxed{1.69} & 4.78 \end{bmatrix} \end{array}$$

$$Q \qquad\qquad K^T$$

Similarity score
between *cats* and *like*

Self-attention: Step 1

$$QK^T$$

Similarity score between all words

What are Q and K really?

What are Q and K really?

- Q is the current item in the sequence that you're processing

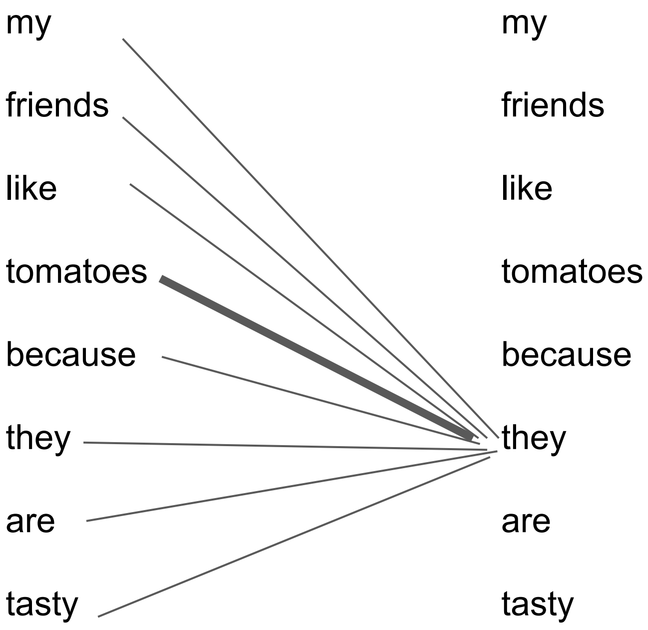
What are Q and K really?

- Q is the current item in the sequence that you're processing
- *“What other words in the sequence are relevant to me, and how relevant are they?”*

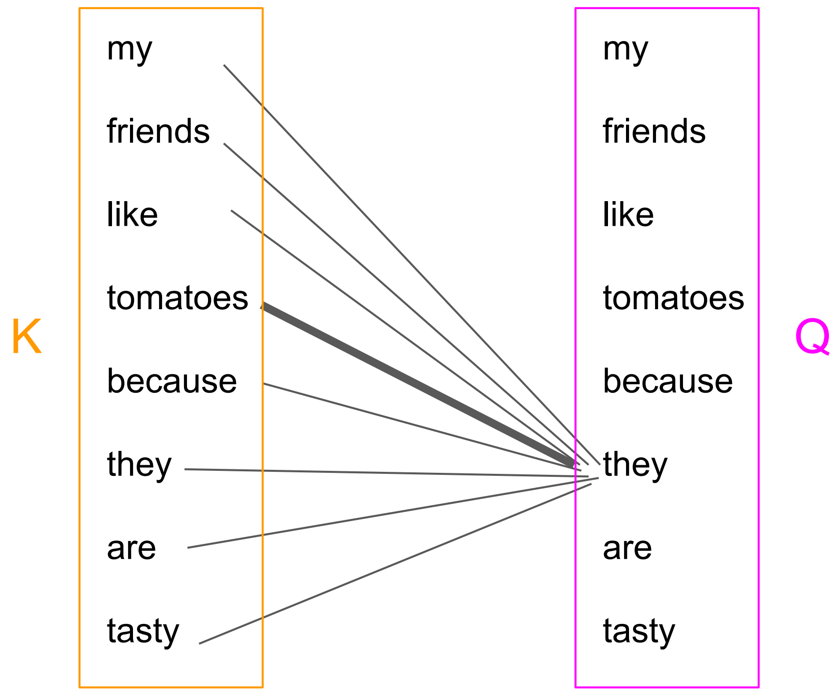
What are Q and K really?

- Q is the current item in the sequence that you're processing
- *“What other words in the sequence are relevant to me, and how relevant are they?”*
- K correspond to all the items in the sequence that the query can attend to

What are Q and K really?



What are Q and K really?



Self-attention: Step 1

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

Self-attention: Step 2

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

Self-attention: Step 2

- Divide QK^T by square root of embedding dimension of K

Self-attention: Step 2

- Divide QK^T by square root of embedding dimension of K

Key (K)

$$\begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.6 & 2.4 \\ 0.8 & 1.7 \\ 2.5 & 0.3 \end{bmatrix}$$

Self-attention: Step 2

- Divide QK^T by square root of embedding dimension of K

Key (K)

embedding dimension = 2

$$\begin{array}{r c} \text{I} & \\ \text{like} & \\ \text{cats} & \end{array}\begin{bmatrix} \boxed{0.6 \quad 2.4} \\ 0.8 \quad 1.7 \\ 2.5 \quad 0.3 \end{bmatrix}$$

Self-attention: Step 2

- Divide QK^T by square root of embedding dimension of K

$$\frac{QK^T}{\sqrt{d_k}} = \frac{1}{\sqrt{2}} \times \begin{bmatrix} 2.7 & 2.4 & 3.49 \\ 8.82 & 6.51 & 2.8 \\ 1.38 & 1.69 & 4.78 \end{bmatrix} = \begin{bmatrix} \frac{2.7}{\sqrt{2}} & \frac{2.4}{\sqrt{2}} & \frac{3.49}{\sqrt{2}} \\ \frac{8.82}{\sqrt{2}} & \frac{6.51}{\sqrt{2}} & \frac{2.8}{\sqrt{2}} \\ \frac{1.38}{\sqrt{2}} & \frac{1.69}{\sqrt{2}} & \frac{4.78}{\sqrt{2}} \end{bmatrix}$$

Self-attention: Step 2

- Divide QK^T by square root of embedding dimension of K
- Scaling -> more stable gradients

$$\frac{QK^T}{\sqrt{d_k}} = \frac{1}{\sqrt{2}} \times \begin{bmatrix} 2.7 & 2.4 & 3.49 \\ 8.82 & 6.51 & 2.8 \\ 1.38 & 1.69 & 4.78 \end{bmatrix} = \begin{bmatrix} \frac{2.7}{\sqrt{2}} & \frac{2.4}{\sqrt{2}} & \frac{3.49}{\sqrt{2}} \\ \frac{8.82}{\sqrt{2}} & \frac{6.51}{\sqrt{2}} & \frac{2.8}{\sqrt{2}} \\ \frac{1.38}{\sqrt{2}} & \frac{1.69}{\sqrt{2}} & \frac{4.78}{\sqrt{2}} \end{bmatrix}$$

Self-attention: Step 2

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

Self-attention: Step 3

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

Self-attention: Step 3

- Normalize similarity scores

Self-attention: Step 3

- Normalize similarity scores
- Apply *softmax*

Self-attention: Step 3

- Normalize similarity scores
- Apply *softmax*
- Each word vector (row) adds up to 1 (probability)

Self-attention: Step 3

- Normalize similarity scores
- Apply *softmax*
- Each word vector (row) adds up to 1 (probability)

$$\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) = \begin{array}{c c} & \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} & \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \end{array}$$

*values in the matrix completely made up

Self-attention: Step 3

- Normalize similarity scores
- Apply *softmax*
- Each word vector (row) adds up to 1 (probability)

$$\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) = \begin{array}{c c c} & \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} & \\ \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} & \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ \boxed{0.2 \quad 0.6 \quad 0.2} \\ 0.4 & 0.1 & 0.5 \end{bmatrix} & \begin{array}{c} \\ 1 \\ \\ \end{array} \end{array}$$

*values in the matrix completely made up

Self-attention: Step 3

$$\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Attention score

Relevance of different parts of the sequence to each other

Self-attention: Step 3

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

Self-attention: Step 4

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

Self-attention: Step 4

- Multiply attention score by value matrix V

Self-attention: Step 4

- Multiply attention score by value matrix V
- Attention matrix Z holds relationship of each word with each other

Self-attention: Step 4

$$Z = \begin{array}{cc} & \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} & \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \end{array} \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.4 & 1.0 \\ 1.2 & 2.8 \\ 1.7 & 0.2 \end{bmatrix}$$

$$\mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \qquad V$$

Self-attention: Step 4

$$
Z = \begin{array}{c} \\ \text{I} \\ \text{like} \\ \text{cats} \end{array}
\begin{array}{c} \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \underbrace{\begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}}_{\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)} \end{array}
\begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array}
\underbrace{\begin{bmatrix} 0.4 & 1.0 \\ 1.2 & 2.8 \\ 1.7 & 0.2 \end{bmatrix}}_{V}
\begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array}
$$

Self-attention: Step 4

$$Z = \begin{array}{c c} & \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} & \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \end{array} \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.4 & 1.0 \\ 1.2 & 2.8 \\ 1.7 & 0.2 \end{bmatrix} \begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.69 & 1.28 \\ 1.14 & 1.92 \\ 1.13 & 0.78 \end{bmatrix}$$

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \qquad V$$

Self-attention: Step 4

$$Z = \begin{array}{c} \\ \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{array}{c} \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \end{array} \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.4 & 1.0 \\ 1.2 & 2.8 \\ 1.7 & 0.2 \end{bmatrix} \begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.69 & 1.28 \\ 1.14 & 1.92 \\ 1.13 & 0.78 \end{bmatrix} = \begin{bmatrix} \vec{z_1} \\ \vec{z_2} \\ \vec{z_3} \end{bmatrix}$$

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \qquad V$$

Self-attention for word “I”

$$Z = \begin{array}{c c} & \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} & \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \end{array} \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.4 & 1.0 \\ 1.2 & 2.8 \\ 1.7 & 0.2 \end{bmatrix} \begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.69 & 1.28 \\ 1.14 & 1.92 \\ 1.13 & 0.78 \end{bmatrix} = \begin{bmatrix} \vec{z}_1 \\ \vec{z}_2 \\ \vec{z}_3 \end{bmatrix}$$

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \qquad V$$

$$\vec{z}_1 = 0.7\vec{v}_1 + 0.2\vec{v}_2 + 0.1\vec{v}_3$$

I like cats

Self-attention for word "I"

$$Z = \begin{array}{c} \\ \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{array}{c} \begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array} \\ \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix} \end{array} \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.4 & 1.0 \\ 1.2 & 2.8 \\ 1.7 & 0.2 \end{bmatrix} \begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array} = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array} \begin{bmatrix} 0.69 & 1.28 \\ 1.14 & 1.92 \\ 1.13 & 0.78 \end{bmatrix} = \begin{bmatrix} \vec{z}_1 \\ \vec{z}_2 \\ \vec{z}_3 \end{bmatrix}$$

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \qquad V$$

$$\vec{z}_1 = \underset{\text{I}}{0.7\vec{v}_1} + \underset{\text{like}}{0.2\vec{v}_2} + \underset{\text{cats}}{0.1\vec{v}_3} = \underset{\text{I}}{0.7\begin{bmatrix}0.4 & 1.0\end{bmatrix}} + \underset{\text{like}}{0.2\begin{bmatrix}1.2 & 2.8\end{bmatrix}} + \underset{\text{cats}}{0.1\begin{bmatrix}1.7 & 0.2\end{bmatrix}}$$

Self-attention for word “I”

$$
Z = \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array}
\overset{\begin{array}{ccc} \text{I} & \text{like} & \text{cats} \end{array}}{\begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}}
\begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array}
\begin{bmatrix} 0.4 & 1.0 \\ 1.2 & 2.8 \\ 1.7 & 0.2 \end{bmatrix}
\begin{array}{c} v_1 \\ v_2 \\ v_3 \end{array}
= \begin{array}{c} \text{I} \\ \text{like} \\ \text{cats} \end{array}
\begin{bmatrix} 0.69 & 1.28 \\ 1.14 & 1.92 \\ 1.13 & 0.78 \end{bmatrix}
= \begin{bmatrix} \vec{z}_1 \\ \vec{z}_2 \\ \vec{z}_3 \end{bmatrix}
$$

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \qquad V$$

$$\vec{z}_1 = \underset{\text{I}}{0.7\vec{v}_1} + \underset{\text{like}}{0.2\vec{v}_2} + \underset{\text{cats}}{0.1\vec{v}_3} = \underset{\text{I}}{0.7\begin{bmatrix}0.4 & 1.0\end{bmatrix}} + \underset{\text{like}}{0.2\begin{bmatrix}1.2 & 2.8\end{bmatrix}} + \underset{\text{cats}}{0.1\begin{bmatrix}1.7 & 0.2\end{bmatrix}}$$

Sum of the value vectors weighted by the scores

A reference problem: Solved

My friends like tomatoes because they are tasty

A reference problem: Solved

My friends like tomatoes because they are tasty

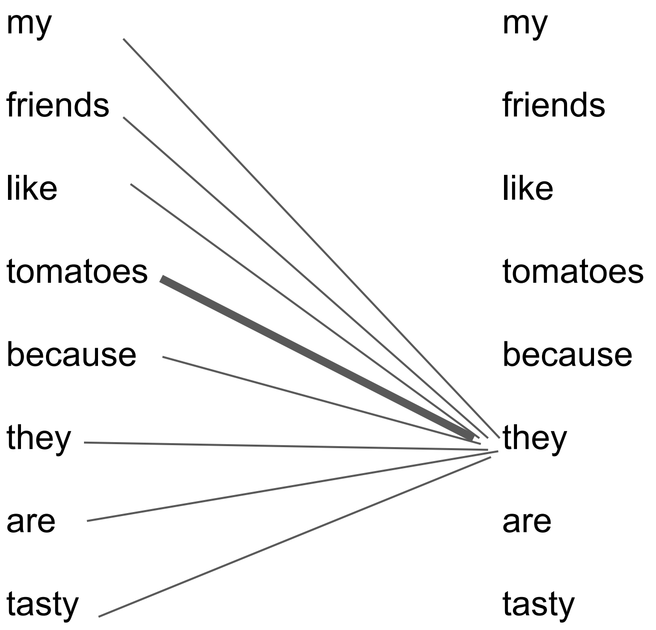
$$\vec{z}_{they} = \underbrace{0.0\vec{v}_1}_{\text{my}} + \underbrace{0.0\vec{v}_2}_{\text{friends}} + \underbrace{0.0\vec{v}_3}_{\text{like}} + \underbrace{0.9\vec{v}_4}_{\text{tomatoes}} + \underbrace{0.0\vec{v}_5}_{\text{because}} + \underbrace{0.1\vec{v}_6}_{\text{they}} + \underbrace{0.0\vec{v}_7}_{\text{are}} + \underbrace{0.0\vec{v}_8}_{\text{tasty}}$$

A reference problem: Solved

My friends like tomatoes because they are tasty

$$\vec{z}_{they} = \underset{\text{my}}{0.0\vec{v}_1} + \underset{\text{friends}}{\boxed{0.0}\vec{v}_2} + \underset{\text{like}}{0.0\vec{v}_3} + \underset{\text{tomatoes}}{\boxed{0.9}\vec{v}_4} + \underset{\text{because}}{0.0\vec{v}_5} + \underset{\text{they}}{0.1\vec{v}_6} + \underset{\text{are}}{0.0\vec{v}_7} + \underset{\text{tasty}}{0.0\vec{v}_8}$$

Intuition meets formalisation



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

Why step 4 is necessary?

Why step 4 is necessary?

- Attention score is about *relevance*

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

Why step 4 is necessary?

- Attention score is about *relevance*
- V holds *content* of the sequence

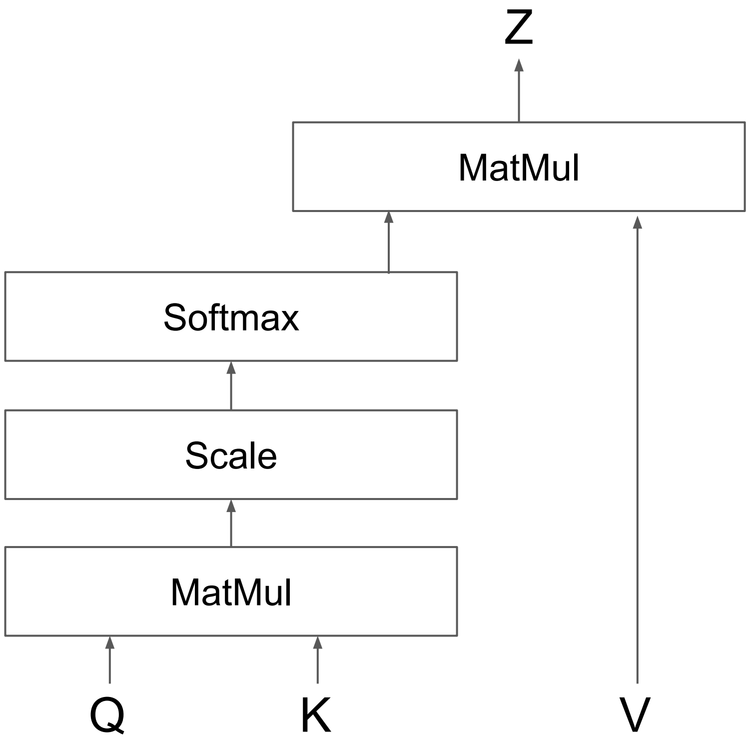
Why step 4 is necessary?

- Attention score is about *relevance*
- V holds *content* of the sequence
- Combine relevance with content

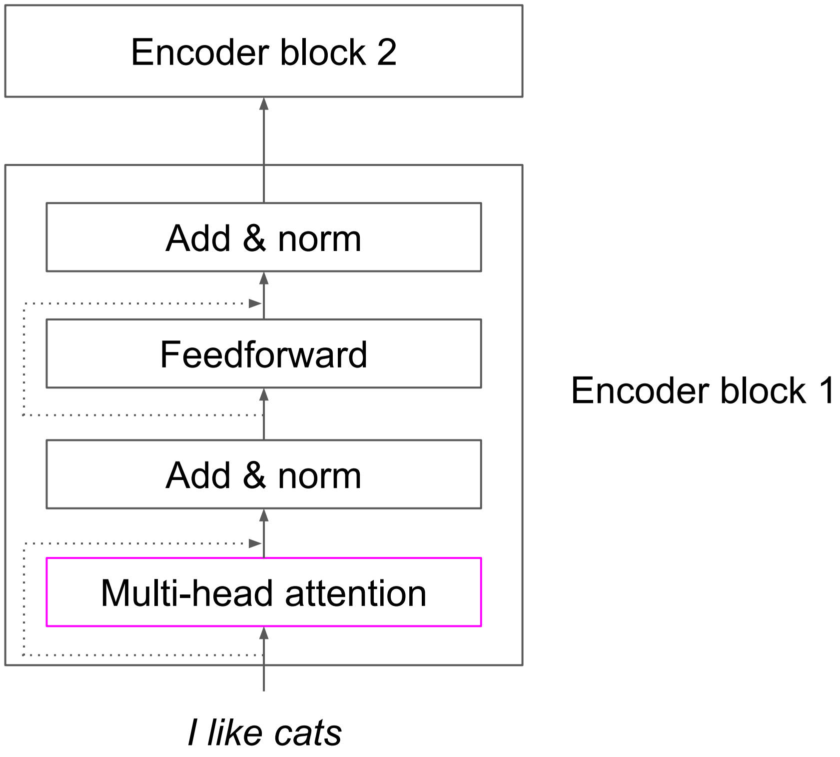
Why step 4 is necessary?

- Attention score is about *relevance*
- V holds *content* of the sequence
- Combine relevance with content
- Weigh content by its relevance

Self-attention: Visual recap



Encoder block



Multi-head attention

- Run multiple instances of the self-attention mechanism in parallel

Multi-head attention

- Run multiple instances of the self-attention mechanism in parallel
- Compute as many Q, K, V, Z matrices as the number of heads

Multi-head attention

- Run multiple instances of the self-attention mechanism in parallel
- Compute as many Q, K, V, Z matrices as the number of heads

$$Z = concatenate(Z_1, Z_2, Z_3, ..., Z_n)W_0$$

Why multiple heads?

Why multiple heads?

- Each attention head can focus on different parts of the input sequence

Why multiple heads?

- Each attention head can focus on different parts of the input sequence
- Increased representation power

Why multiple heads?

- Each attention head can focus on different parts of the input sequence
- Increased representation power
- Reducing the risk of overfitting

RNN-LSTM vs transformer

- LSTM -> feed words sequentially

RNN-LSTM vs transformer

- LSTM -> feed words sequentially
- Transformer -> all words in parallel

RNN-LSTM vs transformer

- LSTM -> feed words sequentially
- Transformer -> all words in parallel
- Parallel:
  - decrease training time
  - help learning long-term dependency


WHEN YOU REALISE TRANSFORMER
CAN'T ACCOUNT FOR SEQUENCE ORDER
imgflip.com

Positional encoding

Positional encoding

- Input matrix doesn’t have order information

Positional encoding

- Input matrix doesn’t have order information
- Goal: Find a strategy to encode positions in the embeddings

Positional encoding: Strategy

- Matrix P with same dimension as input matrix I

Positional encoding: Strategy

- Matrix P with same dimension as input matrix I
- Add P to I before feeding it to encoder

Positional encoding: Strategy

$$I' = \underbrace{\begin{bmatrix} 0.2 & 1.2 \\ 0.5 & 4.1 \\ 2.1 & 0.4 \end{bmatrix}}_{I} + \underbrace{\begin{bmatrix} 0.5 & 1.0 \\ 2.5 & 1.3 \\ 1.1 & 0.3 \end{bmatrix}}_{P} = \begin{bmatrix} 0.7 & 2.2 \\ 3.0 & 5.4 \\ 3.2 & 0.7 \end{bmatrix}$$

Positional encoding: Strategy

I carries semantic information

$$I' = \boxed{\underbrace{\begin{bmatrix} 0.2 & 1.2 \\ 0.5 & 4.1 \\ 2.1 & 0.4 \end{bmatrix}}_{I}} + \underbrace{\begin{bmatrix} 0.5 & 1.0 \\ 2.5 & 1.3 \\ 1.1 & 0.3 \end{bmatrix}}_{P} = \begin{bmatrix} 0.7 & 2.2 \\ 3.0 & 5.4 \\ 3.2 & 0.7 \end{bmatrix}$$

Positional encoding: Strategy

I carries semantic information

P encodes positional information

$$I^{'} = \boxed{\underbrace{\begin{bmatrix} 0.2 & 1.2 \\ 0.5 & 4.1 \\ 2.1 & 0.4 \end{bmatrix}}_{I}} + \boxed{\underbrace{\begin{bmatrix} 0.5 & 1.0 \\ 2.5 & 1.3 \\ 1.1 & 0.3 \end{bmatrix}}_{P}} = \begin{bmatrix} 0.7 & 2.2 \\ 3.0 & 5.4 \\ 3.2 & 0.7 \end{bmatrix}$$


MODEL LEARNS TO INTERPRET COMBINED SIGNAL AS CONTAINING BOTH TYPES OF INFORMATION
imgflip.com

How do we compute P?

How do we compute P?

$$P(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$P(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$


pmitf.com

How do we compute P?

$$\mathrm{P}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$\mathrm{P}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

How do we compute P?

word position (row index)

$$P(\boxed{pos}, 2i) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$P(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

How do we compute P?

word position (row index)

$$P(\boxed{pos}, 2\boxed{i}) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

embedding position (column index)

$$P(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

How do we compute P?

even embedding / column position

$$\boxed{\mathrm{P}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)}$$

$$\mathrm{P}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

How do we compute P?

$$P(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

odd embedding / column position

$$\boxed{P(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)}$$

How do we compute P?

$$P(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$P(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$P = \begin{array}{c} \text{Spaghetti} \\ \\ \text{monster} \\ \\ \text{is} \\ \\ \text{great} \end{array} \begin{bmatrix} \sin\left(\frac{0}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{0}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{0}{10000^{2\cdot 2/3}}\right) \\ \\ \sin\left(\frac{1}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{1}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{1}{10000^{2\cdot 2/3}}\right) \\ \\ \sin\left(\frac{2}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{2}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{2}{10000^{2\cdot 2/3}}\right) \\ \\ \sin\left(\frac{3}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{3}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{3}{10000^{2\cdot 2/3}}\right) \end{bmatrix}$$

How do we compute P?

$$\mathrm{P}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$\mathrm{P}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$P = \begin{array}{r} \text{Spaghetti} \\[2em] \text{monster} \\[2em] \text{is} \\[2em] \text{great} \end{array} \begin{bmatrix} \boxed{\sin\left(\frac{0}{10000^{2\cdot 0/3}}\right)} & \cos\left(\frac{0}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{0}{10000^{2\cdot 2/3}}\right) \\ \sin\left(\frac{1}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{1}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{1}{10000^{2\cdot 2/3}}\right) \\ \sin\left(\frac{2}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{2}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{2}{10000^{2\cdot 2/3}}\right) \\ \sin\left(\frac{3}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{3}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{3}{10000^{2\cdot 2/3}}\right) \end{bmatrix}$$

How do we compute P?

$$\mathrm{P}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$\mathrm{P}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$P = \begin{array}{c c}
\text{Spaghetti} & \multirow{4}{*}{$\begin{bmatrix}
\sin\left(\frac{0}{10000^{2\cdot 0/3}}\right) & \boxed{\cos\left(\frac{0}{10000^{2\cdot 1/2}}\right)} & \sin\left(\frac{0}{10000^{2\cdot 2/3}}\right) \\
\sin\left(\frac{1}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{1}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{1}{10000^{2\cdot 2/3}}\right) \\
\sin\left(\frac{2}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{2}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{2}{10000^{2\cdot 2/3}}\right) \\
\sin\left(\frac{3}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{3}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{3}{10000^{2\cdot 2/3}}\right)
\end{bmatrix}$} \\
\text{monster} & \\
\text{is} & \\
\text{great} &
\end{array}$$

How do we compute P?

$$P(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$P(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/dimension_{model}}}\right)$$

$$P = \begin{array}{c} \text{Spaghetti} \\[2em] \text{monster} \\[2em] \text{is} \\[2em] \text{great} \end{array} \begin{bmatrix} \sin\left(\frac{0}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{0}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{0}{10000^{2\cdot 2/3}}\right) \\ \sin\left(\frac{1}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{1}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{1}{10000^{2\cdot 2/3}}\right) \\ \sin\left(\frac{2}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{2}{10000^{2\cdot 1/2}}\right) & \sin\left(\frac{2}{10000^{2\cdot 2/3}}\right) \\ \sin\left(\frac{3}{10000^{2\cdot 0/3}}\right) & \cos\left(\frac{3}{10000^{2\cdot 1/2}}\right) & \boxed{\sin\left(\frac{3}{10000^{2\cdot 2/3}}\right)} \end{bmatrix}$$

Why a sinusoidal function?

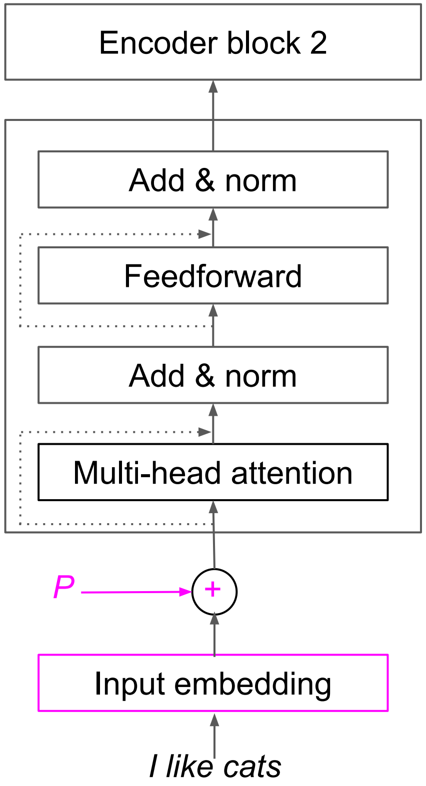
Why a sinusoidal function?

- Different frequencies for each position

Why a sinusoidal function?

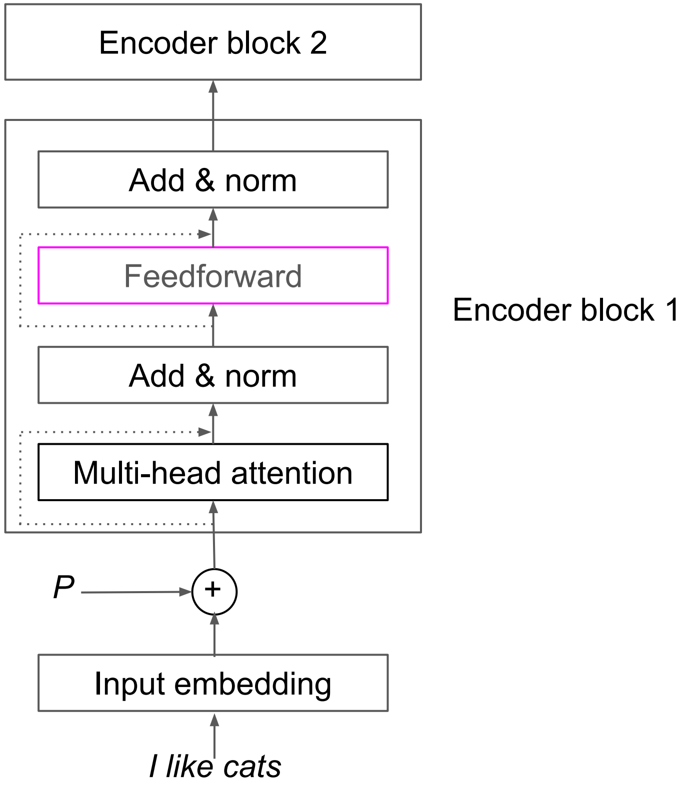
- Different frequencies for each position
- Helps model to learn relative positions

Encoder block: Revised



Encoder block 1

Encoder block: Revised



Feedforward layer

- 2 fully connected layers

Feedforward layer

- 2 fully connected layers
- Process each position data separately

Feedforward layer

- 2 fully connected layers
- Process each position data separately
- ReLU activation

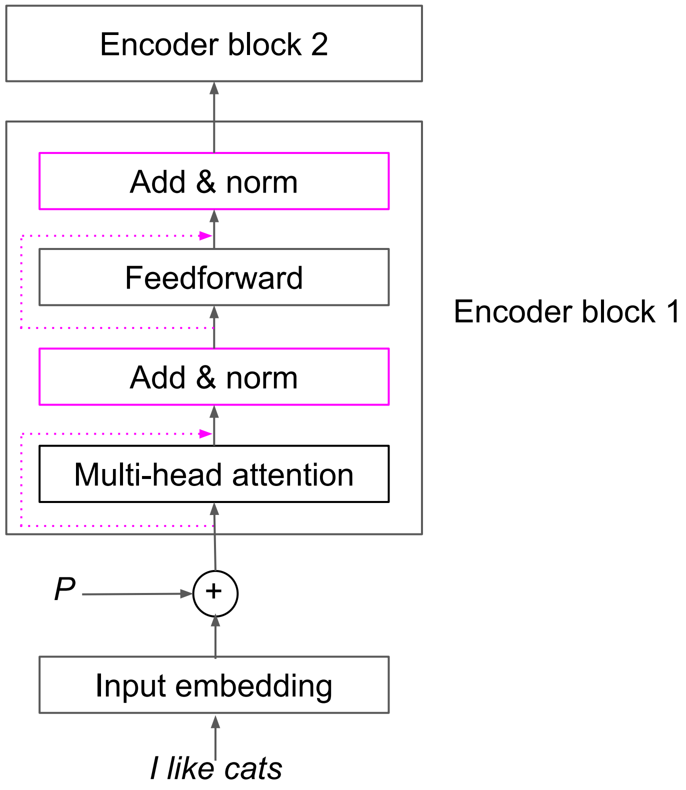
Feedforward layer

- 2 fully connected layers
- Process each position data separately
- ReLU activation
- Non-linear transformation increases complexity

Why apply feedforward?

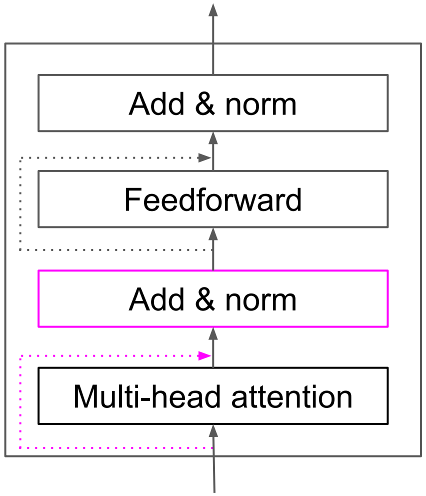
Learn more sophisticated features

Encoder block



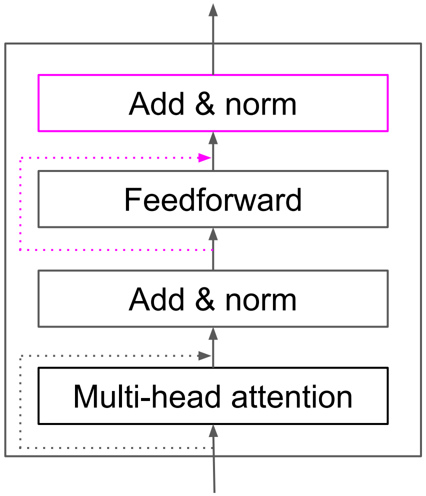
Add and norm layer

- Connects input of attention layer to its output



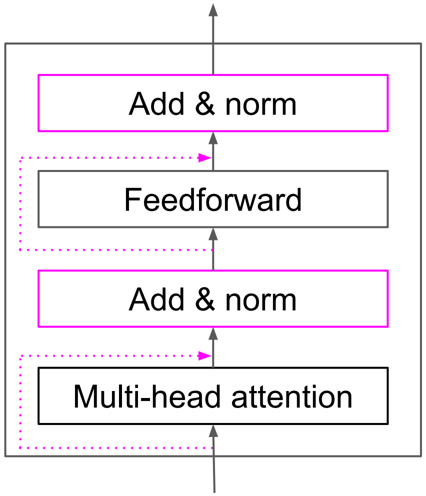
Add and norm layer

- Connects input of attention layer to its output
- Connects input of feedforward layer to its output



Add and norm layer

- Connects input of attention layer to its output
- Connects input of feedforward layer to its output
- *Residual* connections -> help with vanishing gradients



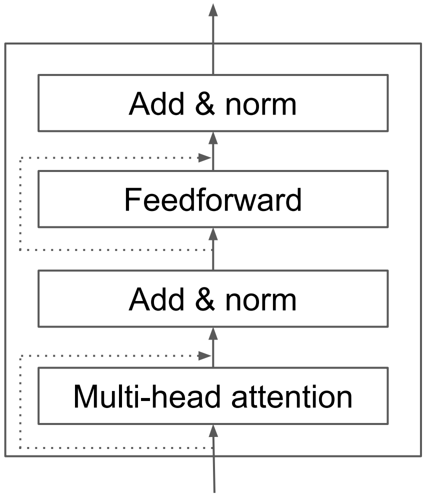
Add and norm layer

- Normalize data across features for each position (mean = 0, std dev = 1)

Add and norm layer

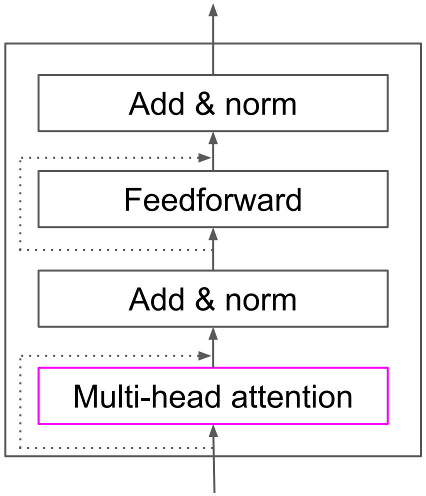
- Normalize data across features for each position (mean = 0, std dev = 1)
- Faster convergence by preventing values to change heavily

Encoder block: Deeper meaning



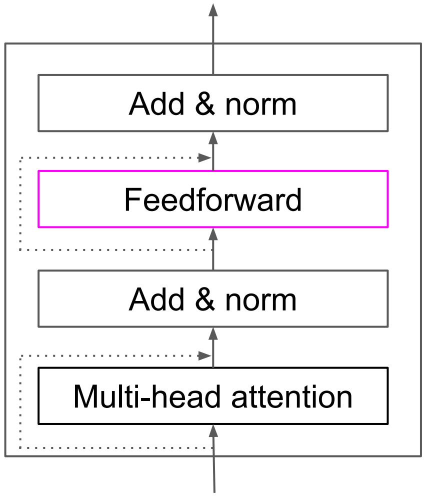
Encoder block: Deeper meaning

- Multi-head attention -> provide context



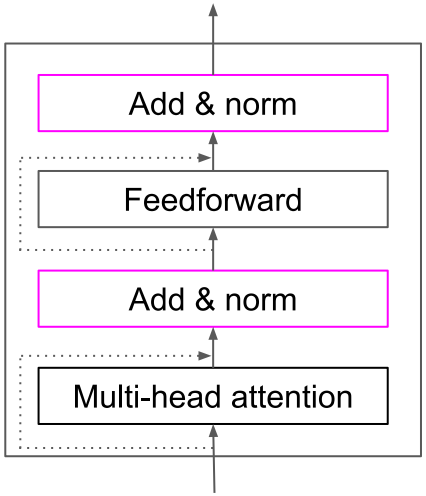
Encoder block: Deeper meaning

- Multi-head attention -> provide context
- Feedforward -> provide nuance

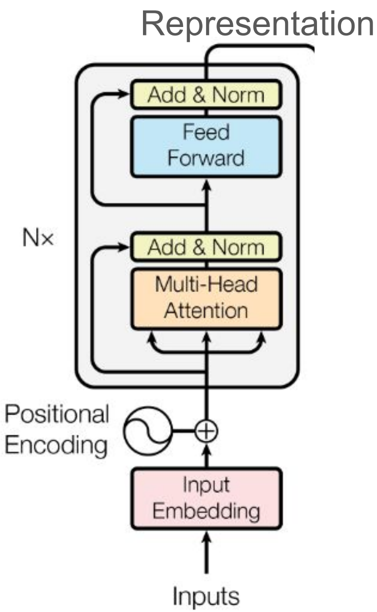


Encoder block: Deeper meaning

- Multi-head attention -> provide context
- Feedforward -> provide nuance
- Add & norm -> streamline learning

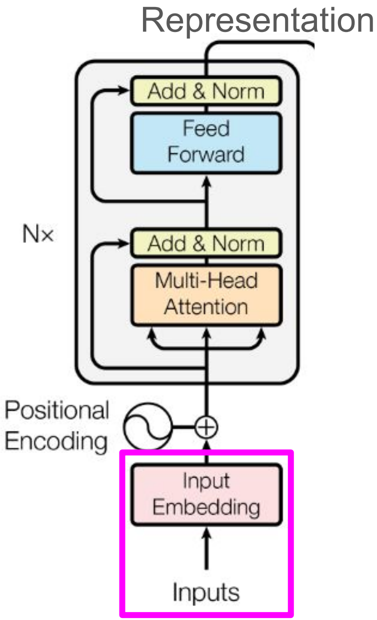


Encoder step-by-step recap



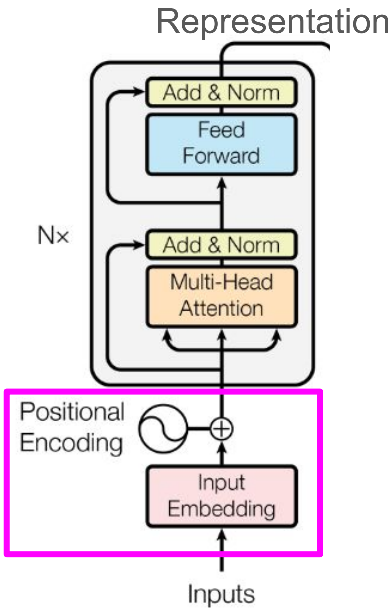
Encoder step-by-step recap

1. Create input embedding



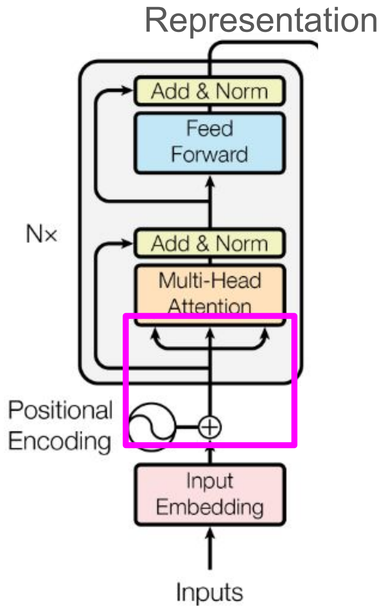
Encoder step-by-step recap

1. Create input embedding
2. Sum embedding and positional encoding



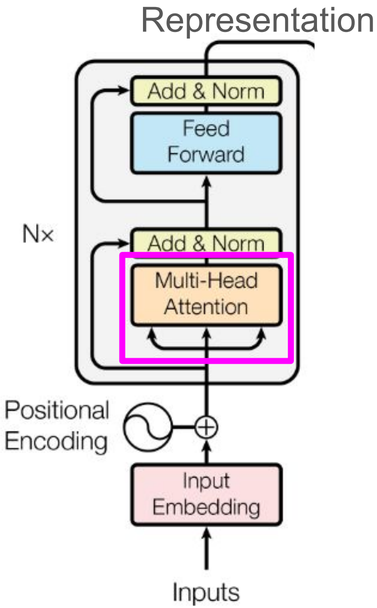
Encoder step-by-step recap

1. Create input embedding
2. Sum embedding and positional encoding
3. Compute Q, K, and V multiplying I and weight matrices



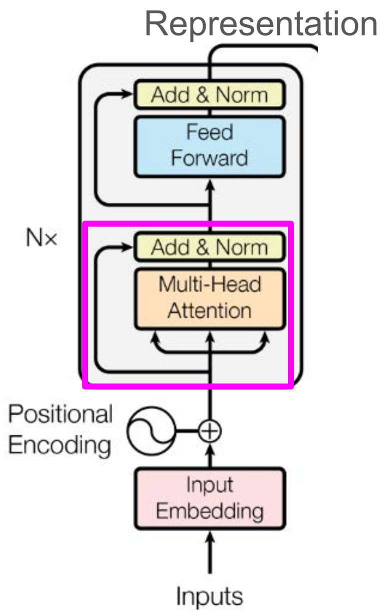
Encoder step-by-step recap

1. Create input embedding
2. Sum embedding and positional encoding
3. Compute Q, K, and V multiplying I and weight matrices
4. Compute Z concatenating all the Zi matrices produced in different attention heads



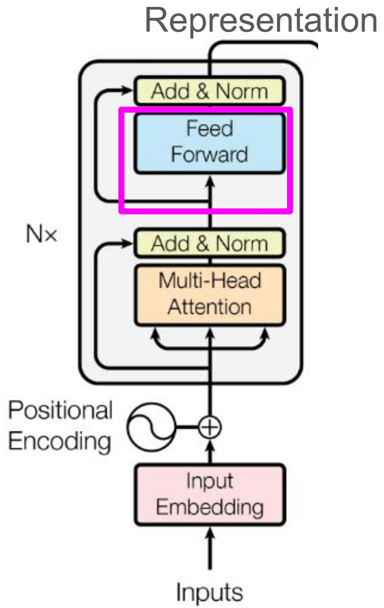
Encoder step-by-step recap

1. Create input embedding
2. Sum embedding and positional encoding
3. Compute Q, K, and V multiplying I and weight matrices
4. Compute Z concatenating all the Zi matrices produced in different attention heads
5. Add Z + input and normalize it



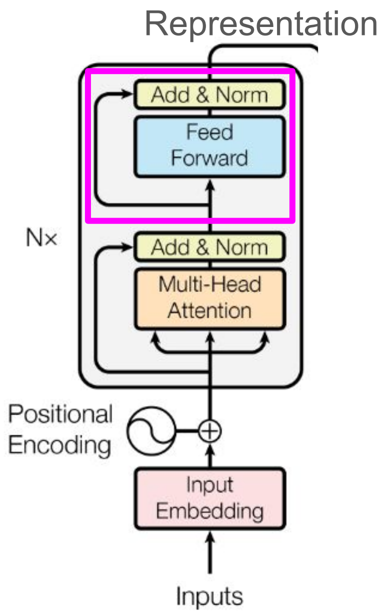
Encoder step-by-step recap

1. Create input embedding
2. Sum embedding and positional encoding
3. Compute Q, K, and V multiplying I and weight matrices
4. Compute Z concatenating all the Zi matrices produced in different attention heads
5. Add Z + input and normalize it
6. Feed output of add & norm to feedforward layer



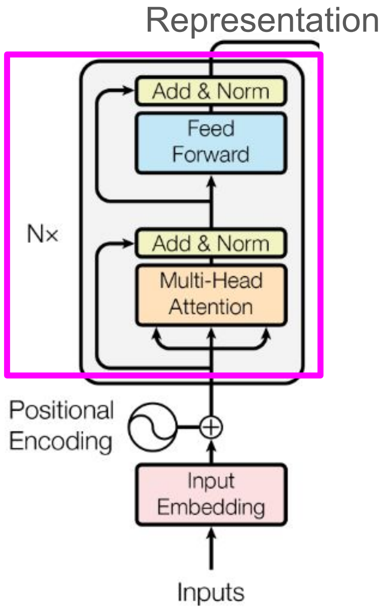
Encoder step-by-step recap

1. Create input embedding
2. Sum embedding and positional encoding
3. Compute Q, K, and V multiplying I and weight matrices
4. Compute Z concatenating all the Zi matrices produced in different attention heads
5. Add Z + input and normalize it
6. Feed output of add & norm to feedforward layer
7. Add output of feedforward layer to its input



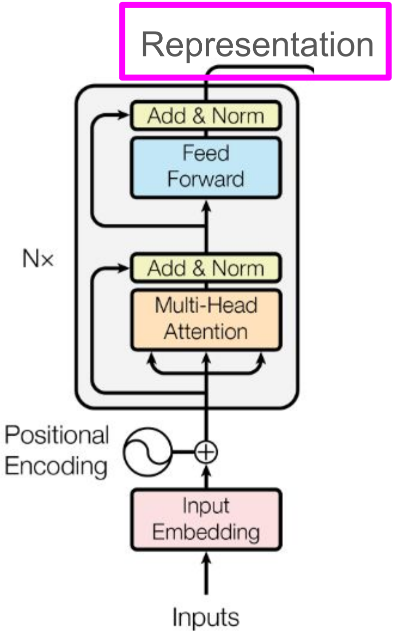
Encoder step-by-step recap

1. Create input embedding
2. Sum embedding and positional encoding
3. Compute Q, K, and V multiplying I and weight matrices
4. Compute Z concatenating all the Zi matrices produced in different attention heads
5. Add Z + input and normalize it
6. Feed output of add & norm to feedforward layer
7. Add output of feedforward layer to its input
8. Repeat for all the encoder blocks



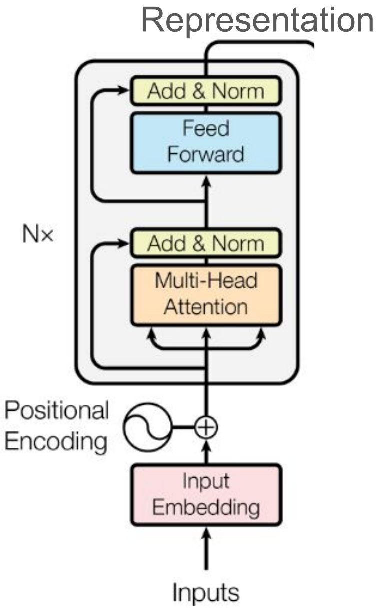
Encoder step-by-step recap

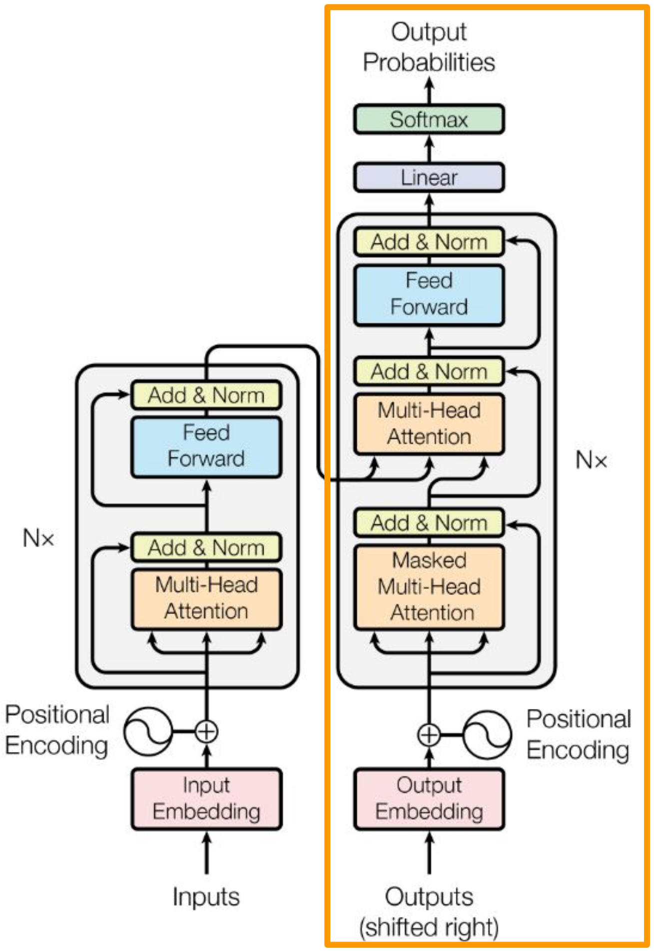
1. Create input embedding
2. Sum embedding and positional encoding
3. Compute Q, K, and V multiplying I and weight matrices
4. Compute Z concatenating all the Zi matrices produced in different attention heads
5. Add Z + input and normalize it
6. Feed output of add & norm to feedforward layer
7. Add output of feedforward layer to its input
8. Repeat for all the encoder blocks
9. Get the final representation R as output of the last block



Encoder step-by-step recap

1. Create input embedding
2. Sum embedding and positional encoding
3. Compute Q, K, and V multiplying I and weight matrices
4. Compute Z concatenating all the Zi matrices produced in different attention heads
5. Add Z + input and normalize it
6. Feed output of add & norm to feedforward layer
7. Add output of feedforward layer to its input
8. Repeat for all the encoder blocks
9. Get the final representation R as output of the last block
10. Feed R to decoder




Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Decoder

Key takeaways

- Transformers capture long-term dependencies

Key takeaways

- Transformers capture long-term dependencies
- Encoder / decoder architecture

Key takeaways

- Transformers capture long-term dependencies
- Encoder / decoder architecture
- Self-attention = relationship between words

Key takeaways

- Transformers capture long-term dependencies
- Encoder / decoder architecture
- Self-attention = relationship between words
- Multiple attention heads capture different info

Key takeaways

- Transformers capture long-term dependencies
- Encoder / decoder architecture
- Self-attention = relationship between words
- Multiple attention heads capture different info
- Encode positions in embeddings

Key takeaways

- Transformers capture long-term dependencies
- Encoder / decoder architecture
- Self-attention = relationship between words
- Multiple attention heads capture different info
- Encode positions in embeddings
- Feedforward layer adds nuance and complexity

Key takeaways

- Transformers capture long-term dependencies
- Encoder / decoder architecture
- Self-attention = relationship between words
- Multiple attention heads capture different info
- Encode positions in embeddings
- Feedforward layer adds nuance and complexity
- Add and norm facilitates learning

Key takeaways

- Transformers capture long-term dependencies
- Encoder / decoder architecture
- Self-attention = relationship between words
- Multiple attention heads capture different info
- Encode positions in embeddings
- Feedforward layer adds nuance and complexity
- Add and norm facilitates learning
- Multiple encoders blocks

Key takeaways

- Transformers capture long-term dependencies
- Encoder / decoder architecture
- Self-attention = relationship between words
- Multiple attention heads capture different info
- Encode positions in embeddings
- Feedforward layer adds nuance and complexity
- Add and norm facilitates learning
- Multiple encoders blocks
- Final representation is fed to decoder

What's up next?

Decoder [redacted]

[redacted]