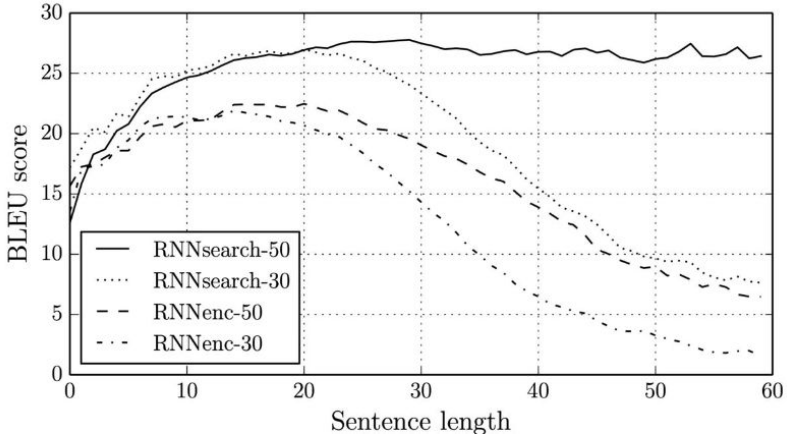
Attention is All You Need

Presenter: Alex Chandler

09-13-2022

Motivation

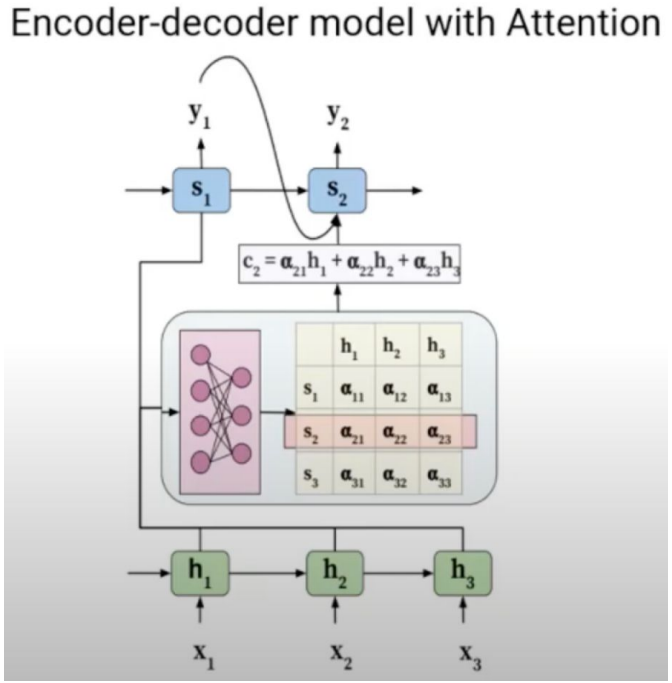
- Transformers were developed to solve the problem machine translation and sequence transduction, or neural machine translation … but they do so much more!
  - Great performance in computer vision like ViT (Dosovitskiy, 2020)
  - Image Classification (CoCa Transformer)
  - Semantic Segmentation (ex: FD-SwinV2-G Transformer)
  - Object Detection (ex: FD-SwinV2-G Transformer)



Neural Machine Translation (Bahdanau, 2014)

Previous Work

- Pre-2014 - RNN, LSTM, GRU
  - RNNs struggled with long sequence data
  - Could not handle long sequence lengths (forgetful)
    - Example on next slide
- Neural Machine Translation by Jointly Learning to Align and Translate (Bahdanau, 2014)
  - Landmark Paper
  - Proposed New Architecture:
    - bidirectional RNN as an encoder and an RNN decoder, with an attention mechanism in between
  - Proposed initial idea of attention
  - Model gave attention to particular hidden states when decoding each word
    - Learned which words to give attention to during translation for each word of decoder
  - Still required on RNN



Context

What issue are they solving?

- Previous Work: RNN, LSTM, GRU
- Slow to train (inefficient / not parallelizable)
  - factorization tricks help efficiency but still slow
- Suffer from exploding or vanishing gradients
- Cannot handle very long-term dependencies
  - In Seq-Seq models, decoder only accesses last hidden state. Early information in sentence can be lost.

"In Robotics today, Professor Martin-Martin assigned us to read the Attention is All You Need Paper. The paper proposes a new simple neural network architecture called the Transformer which relies upon attention. It outperforms RNNs and doesn't rely on recurrence or convolution. <Additional unrelated text> . I am so happy that ___ picked this paper."

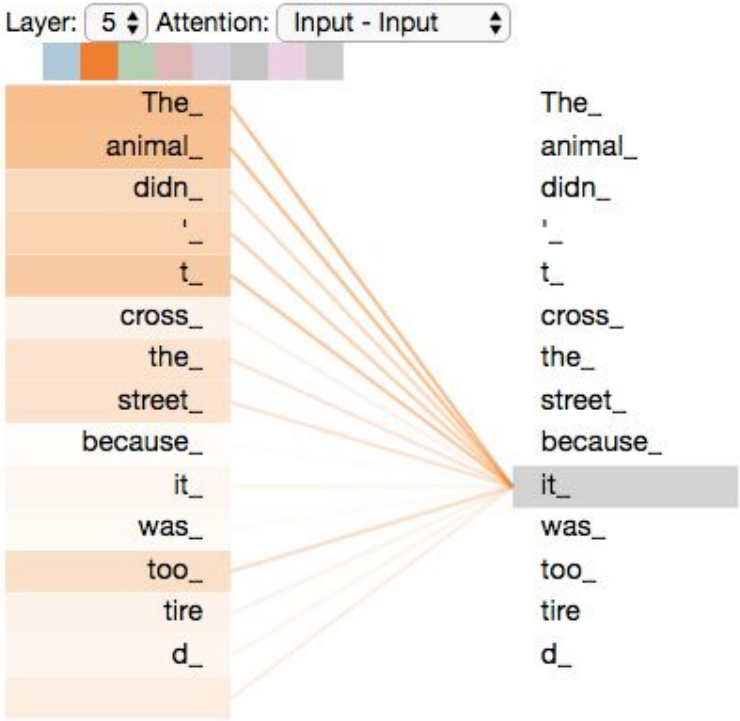
How would a RNN, LSTM, or GRU fill in the word mask?

1. Encode name inside cell state
2. Transfer cell state from one word to another
3. Decode name from previous cell state

Definitely not ideal! There has to be a better way …

Self-Attention

- Each element attends to every other element
- It is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence
- Each element becomes query, key, and value from the input embeddings by multiplying by a weight matrix



https://jalammar.github.io/illustrated-transformer/

Idea Behind Attention

Goal: Learn (differentiable) how to pick relevant information from input data

1. Create three vectors from each of the encoder's input values (query, key, value)
2. Calculate a score for how much to focus on each part of the input when we encode words at specific positions
3. How?
    - Actual math on the next slide, but the idea is to select a value (referenced by a key) relevant to a query (what trying to pull from input)

- **From Paper - "A mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key" - (Vaswani, 2017)**

key	value
name	Quinn
position	quarterback
Handedness	right

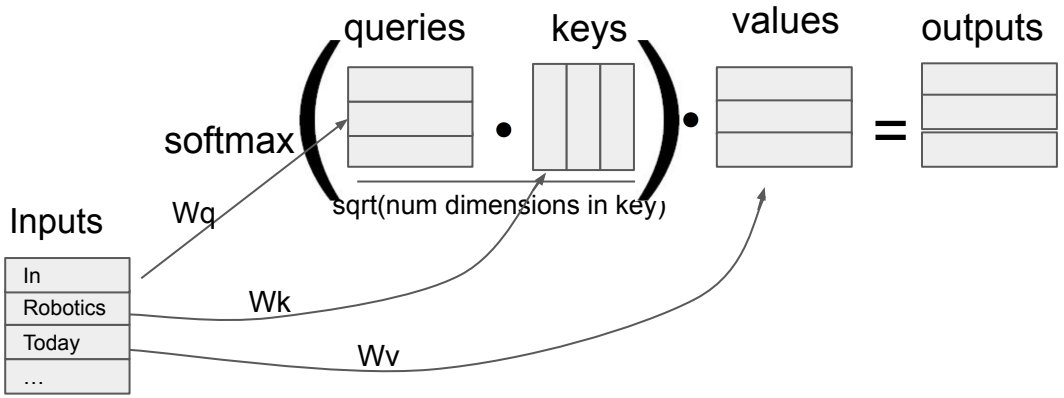
Keys and values are actually word embeddings, not words

Scaled Dot Product Attention

- Key algorithm in transformers
- Attention weights: how likely each query matches key (used for weighted sum of values)
- Dividing by d_k makes algorithm easier to train
  - prevents binary prediction of one 1 and a whole bunch of zeros
  - helps the gradients flow
- then can take loss, backpropagate, update the weights and values
- NOT stepping through a sequence like with a RNN

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

- **input consists of queries and keys of dimension dk, and values of dimension d**
- **Queries packed into matrix Q**
- **The keys and values are also packed together into matrices K and V .**



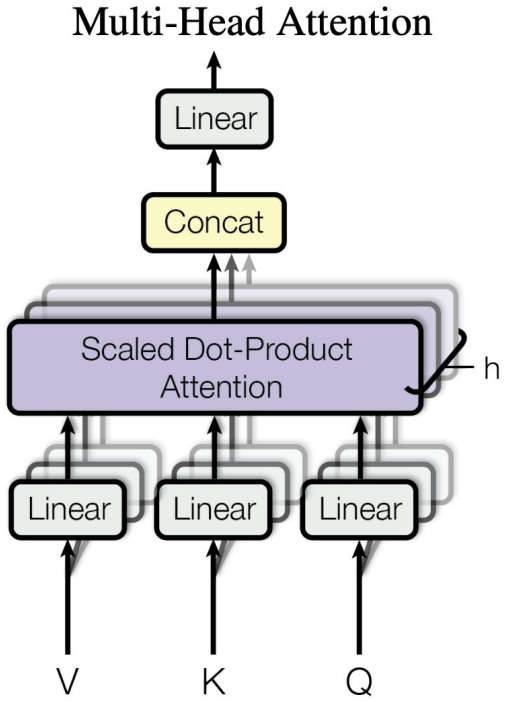
Multi-Head Attention

- Idea:
  a. Stack linear layers (weight matrices without biases) that are independent each for keys, queries, values.
  b. Concatenate output of attention heads to form (plus non-linearity) output layer
- Why?
  a. Allows for model to focus on different positions
  b. Gives attention layer multiple “representation subspaces”
  c. No longer need to oversaturate one attention mechanism

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$h = 8$ parallel attention layers, or heads.

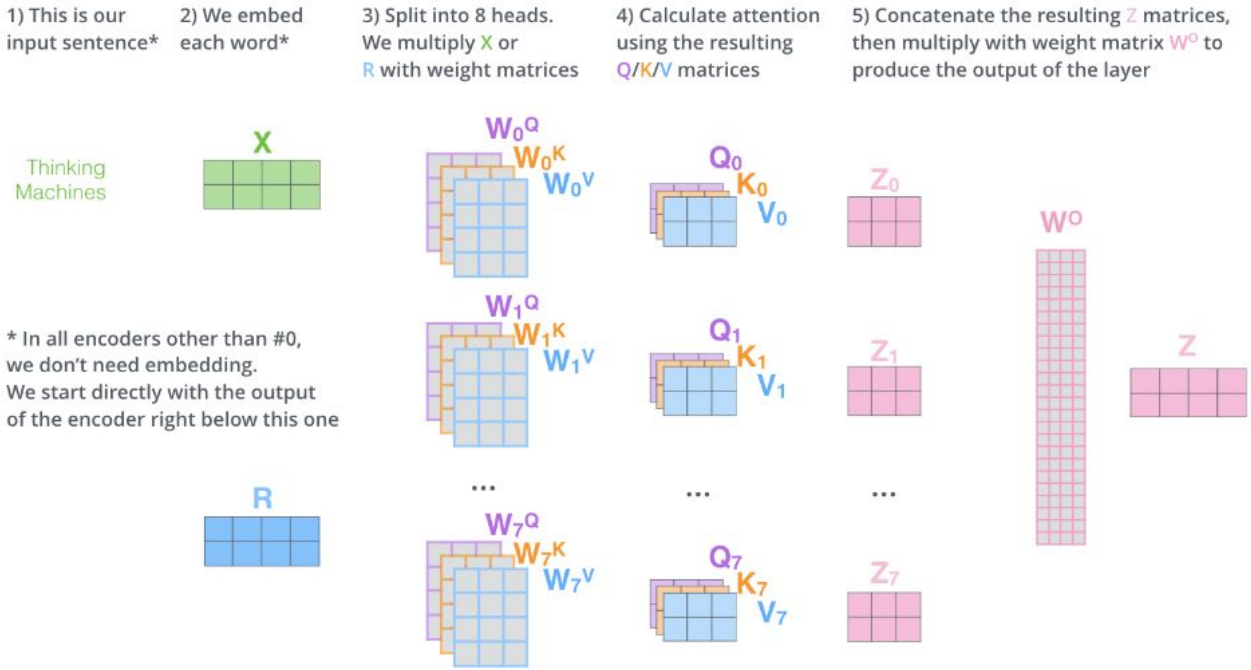
Learnable parameter matrices



Multi-Head Attention Continued

Key points:

- We calculate 8 different attention heads, but we need to combine them
- Attention heads are independent of each other



https://jalammar.github.io/illustrated-transformer/

Positional Encoding

- Need for information about the position and order of tokens in a sequence
- Positional Embedding: Vector that represents position of each token
  - Element wise addition the position embedding to the word embedding vector
  - Positional embedding be fixed or learned

"We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PEpos+k can be represented as a linear function of PEpos."

Encoding used in paper

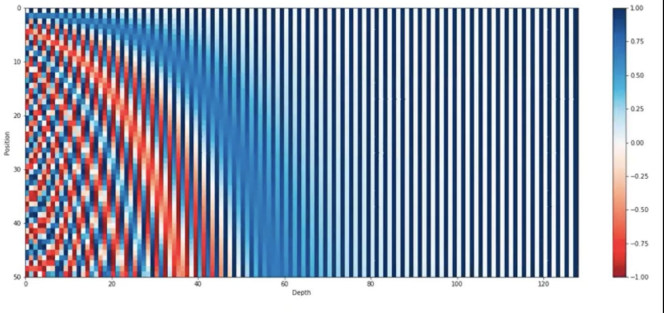
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

pos is the position and i is the dimension.

* Sine method:

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \vdots \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$



For every sine-cosine pair corresponding to frequency ω_k, there is a linear transformation $M \in \mathbb{R}^{2\times 2}$ (independent of t) where the following equation holds:

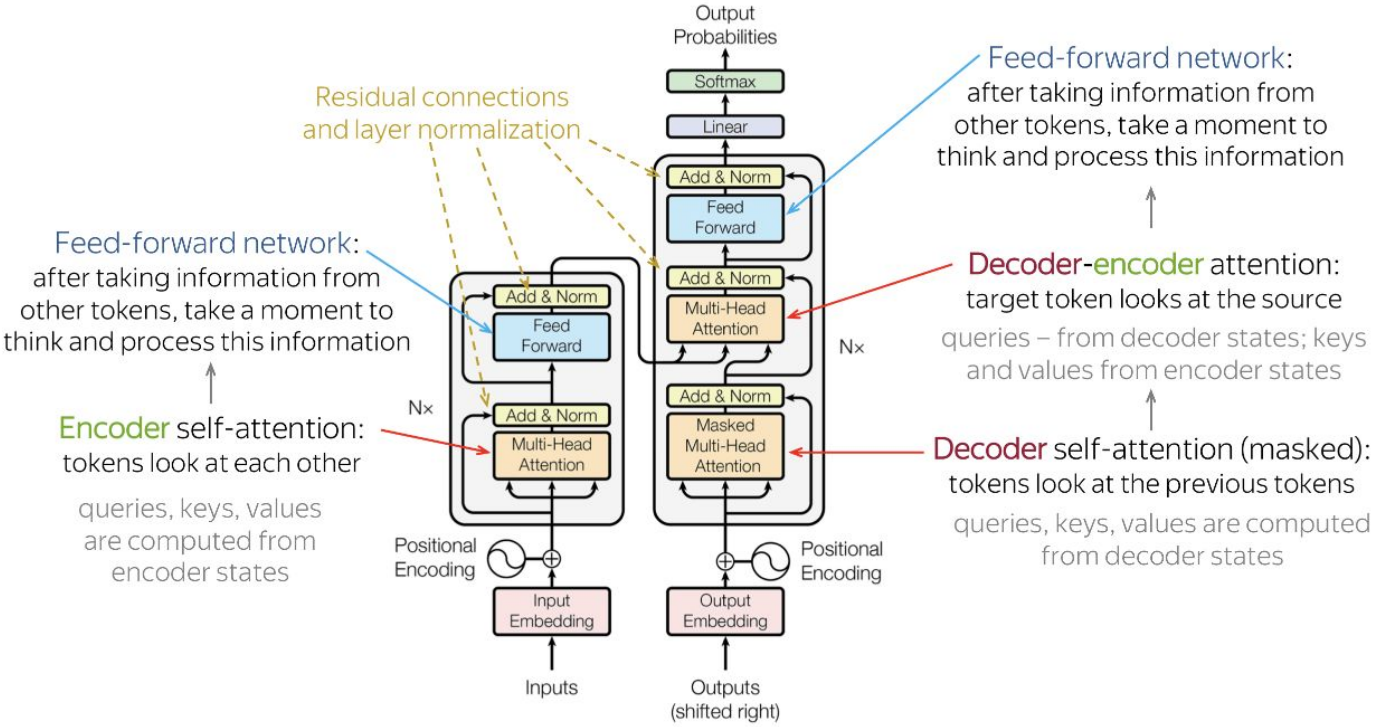
=>

$$M . \begin{bmatrix} \sin(\omega_k . t) \\ \cos(\omega_k . t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k . (t+\phi)) \\ \cos(\omega_k . (t+\phi)) \end{bmatrix}$$

$$M_{\phi,k} = \begin{bmatrix} \cos(\omega_k . \phi) & \sin(\omega_k . \phi) \\ -\sin(\omega_k . \phi) & \cos(\omega_k . \phi) \end{bmatrix}$$

Note: positional embeddings are now learned

Full-Architecture



Training Details and Experimental Setup

- Hardware Details:
  - Trained on 8 NVIDIA P100 GPU
  - The big models were trained for 300,000 steps (3.5 days)
- Datasets:
  - Trained on WMT 2014 English-to-German translation task
    - 4.5 million sentence pairs
  - Trained on WMT 2014 English-to-French translation task
    - 36 million sentences
  - Sentences encoded using byte-pair encoding
  - Sentence pairs were batched together by approximate sequence length

Experimental Results

- New single-model state-of-the-art BLEU score of 41.8 for English-to-French
  - Over 2 BLEU improvement
- 28.4 BLEU on the WMT 2014 English-to-German translation task
- Over 100x less training loss
- Generalizes well to other task

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Discussion of Results

- Extremely efficient for parallelization (great for GPU)
- State-of-the-art performance
- Authors understood significance of work:
    - In conclusion, authors state desire to “investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video”
- Unlike RNN, ability to directly learn dependencies

Critique / Limitations / Open Issues

- Baseline transformers use O(n^2) in memory/computation.
  - increasing size of length increases computation quadratically

- Fixed length unlike RNN of arbitrary length
- Inability to process input sequentially (not like how the human brain works)
- Paper addressing limitation of transformers: https://arxiv.org/pdf/2002.09402.pdf
  - Limited access to long memory
  - Limited ability to update state.

Future Work for Paper / Reading

Bigger and bigger transformers

- NLP:
  - Masked Language Models
    - BERT - Bidirectional Transformers for NLP
    - RoBERTa
  - GPT-3
- Computer Vision:
  - Vision Transformer (ViT)
  - Swin Transformers

Extended Readings

Good Sources for Explainability:

- **Blogs:**
  - https://jinglescode.github.io/2020/05/27/illustrated-guide-transformer/
  - https://towardsdatascience.com/transformers-141e32e69591
  - https://jalammar.github.io/illustrated-transformer/
  - https://medium.com/analytics-vidhya/neural-machine-translation-using-bahdanau-attention-mechanism-d496c9be30c3
- **Youtube:**
  - https://www.youtube.com/watch?v=TQQlZhbC5ps

Relevant Papers:

- Attention is All You Need: https://arxiv.org/abs/1706.03762
- BERT: https://arxiv.org/abs/1810.04805
- Roberta: https://arxiv.org/abs/1907.11692
- Swin Transformer: Hierarchical Vision Transformer using Shifted Windows: https://arxiv.org/abs/2103.14030

Summary

- Transformers:
  - Dominate Everything
  - State of the art in Image Classification, Language Modeling, Speech Recognition, Vision Transformers (ex: ViT), Semantic Segmentation, Behavior / Decision Making, etc.

Image Classification on ImageNet

Top 1 Accuracy	Top 5 Accuracy	Number of params	GFLOPs	Top-5 Error	Top-1 Error	Extra Training Data	Paper	Code	Result	Year	Tags
91.0%		2100M				✓	CoCa: Contrastive Captioners are Image-Text Foundation Models			2022	Transformer, JFT-3B, ALIGN
90.98%		2440M				✓	Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time			2022	Conv+Transformer, JFT-3B, ALIGN
90.94%		1843M				✓	Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time			2022	Transformer, JFT-3B
90.88%		2440M	2586			✓	CoAtNet: Marrying Convolution and Attention for All Data Sizes			2021	Conv+Transformer, JFT-3B
							CoCa: Contrastive				

Transformers