



M2177.003100

Deep Learning

[11: Attention and Transformer]

Electrical and Computer Engineering
Seoul National University

© 2019 Sungroh Yoon. this material is for educational uses only. some contents are based on the material provided by other paper/book authors and may be copyrighted by them.

(last compiled at 23:20:00 on 2019/11/05)

Outline

Attention Mechanism

Transformer

Applications

Summary

References

- online resources:
 - ▶ *A Tutorial on Attention in DL (ICML 2019)* [▶ Link](#)
 - ▶ *Stanford CS224n: NLP with Deep Learning* [▶ Link](#)
 - ▶ *Attn: Illustrated Attention* [▶ Link](#)
 - ▶ *Attention? Attention!* [▶ Link](#)
 - ▶ *The Annotated Transformer* [▶ Link](#)
 - ▶ *Attention and Augmented RNNs* [▶ Link](#)
 - ▶ *The Illustrated Transformer* [▶ Link](#)
- note:
 - ▶ you should [open this file in Adobe Acrobat](#) to see animated images
(other types of pdf readers will not work)

Outline

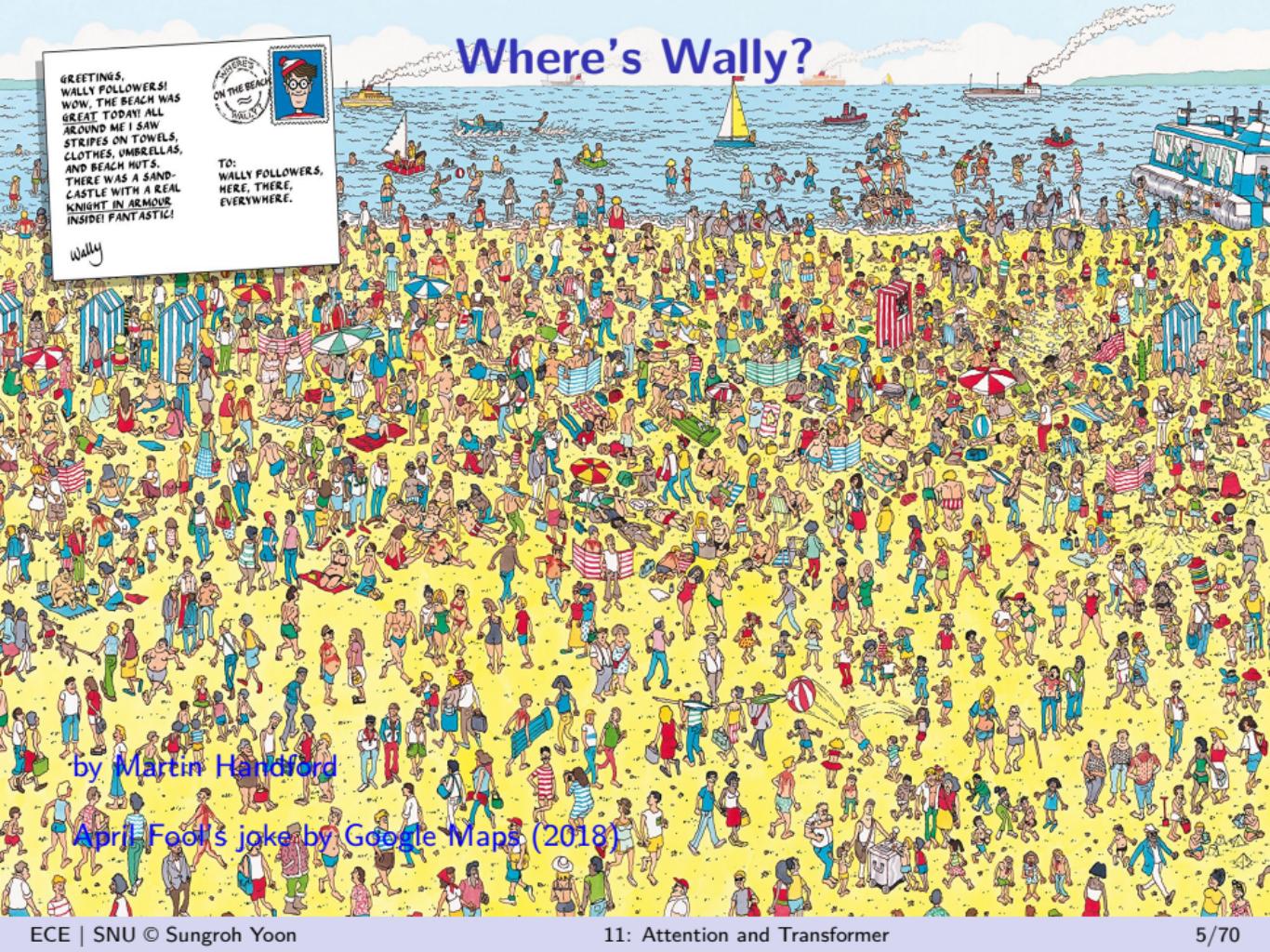
Attention Mechanism

Transformer

Applications

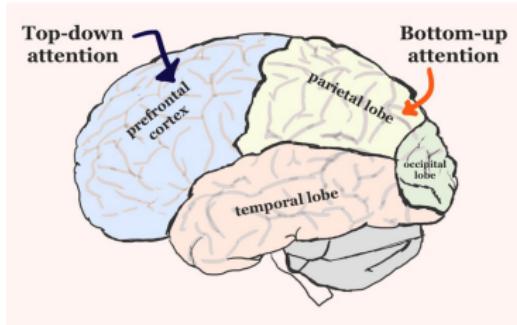
Summary

Where's Wally?



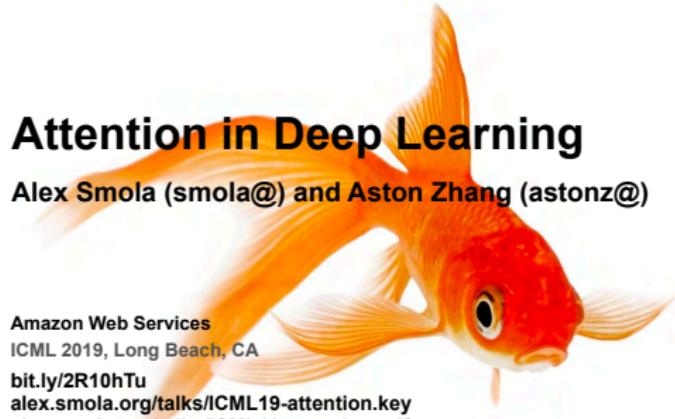
Attention

- literally (Google dictionary)
 - ▶ n. notice taken of someone or something; the regarding of someone or something as interesting or important.
- biologically (Wikipedia)
 - ▶ behavioral and cognitive process of *selectively concentrating* on a discrete aspect of information, while *ignoring other* perceivable information
 - ▶ for _____ saving



(source: MacKay, The science of staying focused, 2017)

- in machine learning
 - ▶ an iterative process
 - ▶ applicable to input, memory, and representation
 - ▶ _____ trainable (soft attention)
 - ▶ ICML 2019 tutorial: [▶ Link](#)



Attention in Deep Learning

Alex Smola (smola@) and Aston Zhang (astonz@)

Amazon Web Services

ICML 2019, Long Beach, CA

bit.ly/2R10hTu

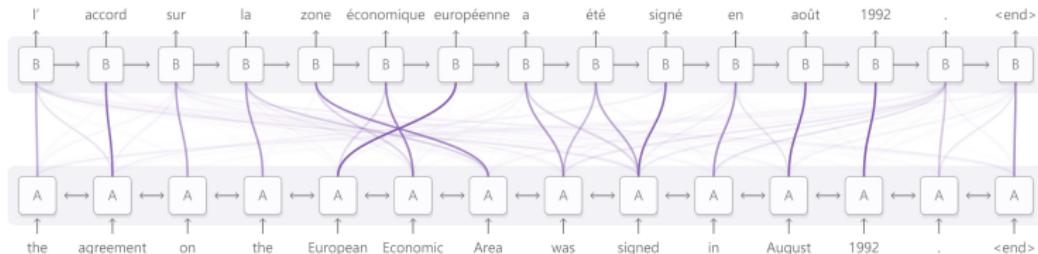
alex.smola.org/talks/ICML19-attention.key

alex.smola.org/talks/ICML19-attention.pdf

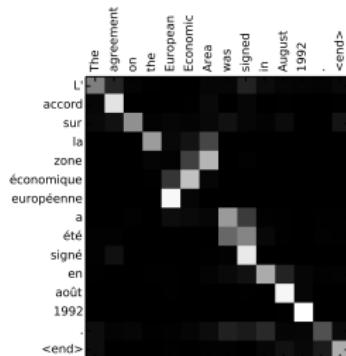


Attention in deep learning

- NLP (neural machine translation)



► attention \Rightarrow learned _____ (between **source** and **target** sentences)



source: Attention and Augmented RNNs; Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: arXiv preprint arXiv:1409.0473 (2014)

- computer vision



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.

(source: Xu et al.¹)

- trend + key papers

2014 recurrent models of **visual attention**²

2014–15 attention in neural machine translation³

2015–16 attention-based RNN/CNN in NLP

2017 **self-attention**⁴

¹ Kelvin Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *International conference on machine learning*. 2015, pp. 2048–2057

² Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. "Recurrent models of visual attention". In: *Advances in neural information processing systems*. 2014, pp. 2204–2212

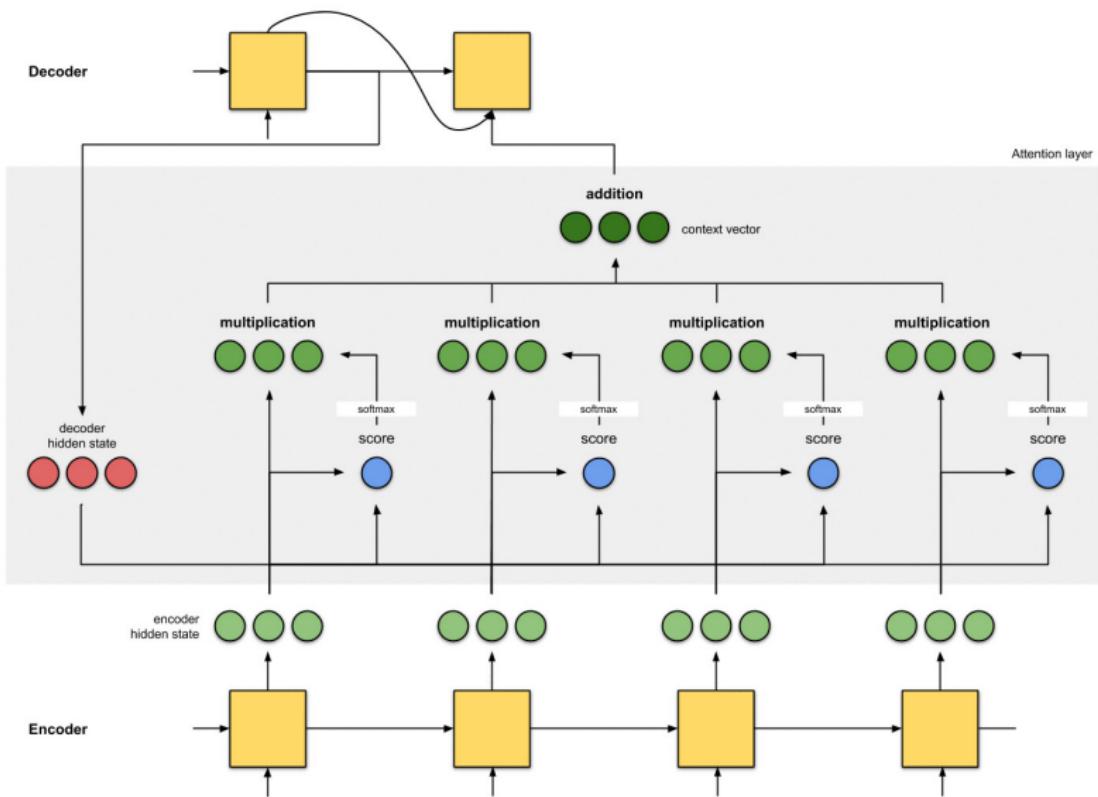
³ Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014)

⁴ Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008

Recall: sequence-to-sequence model

- basic components
 - ▶ encoder RNN + decoder RNN
- seq2seq (**w/o attention**)
 - ▶ encoder RNN: input encoding + context vector computing
 - ⇒ may be overloaded ⇒ suboptimal performance
- seq2seq **with attention**
 - ▶ encoder RNN: input encoding
 - ▶ attention: context vector computing
 - ⇒ offloads encoder RNN ⇒ better performance

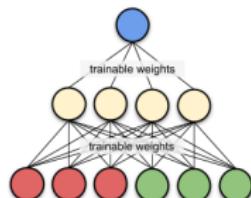
- seq2seq with attention [source](#)



Attention scoring

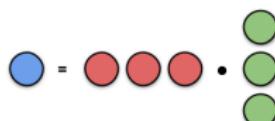


Additive / Concat



$$a = \mathbf{v}^\top \tanh(\mathbf{W}[\mathbf{s}; \mathbf{h}])$$

Dot product



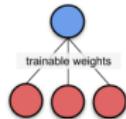
$$a = \mathbf{s}^\top \mathbf{h}$$

Scaled dot product



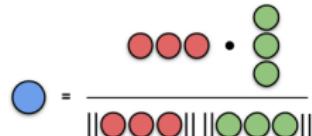
$$a = \frac{1}{\sqrt{n}} \mathbf{s}^\top \mathbf{h}, \text{ where } n = \|\mathbf{h}\|$$

Location-based



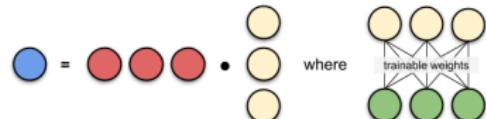
$$a = \text{softmax}(\mathbf{W}\mathbf{s})_i$$

Cosine similarity



$$a = \frac{\mathbf{s}^\top \mathbf{h}}{\|\mathbf{s}\| \cdot \|\mathbf{h}\|}$$

General

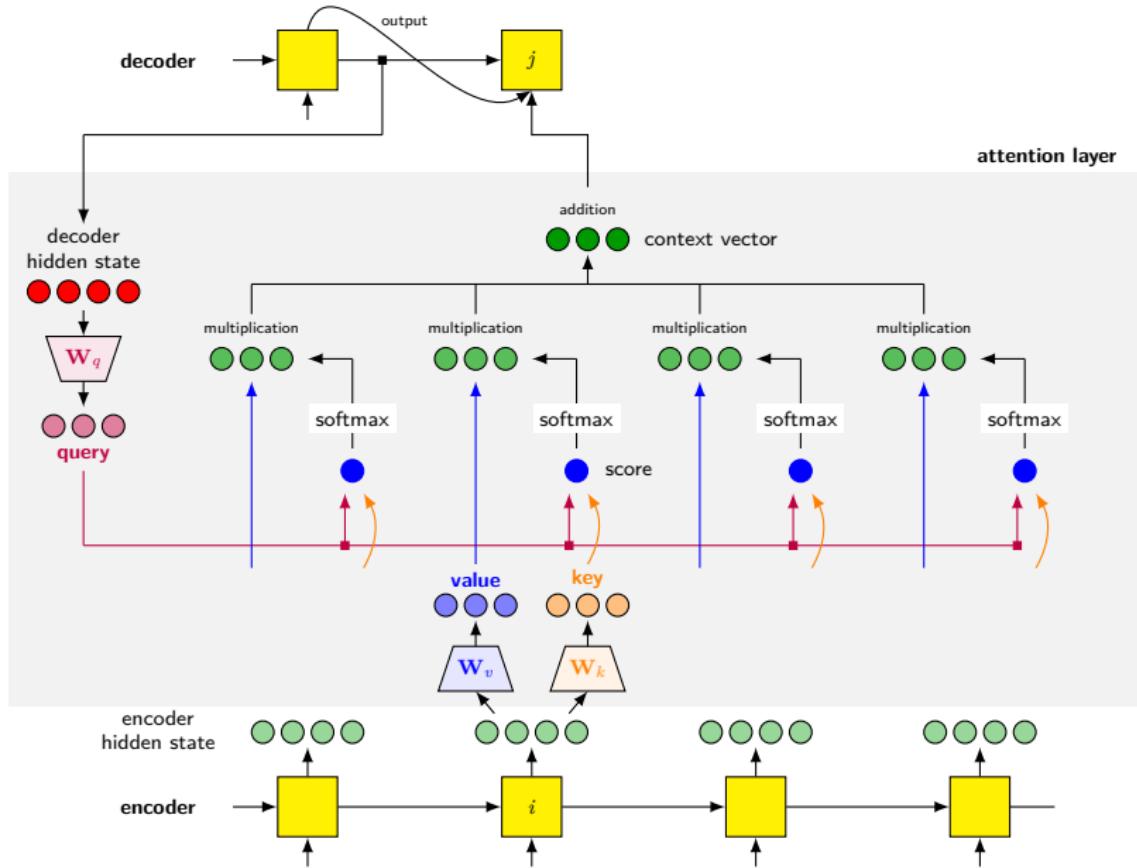


$$a = \mathbf{s}^\top \mathbf{W}\mathbf{h}$$

(source: Attention? Attention!, Attn: Illustrated Attention)

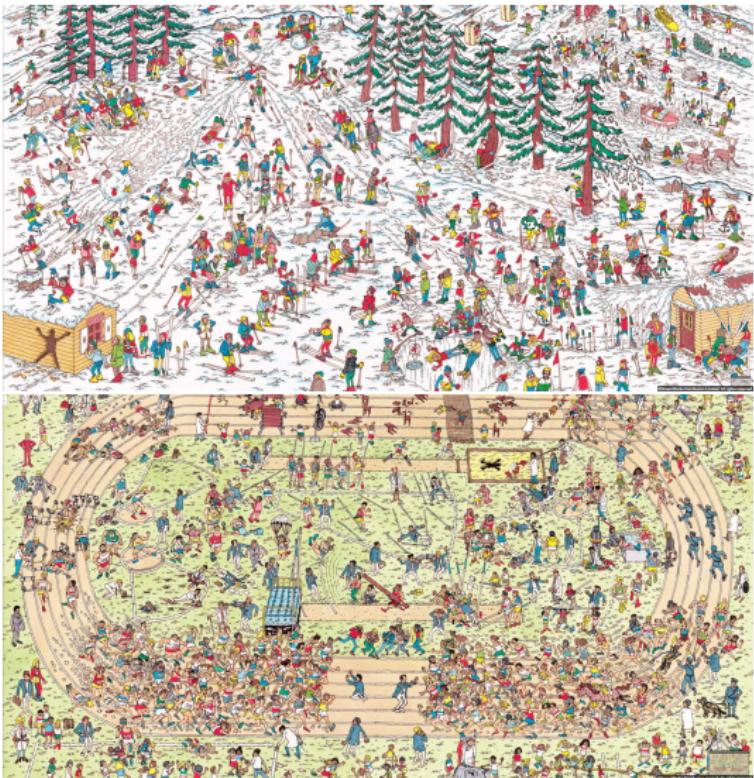
Key-value description of attention

- three types of vectors
 - ▶ **query** = (projected) decoder hidden state
 - ▶ **key** = (projected) encoder hidden state (for attn weight computation)
 - ▶ **value** = (projected) encoder hidden state (for context vector buildup)
- outcome
 - ▶ context vector = $\underbrace{\text{weighted sum of } ___ \text{ s}}_{\text{each weight} = \text{function of (query, key)}}$
- idea: separation of encoder hidden state into
 - ▶ $\underbrace{\text{attention information}}_{\text{key}} \text{ and } \underbrace{\text{content information for context vector}}_{\text{value}}$



Taxonomy of attention mechanisms

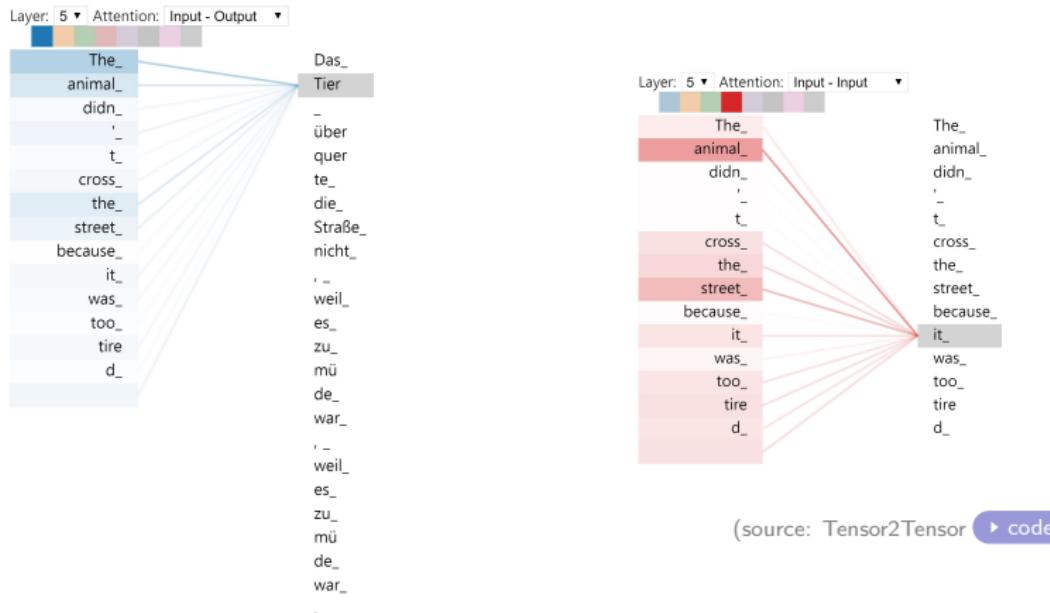
- cross vs self
- soft vs hard
- global vs local



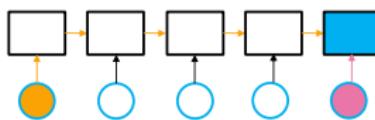
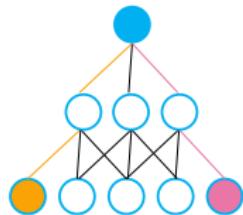
(source: DreamWorks)

Self-attention

- (cross-)attention: works on different sequences
- self-attention (*aka* intra-attention)
 - ▶ relates different positions of a _____ seq to compute its representation

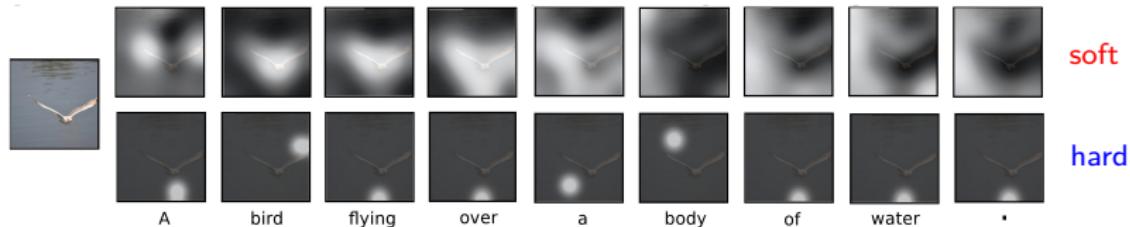


- comparison
 - ▶ CNN vs RNN vs self-attention



Soft vs hard attention

example: image caption generation⁵



- **soft** attention: deterministic
 - ▶ attn weights: learned and placed “softly” over all patches in source image
 - ▶ context = \sum (attention map \times image feature map) \Rightarrow “average”
- **hard** attention: _____
 - ▶ only (randomly) selects patch(es) of the image to attend to at a time
 - ▶ context = {feature(s) randomly sampled wrt attention map} \Rightarrow “subset”

⁵Xu et al., “Show, attend and tell: Neural image caption generation with visual attention”

comparison:

- **soft** attention

- 😊 model is smooth and _____

- 😢 slow (heavy computation) when source input is large

- **hard** attention

- 😊 fast inference (less computation)

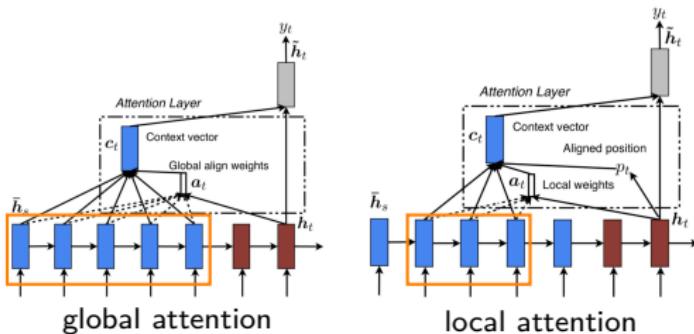
- 😢 model is non-differentiable (stochastic)

- ⇒ requires complicated techniques (*e.g.* variance reduction, RL) to train

Global vs local attention

first proposed in machine translation (source sentence → target sentence)

- **global** attention: basically soft attention
 - ▶ to generate a target word, consider all source words
- **local** attention⁶: blend of soft and hard (to make hard attention differentiable)
 - ▶ to generate a target word, first predicts a source word position
 - ▷ then uses a _____ around this position to compute the target word



⁶ Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation". In: *arXiv preprint arXiv:1508.04025* (2015)

Outline

Attention Mechanism

Transformer

Overview

Self-Attention

Multi-Head Attention

Positional Encoding

Residuals and Layer Norm

Decoding

Applications

Summary



Transformer

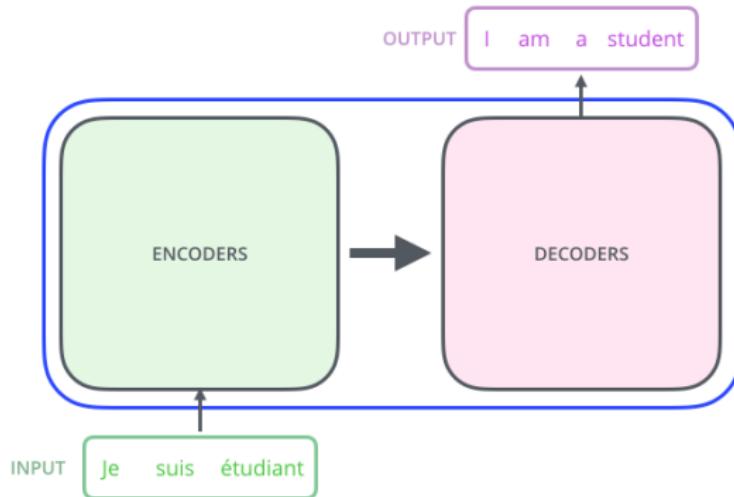
A high-level look

- a **sequence-to-sequence** model
 - ▶ input sequence (arbitrary length) → output sequence (arbitrary length)



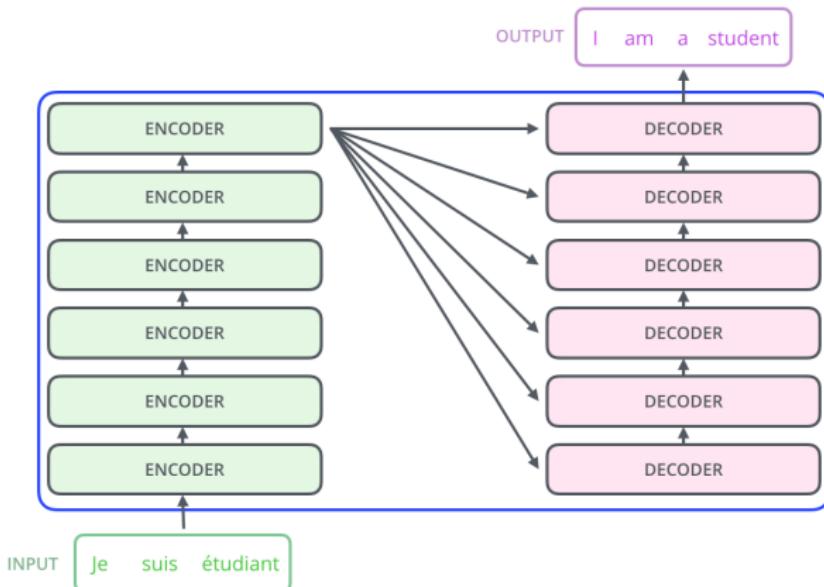
(source: The Illustrated Transformer)

- encoder-decoder architecture
 - ▶ an **encoding** component + a **decoding** component
 - ▶ and connections between them



(source: The Illustrated Transformer)

- **encoding** component: a stack of encoders⁷
- **decoding** component: a stack of decoders of the same number

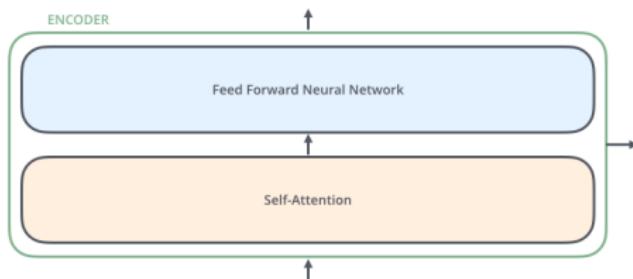


(source: The Illustrated Transformer)

⁷the original transformer paper stacks six of them on top of each other; there is nothing magical about the number six, and one can definitely experiment with other arrangements.

Encoder

- encoders: all identical in structure (yet they do not share weights)
 - ▶ each one is broken down into two sub-layers:

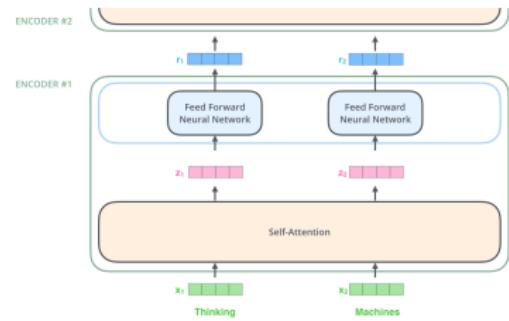
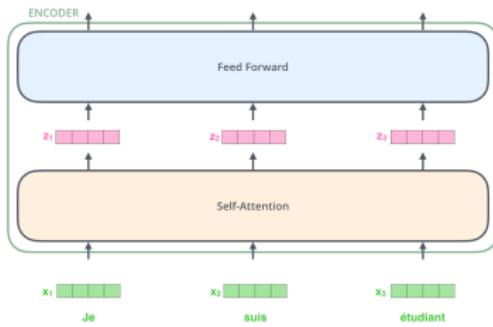


(source: The Illustrated Transformer)

- **self-attention** layer
 - ▶ helps the encoder look at other words in the input sentence as it _____ a specific word
- **feed-forward** neural net layer
 - ▶ the exact same feed-forward net is independently applied to each position

- word embedding

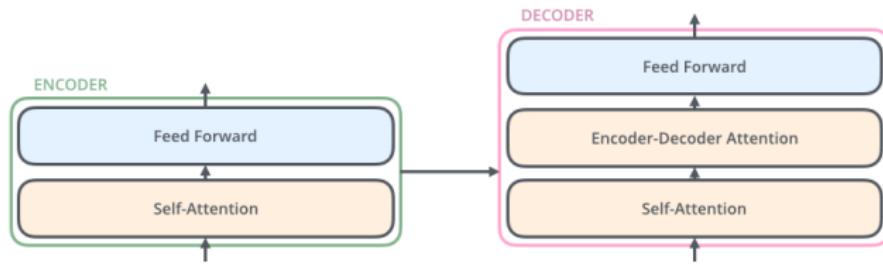
- ▶ we begin by turning each input word into a vector
- ▶ only happens in the _____-most encoder
- ▶ embedding vector size in the transformer paper: 512



(source: The Illustrated Transformer)

Decoder

- **decoder:** has three sub-layers
 - ▶ feed-forward layers
 - ▶ **encoder-decoder** attention layer
 - ▶ self-attention layer



(source: The Illustrated Transformer)

- **encoder-decoder** attention layer
 - ▶ helps the decoder focus on relevant parts of the _____ sentence
(similar to what attention does in seq2seq models)

Outline

Attention Mechanism

Transformer

Overview

Self-Attention

Multi-Head Attention

Positional Encoding

Residuals and Layer Norm

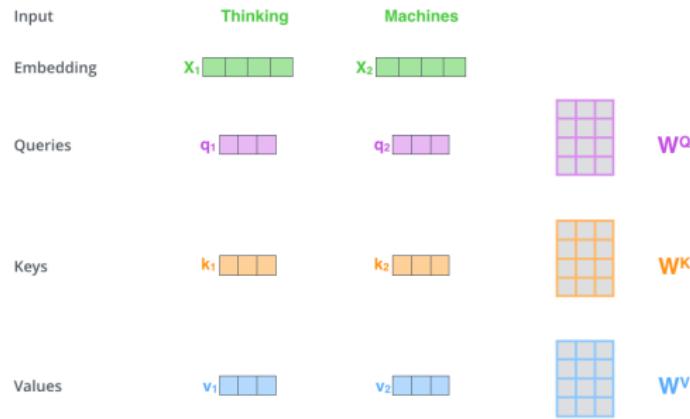
Decoding

Applications

Summary

Step 1

- create three vectors from each of the encoder's input (embedding) vectors
 - ▶ **query**, **key**, **value**
 - ▶ how? multiply embedding vectors by three trainable matrices

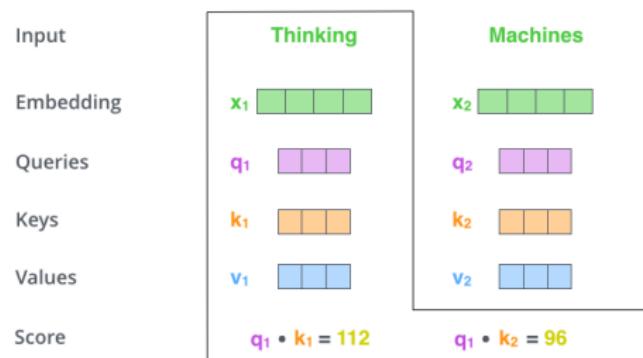


(source: The Illustrated Transformer)

- size of **query**/**key**/**value** vectors: _____ than embedding
 - ▶ in the original paper: 64 vs 512

Step 2

- calculate a self-attention score for each word
 - ▶ the score determines how much focus to place on other parts of the input as we encode a word at a certain position
- how? take a dot product:
 - ▶ score = (query vector) \cdot (key vector of the word we're scoring)



(source: The Illustrated Transformer)

Step 3

- normalize attention scores

(3a) divide each score by $\sqrt{\text{key vector size}} = \sqrt{d_k}$ (original paper: $\sqrt{64} = 8$)

- leads to having more stable gradients

(3b) softmax operation

- gives positional _____ (which sum to 1)

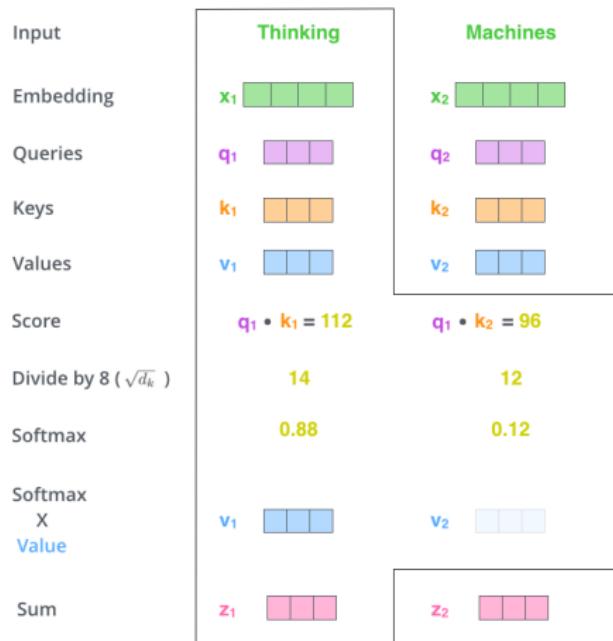
Input	Thinking	Machines
Embedding	x_1 [green green green green]	x_2 [green green green green]
Queries	q_1 [purple purple purple]	q_2 [purple purple purple]
Keys	k_1 [orange orange orange]	k_2 [orange orange orange]
Values	v_1 [blue blue blue]	v_2 [blue blue blue]
Score	$q_1 \cdot k_1 = 112$	$q_1 \cdot k_2 = 96$
Divide by 8 ($\sqrt{d_k}$)	14	12
Softmax	0.88	0.12

(source: The Illustrated Transformer)

Step 4

- calculate the output of self-attention layer

↑
weighted sum of value vectors



(source: The Illustrated Transformer)

Matrix calculation of self-attention

- can process all embedding vectors simultaneously
 - ▶ by stacking them as a _____

$$\begin{array}{ccc} \mathbf{X} & \mathbf{WQ} & \mathbf{Q} \\ \text{Thinking Machines} \quad \begin{matrix} \text{[} \text{[} \text{[} \text{[} \text{]} \end{matrix} & \times \quad \begin{matrix} \text{[} \text{[} \text{[} \text{[} \text{]} \end{matrix} & = \quad \begin{matrix} \text{[} \text{[} \text{[} \text{[} \text{]} \end{matrix} \\ \\ \mathbf{X} & \mathbf{WK} & \mathbf{K} \\ \begin{matrix} \text{[} \text{[} \text{[} \text{[} \text{]} \end{matrix} & \times \quad \begin{matrix} \text{[} \text{[} \text{[} \text{[} \text{]} \end{matrix} & = \quad \begin{matrix} \text{[} \text{[} \text{[} \text{[} \text{]} \end{matrix} \\ \\ \mathbf{X} & \mathbf{WV} & \mathbf{V} \\ \begin{matrix} \text{[} \text{[} \text{[} \text{[} \text{]} \end{matrix} & \times \quad \begin{matrix} \text{[} \text{[} \text{[} \text{[} \text{]} \end{matrix} & = \quad \begin{matrix} \text{[} \text{[} \text{[} \text{[} \text{]} \end{matrix} \end{array}$$
$$\text{softmax}\left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V}$$
$$= \quad \mathbf{Z}$$

(source: The Illustrated Transformer)

Outline

Attention Mechanism

Transformer

Overview

Self-Attention

Multi-Head Attention

Positional Encoding

Residuals and Layer Norm

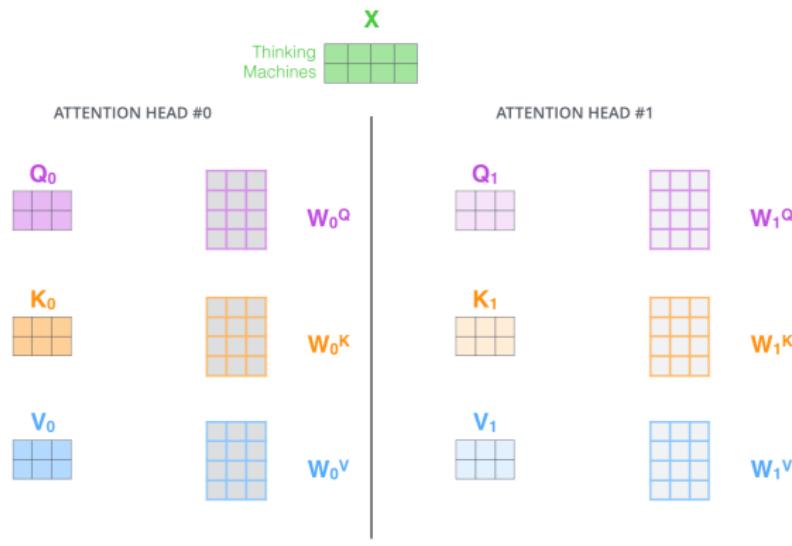
Decoding

Applications

Summary

Multi-head attention

- further refinement of self-attention layer: use **multiple** attention _____
 - ▶ expands the model's ability to focus on **different positions**
 - ▶ gives the attention layer **multiple representation subspaces**

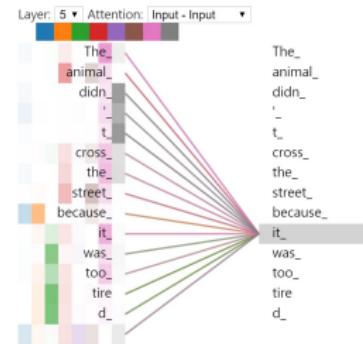
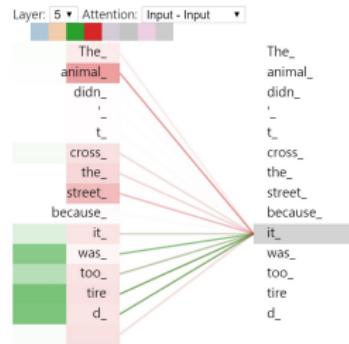
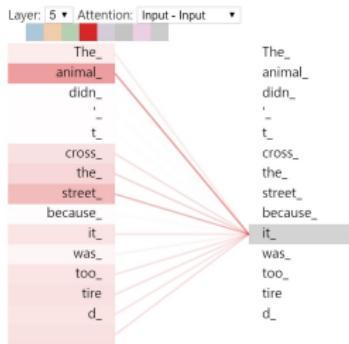


(source: The Illustrated Transformer)

- different attention heads

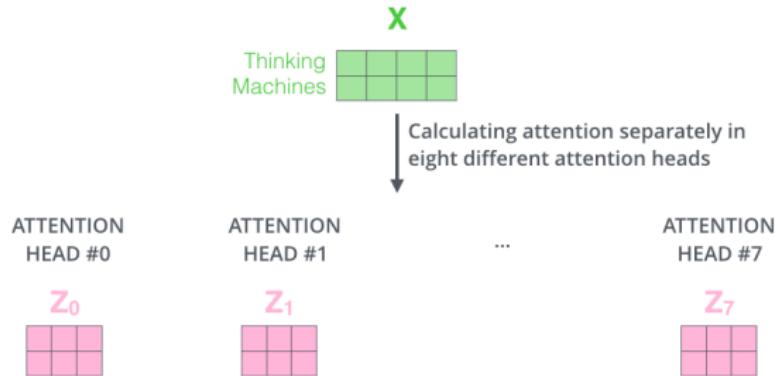
► focus differently as we encode the word “it”

e.g. 1 head, 2 heads, and 8 heads



(source: Tensor2Tensor [▶ code](#))

- the original transformer paper
 - ▶ 8 attention heads



(source: The Illustrated Transformer)

Condensing multi-head attention results

- **concatenate** heads and **multiply** with a learnable weight matrix

1) Concatenate all the attention heads



2) Multiply with a weight matrix W^o that was trained jointly with the model

\times



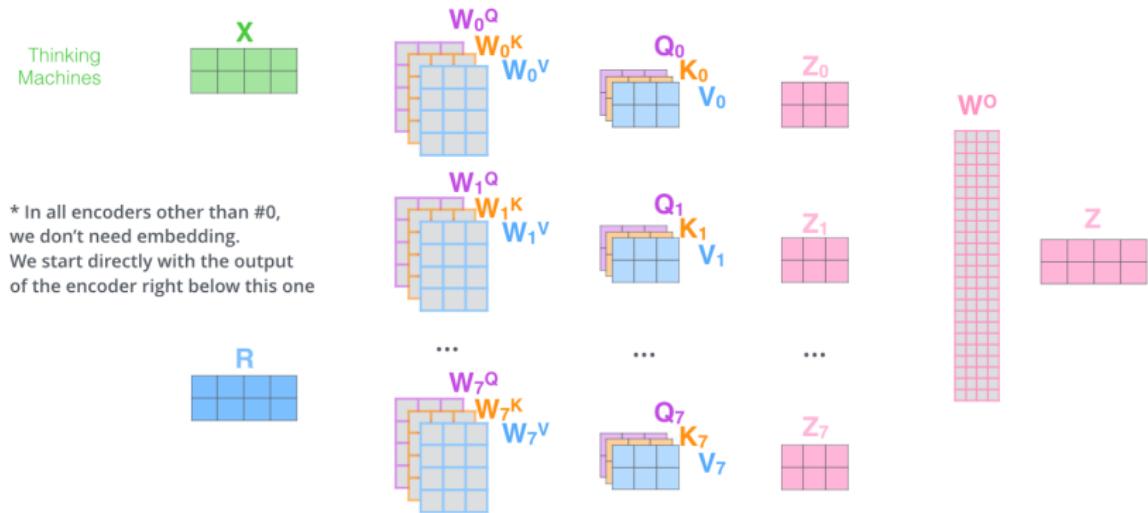
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

$$= \begin{matrix} Z \\ \begin{matrix} \text{---} & \text{---} & \text{---} & \text{---} \end{matrix} \end{matrix}$$

(source: The Illustrated Transformer)

Recap

- 1) This is our input sentence*
Thinking Machines
- 2) We embed each word*
 X
- 3) Split into 8 heads.
We multiply X or R with weight matrices
- 4) Calculate attention using the resulting $Q/K/V$ matrices
- 5) Concatenate the resulting Z matrices, then multiply with weight matrix W^o to produce the output of the layer



(source: The Illustrated Transformer)

Outline

Attention Mechanism

Transformer

Overview

Self-Attention

Multi-Head Attention

Positional Encoding

Residuals and Layer Norm

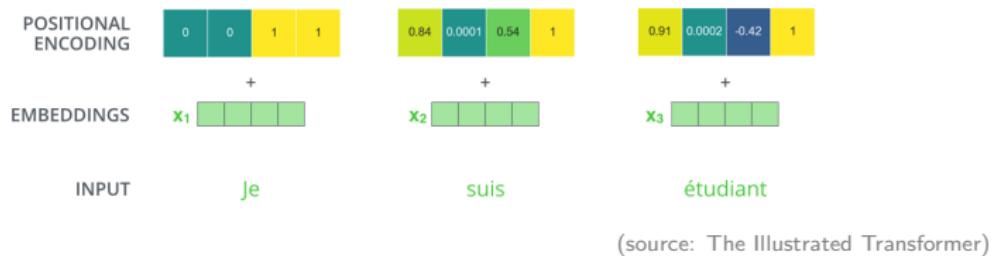
Decoding

Applications

Summary

Representing the order of words

- one thing missing from the model we have described so far
 - ▶ a way to represent the _____ of the words in the input sequence
- solution: add a vector (“**positional encoding**”) to each input embedding

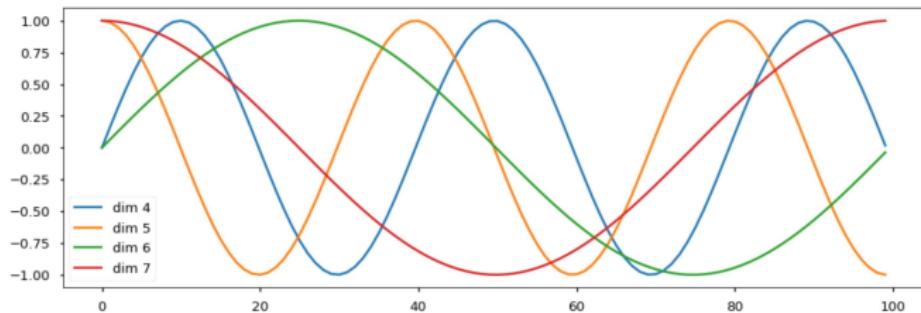


(source: The Illustrated Transformer)

- positional encoding vectors: follow a specific pattern the model learns
 - ▶ helps the model to determine
 - ▶ **position** of each word or
 - ▶ **distance** between different words in input

Positional patterns

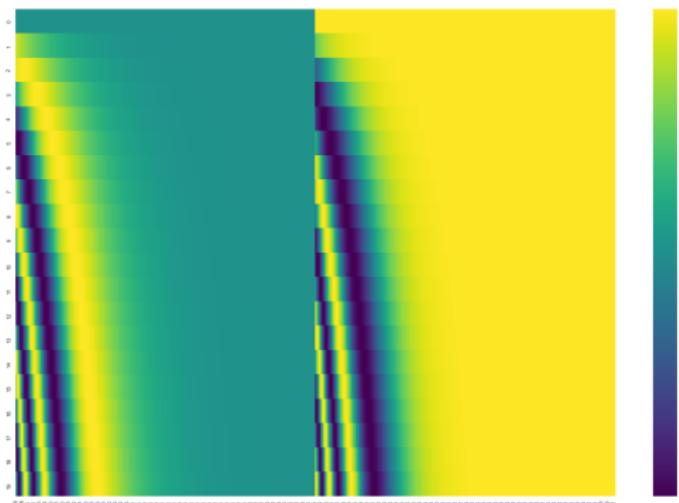
- many choices for such a pattern: **learned** or **fixed**
- example: adding a *sine wave* based on position
 - ▶ frequency and offset of the wave: different for each dimension



(source: The Annotated Transformer)

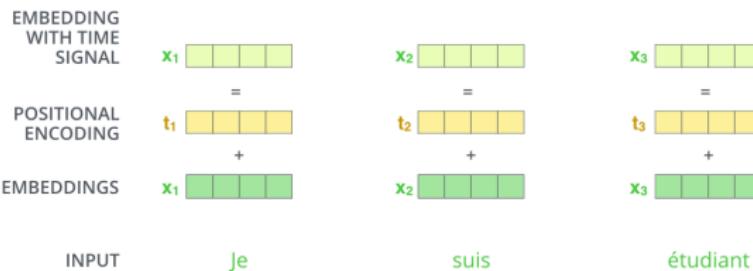
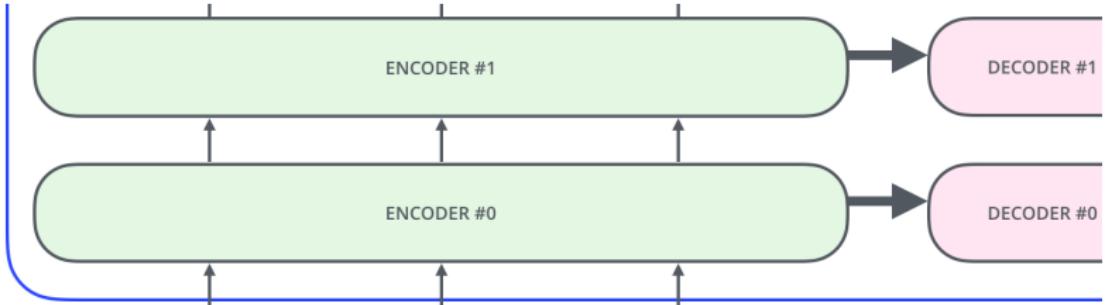
- **positional encoding (PE)** in the original transformer

- ▶ 512 dimensions: 256 (sine) + 256 (cosine)



(source: The Illustrated Transformer)

- each row
 - ▶ a PE of a vector
- each column
 - ▶ PE dimension
- shown left
 - ▶ 20 words
 - ▶ 512 dimensions



INPUT

je

suis

étudiant

(source: The Illustrated Transformer)

Outline

Attention Mechanism

Transformer

Overview

Self-Attention

Multi-Head Attention

Positional Encoding

Residuals and Layer Norm

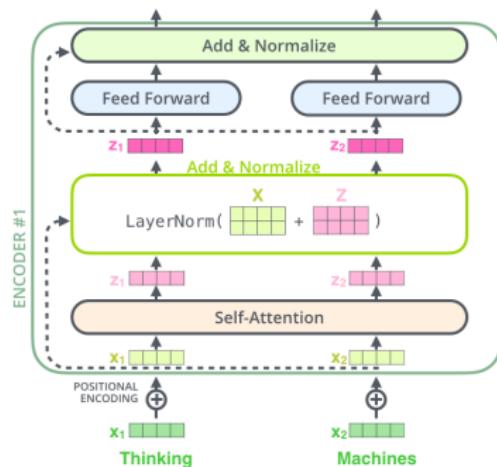
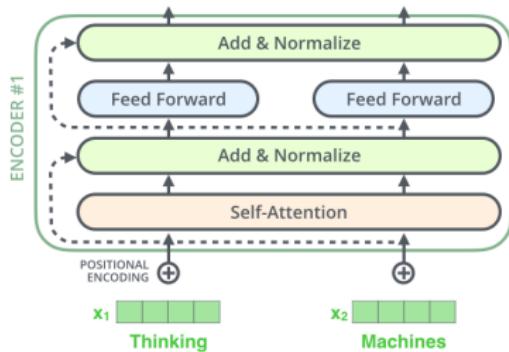
Decoding

Applications

Summary

Add and normalize

- each sublayer (self-attention and feed-forward) in encoder
 - (1) has a _____ connection around it and
 - (2) is followed by layer normalization⁸



(source: The Illustrated Transformer)

⁸Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016)

Outline

Attention Mechanism

Transformer

Overview

Self-Attention

Multi-Head Attention

Positional Encoding

Residuals and Layer Norm

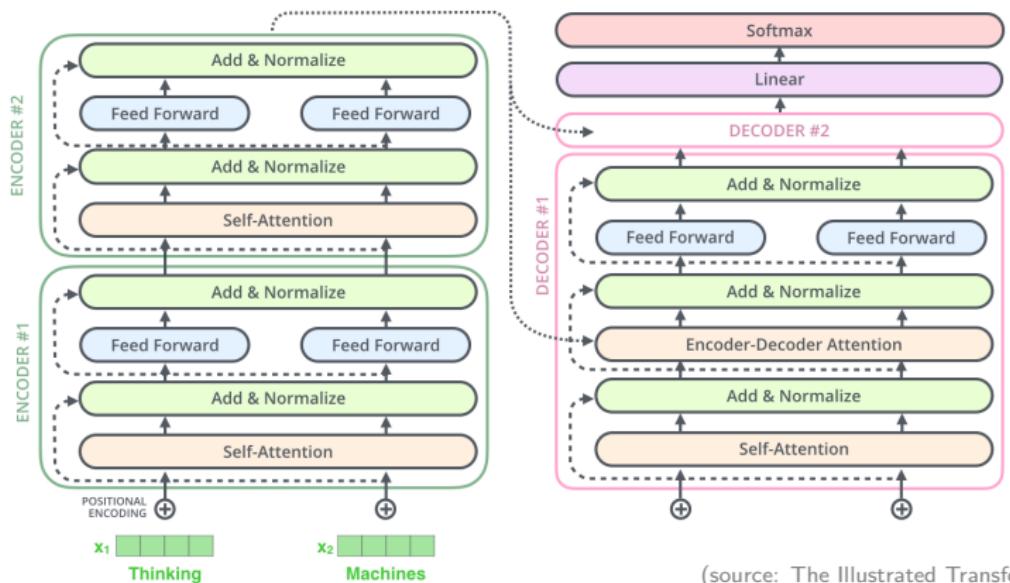
Decoding

Applications

Summary

Decoder architecture

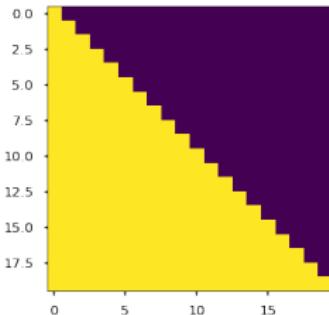
- overall, similar to encoder architecture
- **difference #1:** encoder-decoder attention (multi-head ____-attention)
 - ▶ helps the decoder focus on appropriate places in input sequence



- difference #2: modified self-attention (_____ multi-head self-attention)
 - ▶ for autoregressive decoding

$$P(y_1, y_2, \dots, y_T | \mathbf{x}) = \prod_{t=1}^T p(y_t | y_1, y_2, \dots, y_{t-1}, \mathbf{x})$$

- ▶ need to prevent positions from attending to **subsequent positions**
- ▶ example:

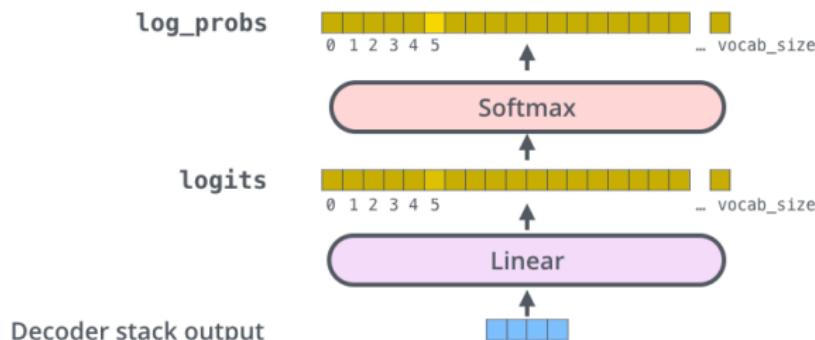


- ← attention mask shows
 - ▶ the position each target word (row) is allowed to look at (column)
- words are blocked for
 - ▶ attending to **future words** during training

(source: The Annotated Transformer)

Logits and softmax layers

- linear layer: float vector → logits⁹
decoder output length: _____ size (very large)
- softmax layer: logits → probability vector



(source: The Illustrated Transformer)

* training loss: cross-entropy loss

⁹a vector of raw (non-normalized) predictions that a classification model generates, which is ordinarily then passed to a normalization function

Output symbol generation

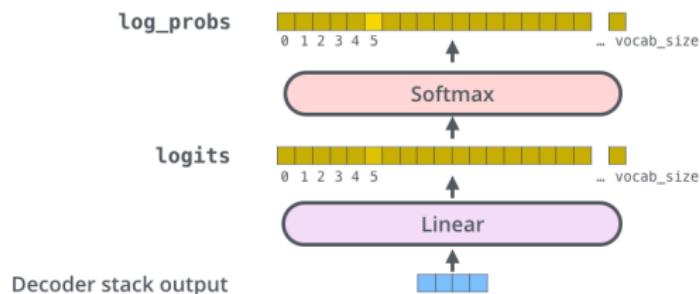
- method #1: _____ decoding
 - ▶ output the word with the highest probability

Which word in our vocabulary
is associated with this index?

am

Get the index of the cell
with the highest value
(argmax)

5



(source: The Illustrated Transformer)

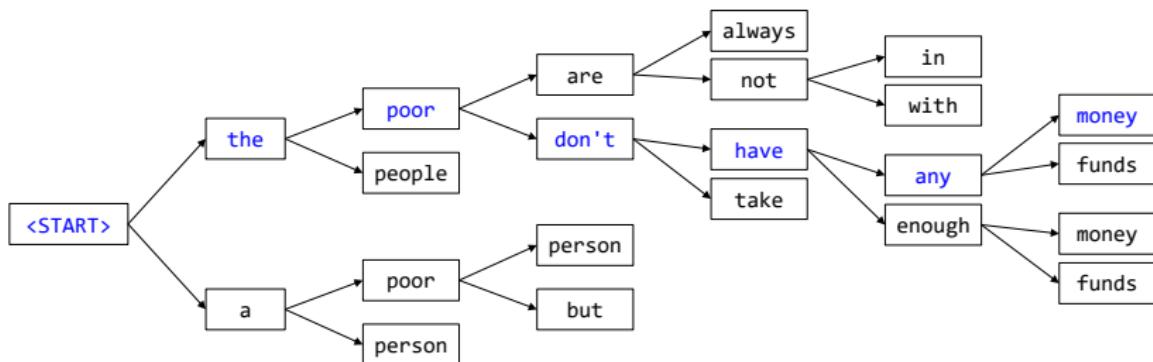
- method #2: _____ search (hyperparameters: beam size k , top beams t)¹⁰

- ▶ in a decoding step, consider k choices
- ▶ run k different models simultaneously
- ▶ in the next step, return t most probable paths; continue

e.g. $k = t = 2$

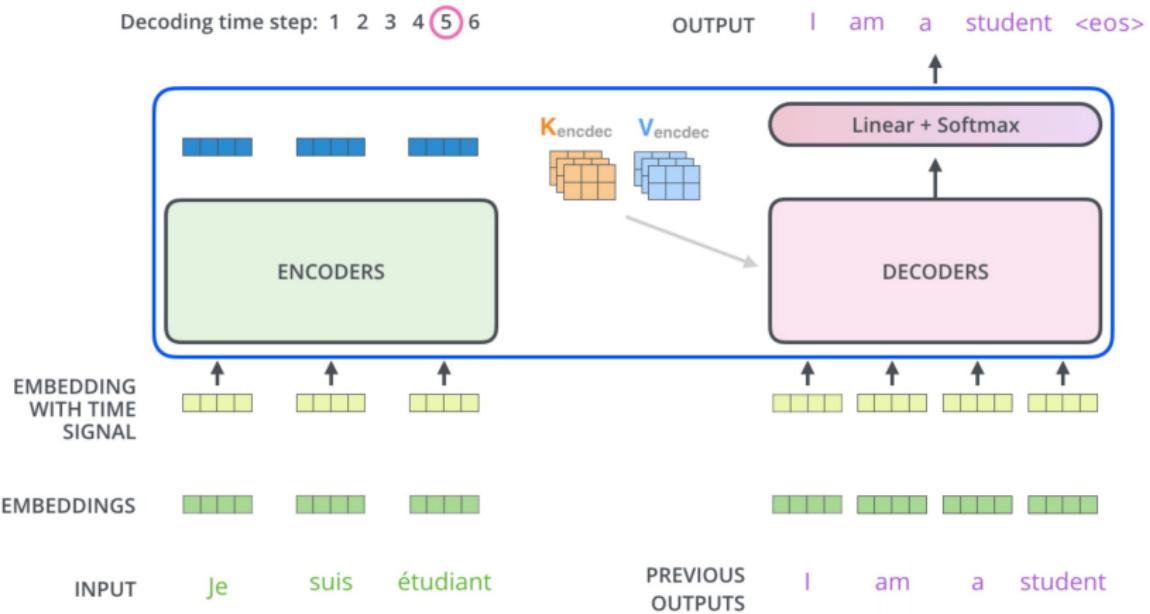
source: les pauvres sont démunis

target: the poor don't have any money



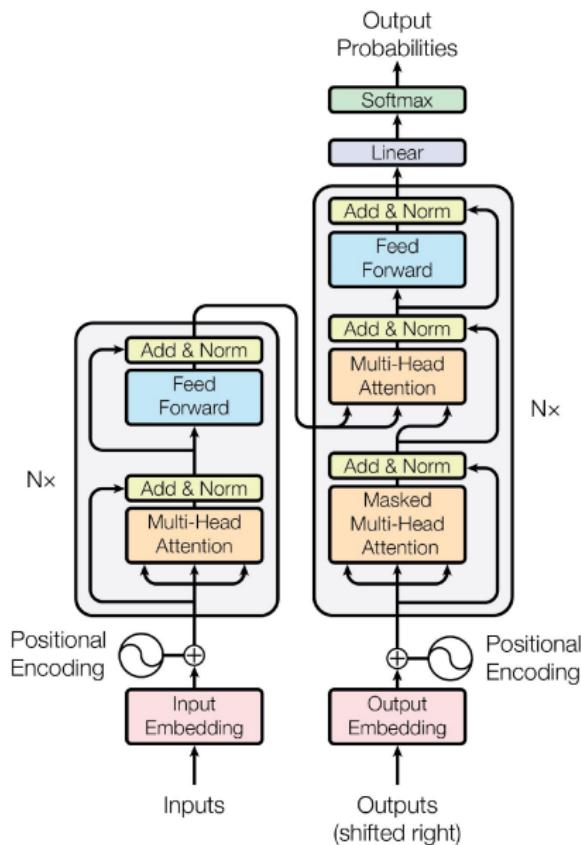
¹⁰ often set $k = t$; $k = t = 1$ corresponds to greedy decoding

Altogether



(source: The Illustrated Transformer)

The transformer



(source: Vaswani et al., 2017¹¹)

¹¹Vaswani et al., "Attention is all you need"

Outline

Attention Mechanism

Transformer

Applications

Summary

BERT (Bidirectional Encoder Representations from Transformers¹²)

- a language representation model
 - ▶ can pre-train deep _____ representations from unlabeled text
 - ▶ by jointly conditioning on both left and right context in all layers
- **pre-trained** BERT can be **fine-tuned** with just one additional output layer
 - ▶ gives state-of-the-art models for various tasks (*e.g.* question answering)
 - ▶ without substantial task-specific architecture modifications
- in a sense, a **denoising autoencoder** for NLP

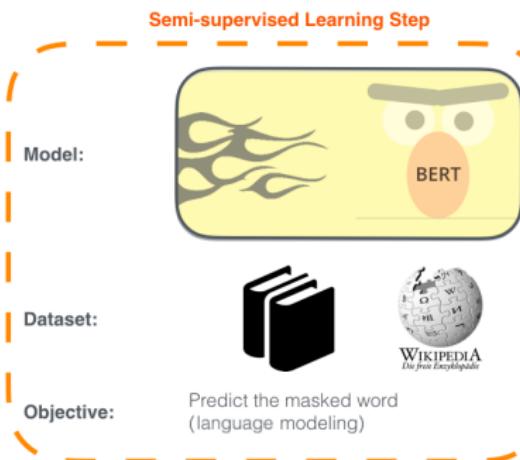


¹² Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: arXiv preprint arXiv:1810.04805 (2018)

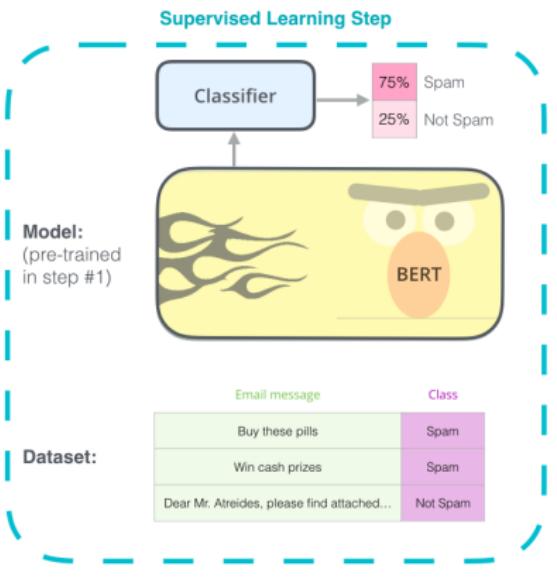
- pre-training → fine-tuning

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.



2 - Supervised training on a specific task with a labeled dataset.



(source: Trivedi)

Defining a prediction goal

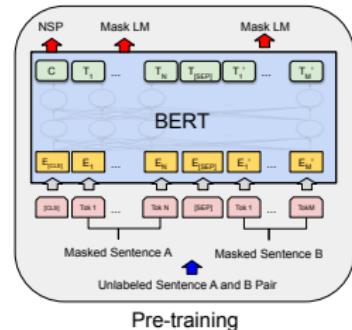
- a major challenge in training language models
- many models
 - ▶ predict the **next word** in a sequence
- e.g. The child came home from ___
 - ▶ a directional approach \Rightarrow inherently limits context learning

- BERT uses two “pre-training” strategies to overcome this challenge

1. **language model (MLM)**

2. **next sentence prediction (NSP)**

- ▶ MLM and NSP are trained together, minimizing a combined loss function



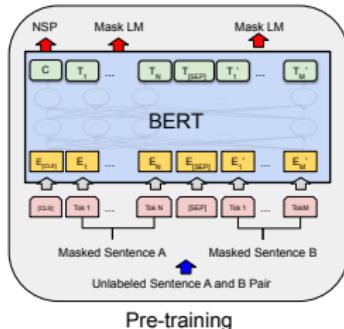
Pre-training

(source: Devlin et al., 2018)

Masked LM (MLM)

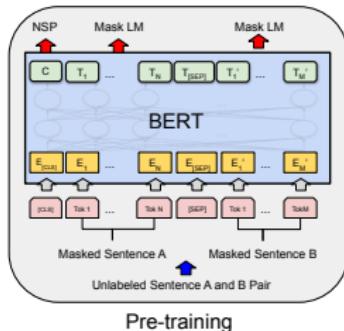
during training:

- before feeding word sequences into BERT
 - ▶ select 15% of words in each sentence, and then
 - ▷ 80%: replace with [MASK] token
 - ▷ 10%: replace with random words
 - ▷ 10%: do nothing
- the model attempts to predict the original value of the masked words
 - ▶ based on the context provided by the other, _____ words



(source: Devlin et al., 2018)

Next sentence prediction (NSP)

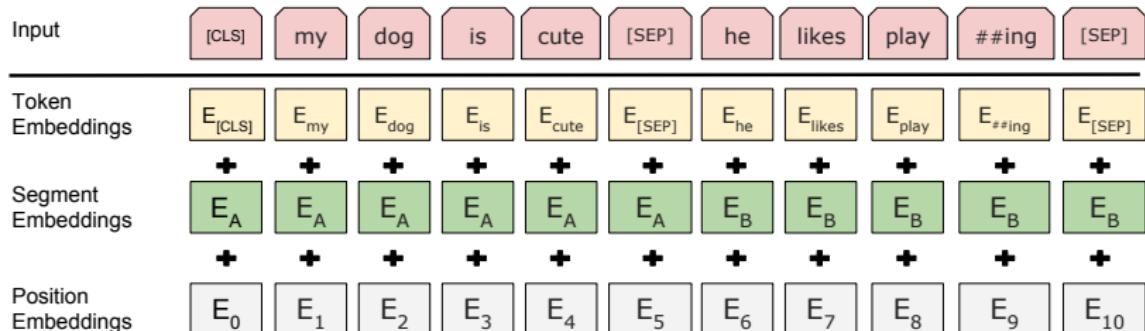


during training:

(source: Devlin et al., 2018)

- the model receives **two sentences** as input and learns to predict
 - ▶ if the second sentence is the _____ sentence in the original document
- training set composition
 - ▶ 50%: pairs of sentences actually appearing in the original document
 - ▶ 50%: random pairs of sentences chosen from the corpus

- input pre-processing¹³ for NSP
 1. [CLS] token: inserted at the beginning of 1st sentence
 2. [SEP] token: inserted at the end of each sentence
 3. _____ embedding (to indicate each sentence) is added to each token
 4. positional embedding is added to each token

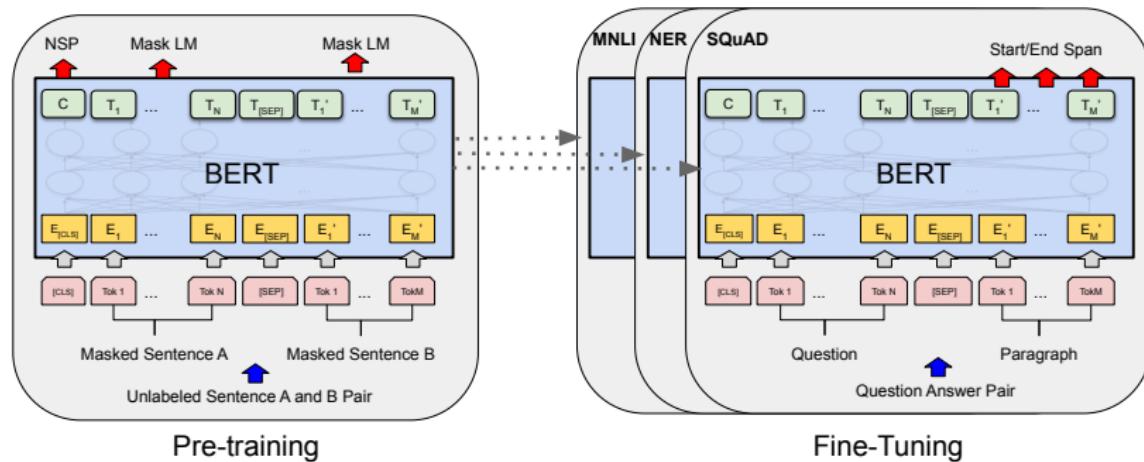


(source: Devlin et al., 2018)

¹³ [CLS] and [SEP] refer to classification and separation, respectively

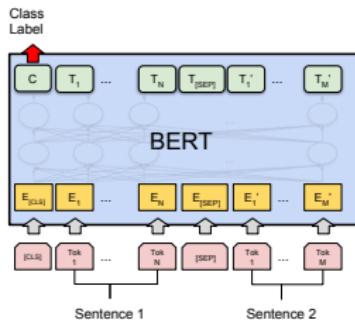
How to use BERT (fine-tuning)

- can be used for various NLP tasks (only adding a small layer to the core)
 - ▶ classification
 - ▶ question answering (QA)
 - ▶ named entity recognition (NER)

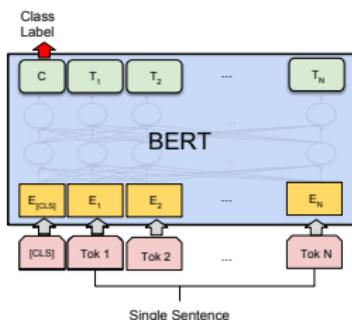


(source: Devlin et al., 2018)

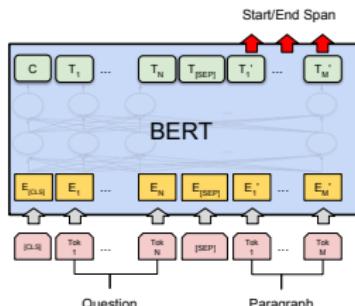
- fine-tuning BERT on different tasks



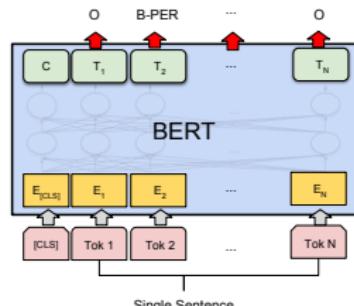
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1

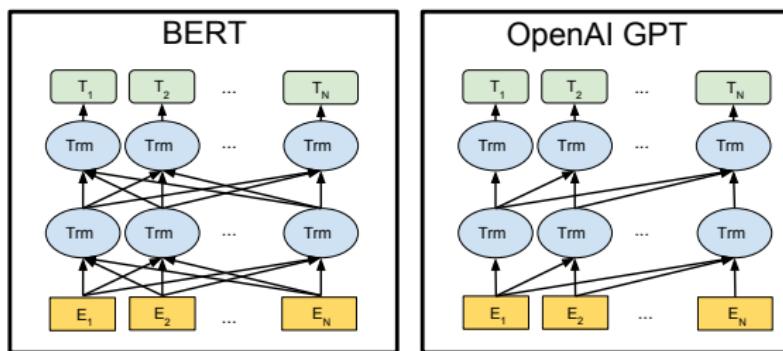


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

(source: Devlin et al., 2018)

OpenAI GPT (Generative Pre-trained Transformer¹⁴)

- similar idea
 - ▶ unsupervised **pre-training** + transformer
- key difference
 - ▶ BERT: bidirectional transformer
 - ▶ GPT: _____ transformer

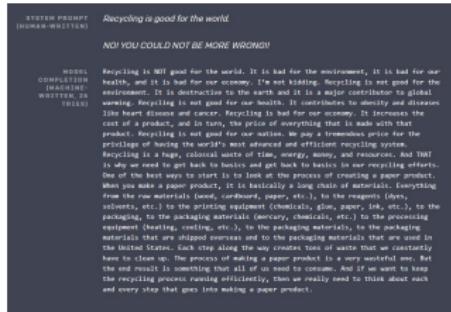


(source: Devlin et al., 2018)

¹⁴ Alec Radford et al. "Improving language understanding by generative pre-training". In: *OpenAI Blog* (2018)

OpenAI GPT-2¹⁵

- direct scale-up version of GPT
 - ▶ 1.5 billion parameters (> 10x) trained with 8 million webpages (> 10x)



Home > Technology > Elon Musk's OpenAI builds artificial intelligence so powerful it must be kept...

Technology

Elon Musk's OpenAI builds artificial intelligence so powerful it must be kept locked up for the good of humanity

February 15, 2019



(source: The Verge, 2019)

¹⁵ Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI Blog* (2019)

XLNet¹⁶

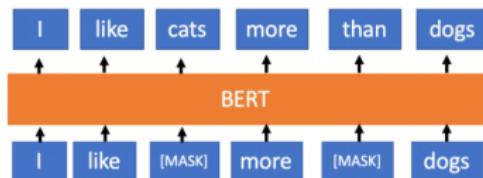
- BERT limitations
 - ▶ neglects dependency between masked positions
 - ▶ suffers from a pretrain-finetune discrepancy
- XLNet: a generalized autoregressive pretraining method
 - ▶ based on **transformer-XL** (autoregressive model: **RNN** + **transformer**)
 - ▶ learns bidirectional contexts
 - ▷ by considering all _____ of the factorization order

(source: Kurita)

¹⁶ Zhilin Yang et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *arXiv preprint arXiv:1906.08237* (2019)

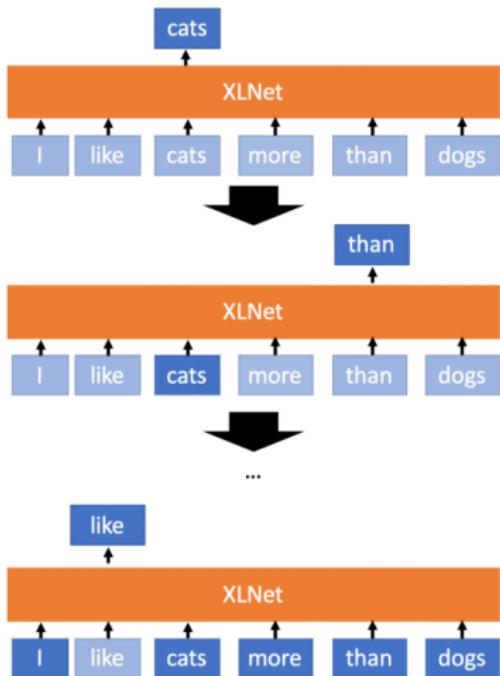
comparison:

- traditional LM



- BERT¹⁷

- XLNet¹⁸



(source: Kurita)

¹⁷ predicts all masked words simultaneously

¹⁸ learns to predict the words in an arbitrary order but in an autoregressive, sequential manner (not necessarily left-to-right)

Outline

Attention Mechanism

Transformer

Applications

Summary

Summary

- “attention is all you need”: attention in deep learning
 - ▶ produces state-of-the-art results in natural language processing tasks
 - ▶ self/cross-attention, soft/hard attention, global/local attention
 - ▶ potentially more efficient/effective in temporal modeling than CNN/RNN
- the transformer
 - ▶ a sequence-to-sequence model having encoder-decoder architecture
 - ▶ encoder: self-attention + feed-forward layers
 - ▶ decoder: masked self-attention + cross-attention + feed-forward layers
 - ▶ multi-head attention, residual connection, layer norm
- applications of transformer
 - ▶ powerful language representation models: BERT, OpenAI GPT/GPT-2
 - ▶ BERT: pre-training (MLM + NSP) → fine-tuning
 - ▶ XLNet: based on transformer-XL (RNN + transformer)