
# M2177.003100
# Deep Learning

## [12: Attention and Transformer]

Electrical and Computer Engineering
Seoul National University



(last compiled at 17:03:00 on 2020/11/01)

# Outline

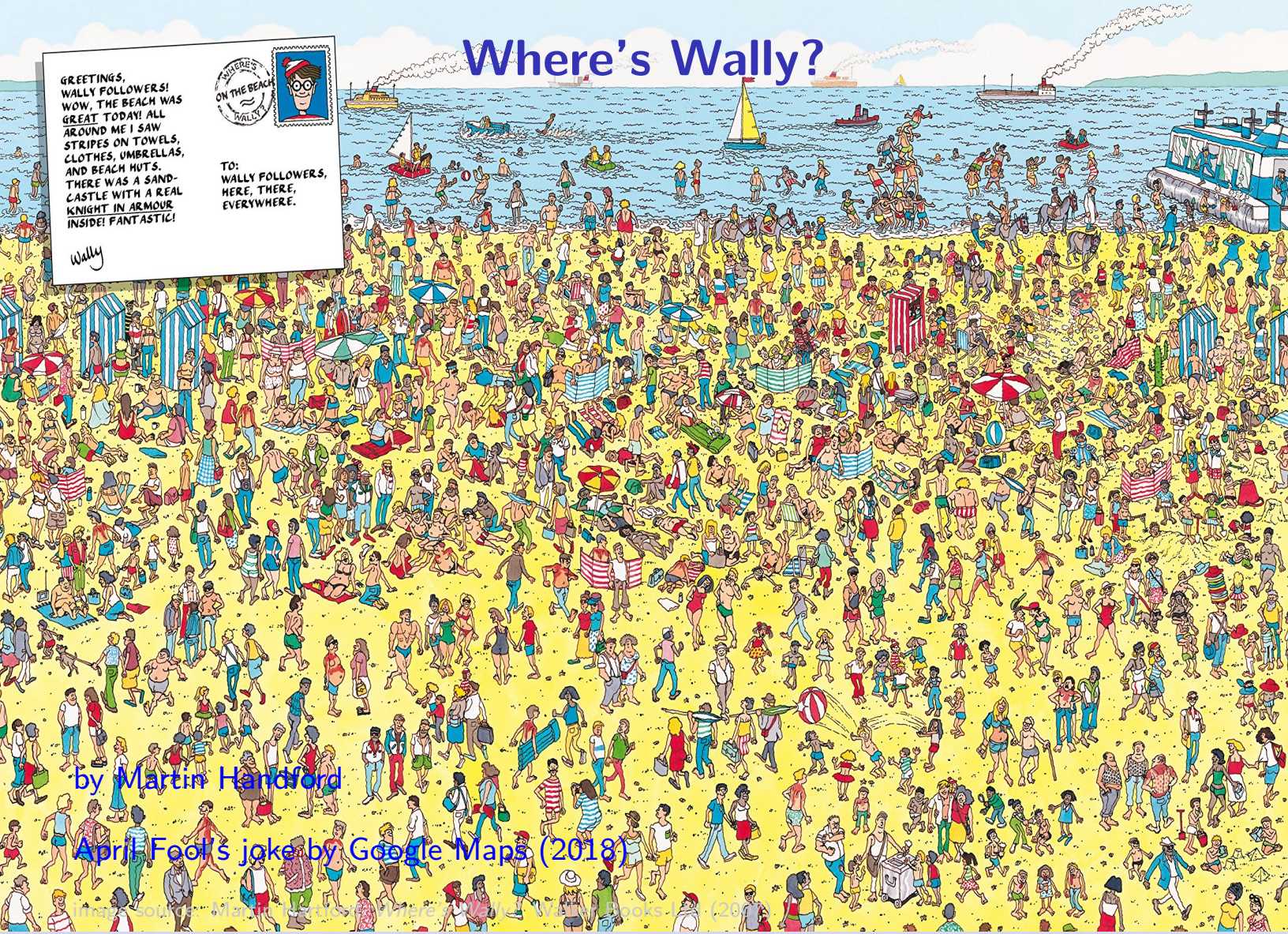
# References

- 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

# Where's Wally?




by Martin Handford

April Fool's joke by Google Maps (2018)

Image source: Martin Handford, Where's Wally? Walker Books Ltd. (2007)

# 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 resource saving

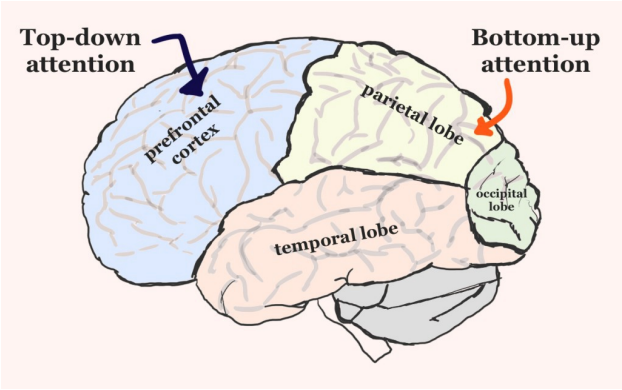


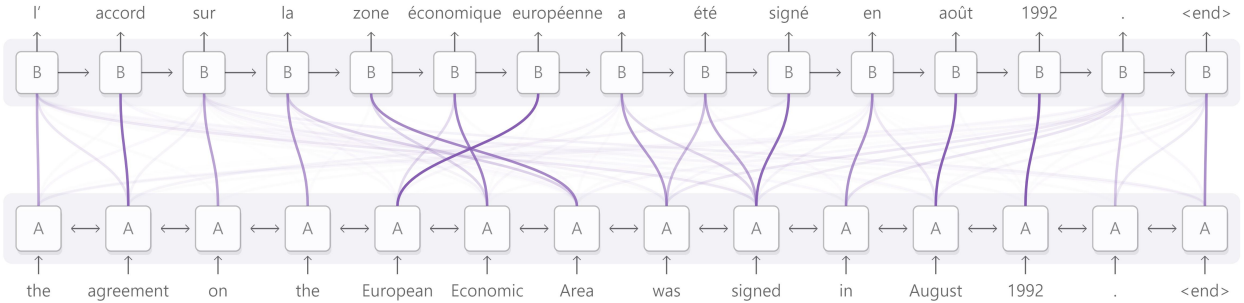

image source: MacKay, *The science of staying focused*, 2017

- in machine learning
  - an iterative process
  - applicable to input, memory, and representation
  - backprop trainable (soft attention)
  - ICML 2019 tutorial: ▸ Link


image source: Alex Smola, *Attention in Deep Learning* (2019), http://alex.smola.org/talks/ICML19-attention.pdf

# Attention in deep learning

- NLP (neural machine translation)




  - attention $\Rightarrow$ learned alignment (between source and target sentences)

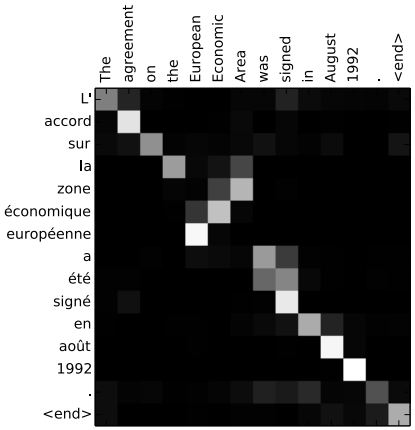



image sources: Olah and Carter, *Attention and Augmented Recurrent Neural Networks* (2016), https://distill.pub/2016/augmented-rnns/; Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014)

- 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.

- trend + key papers

  2014 recurrent models of visual attention[1]

  2014–15 attention in neural machine translation[2]

  2015–16 attention-based RNN/CNN in NLP

  2017 self-attention[3]

image source: 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

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

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

[3] 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

  $\Rightarrow$ may be overloaded $\Rightarrow$ suboptimal performance
- seq2seq with attention
  - encoder RNN: input encoding
  - attention: context vector computing

  $\Rightarrow$ offloads encoder RNN $\Rightarrow$ better performance

- seq2seq with attention ▸ source → acrobat of animation

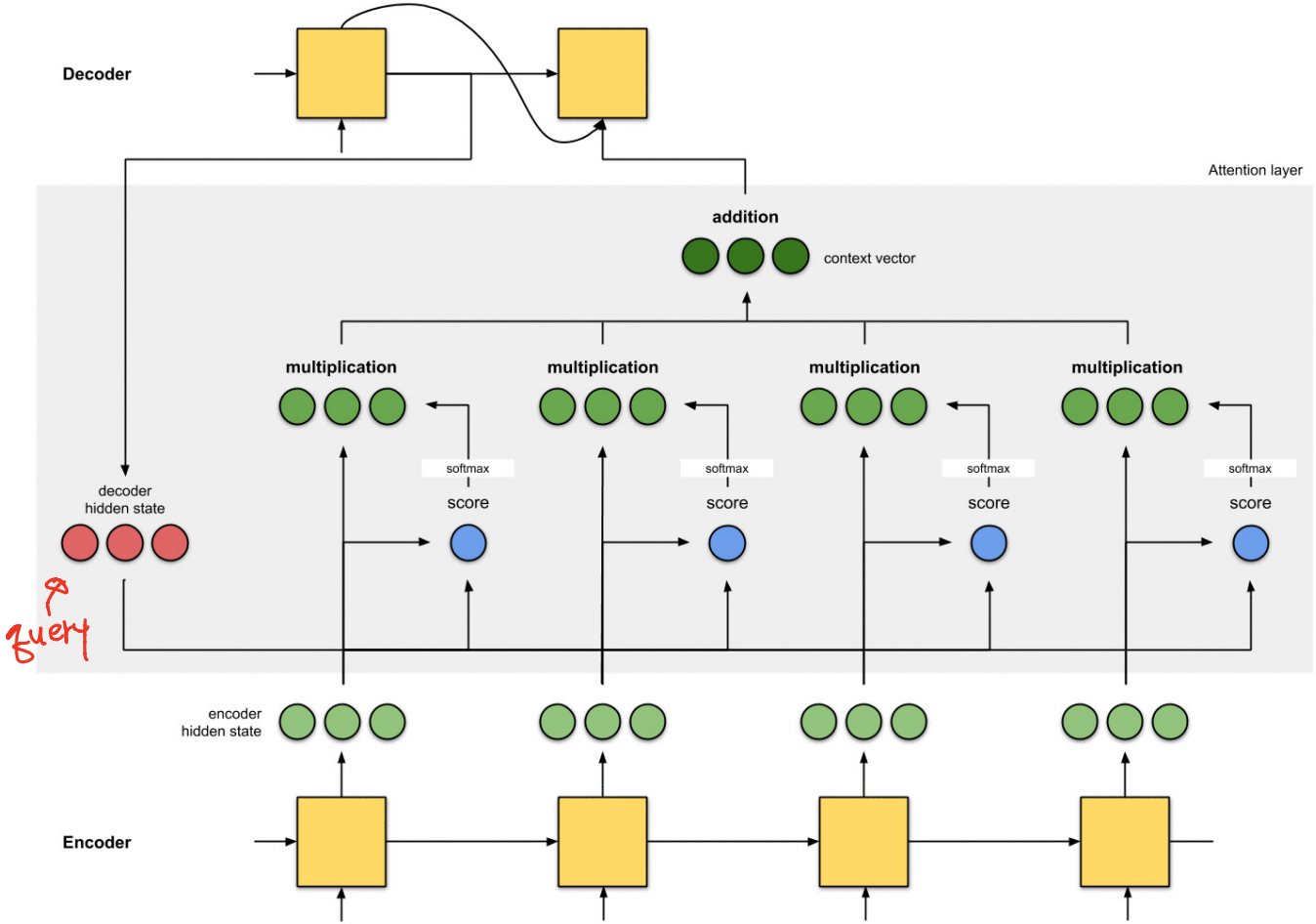



image source: Raimi Karim, *Attn: Illustrated Attention* (2019),
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

# Attention scoring

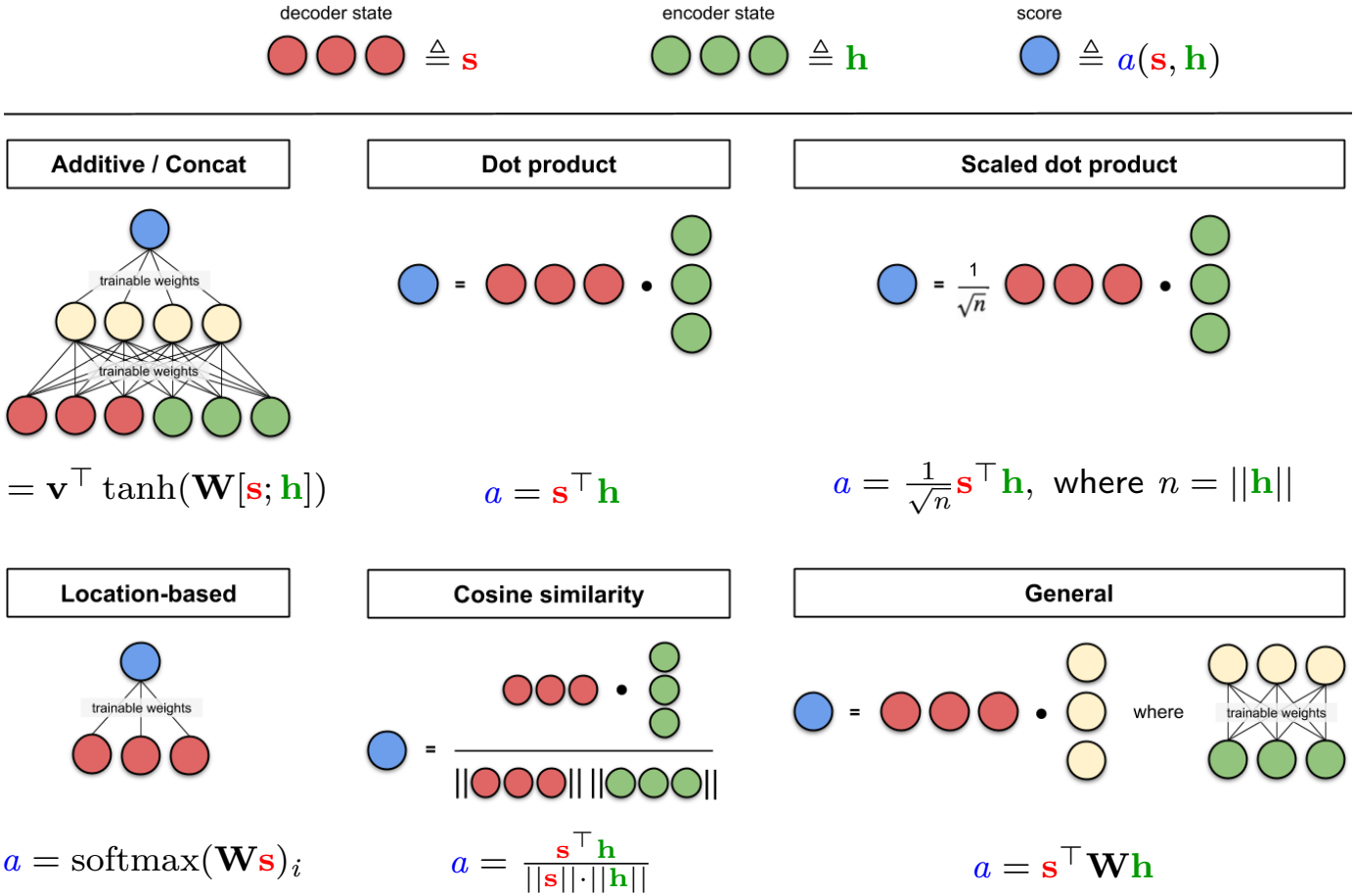


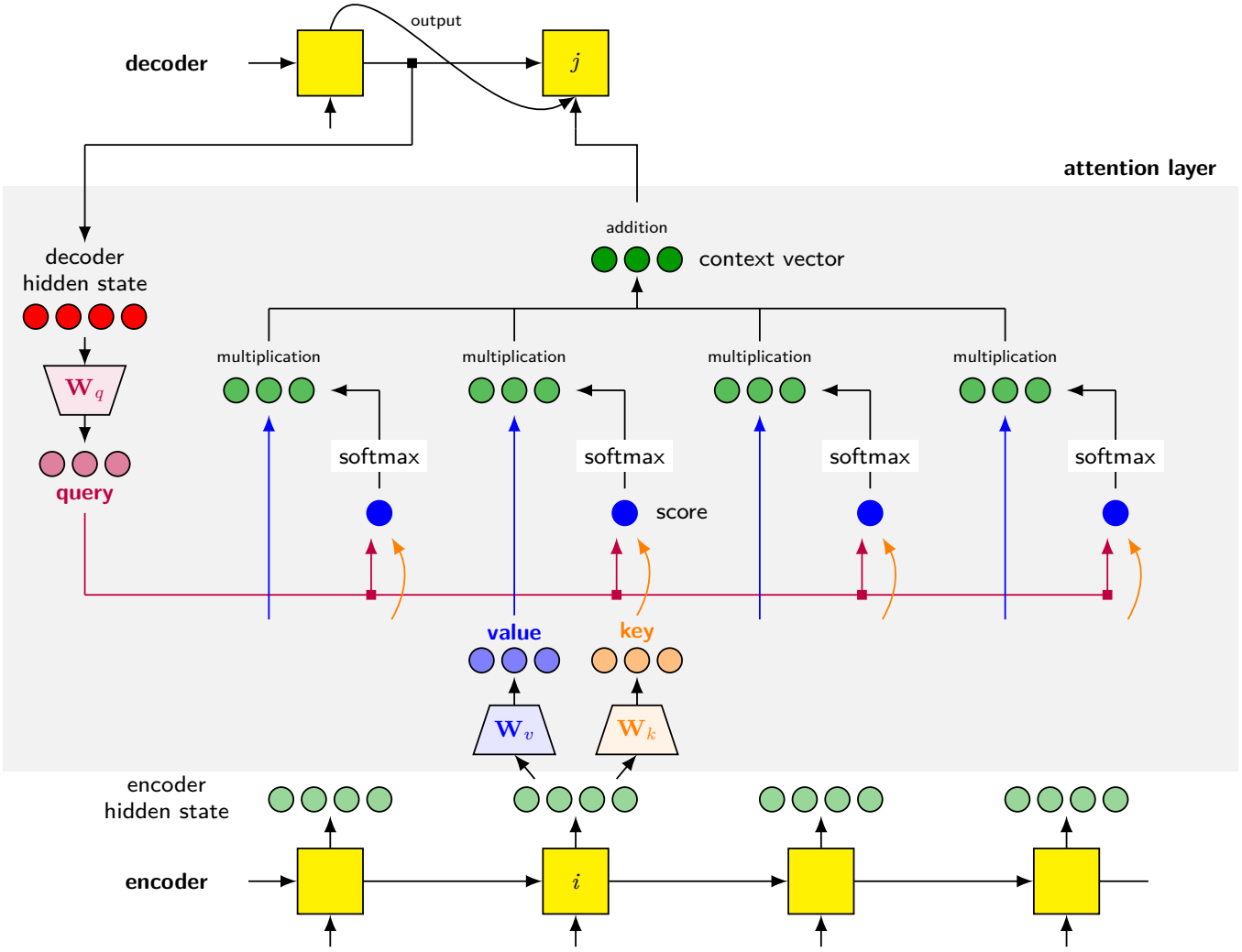

image source: Raimi Karim, *Attn: Illustrated Attention* (2019),
https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3

# 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 = weighted sum of values
    - each weight = function of (query, key)

- idea: separation of encoder hidden state into
  - attention information (key) and content information for context vector (value)


decoder
output
$j$
attention layer
addition
context vector
decoder
hidden state
$\mathbf{W}_q$
query
multiplication
multiplication
multiplication
multiplication
softmax
softmax
softmax
softmax
score
value
key
$\mathbf{W}_v$
$\mathbf{W}_k$
encoder
hidden state
encoder
$i$

# Taxonomy of attention mechanisms

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

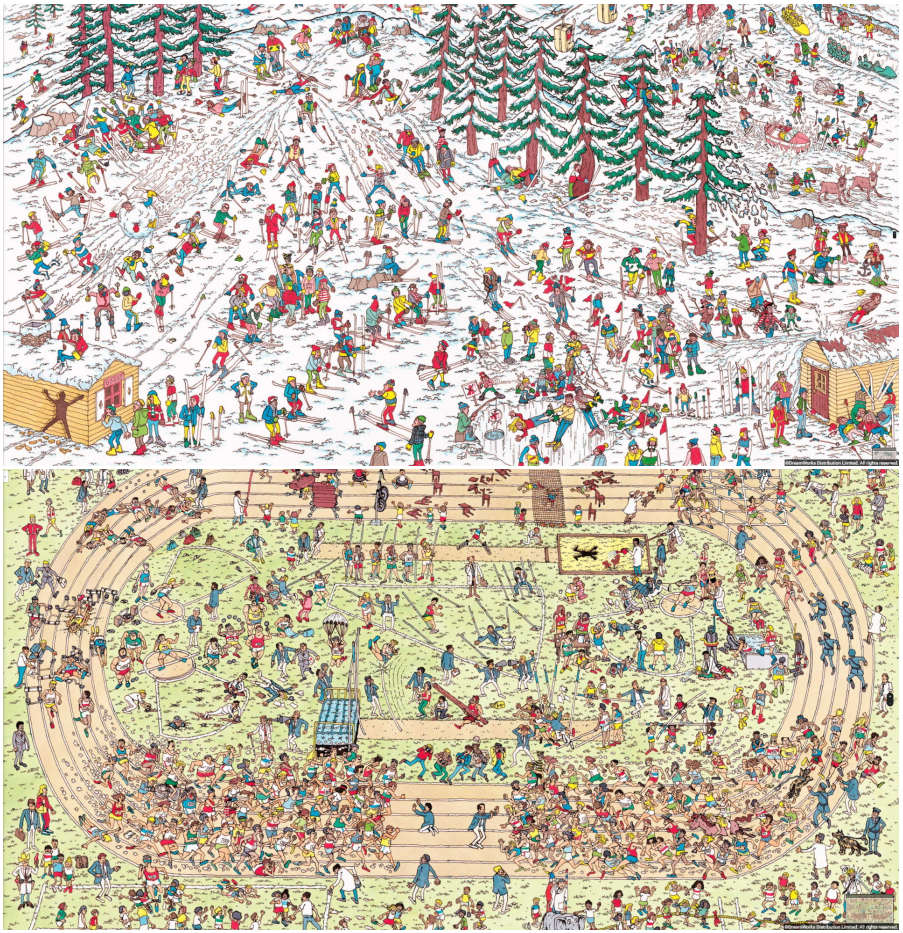
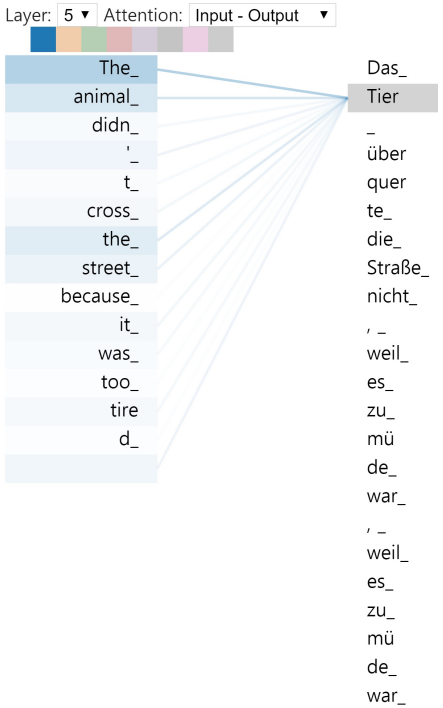


image source: Martin Hartford, *Where's Wally?* Walker Books Ltd (2007)

# Self-attention

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



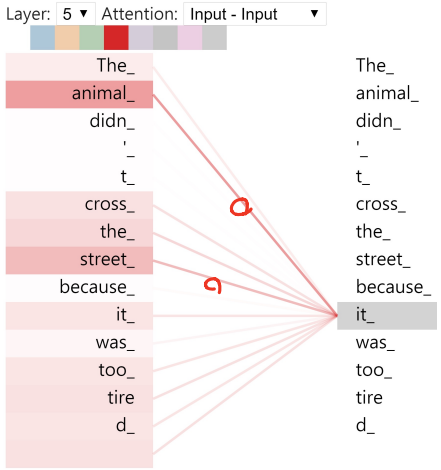
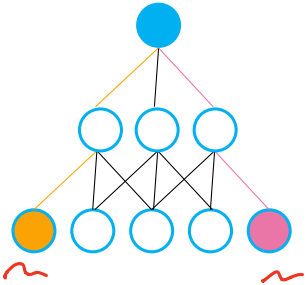
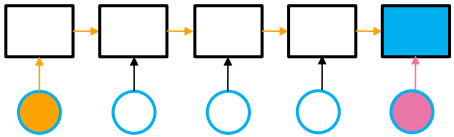




image source: Tensor2Tensor, https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

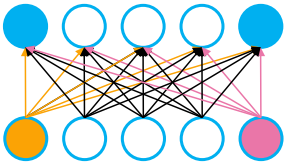
- comparison
  - CNN vs RNN vs self-attention



CNN



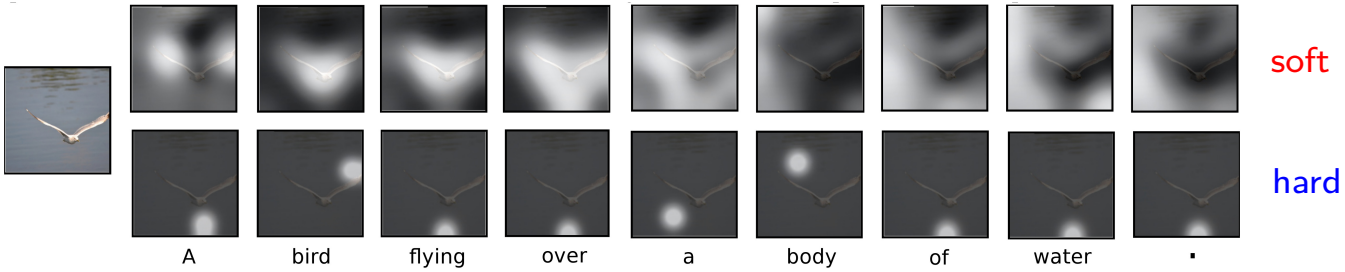
RNN



Self-attn.

# Soft vs hard attention

example: image caption generation




- 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: Stochastic
  - only (randomly) selects patch(es) of the image to attend to at a time
  - context $=$ {feature(s) randomly sampled wrt attention map} $\Rightarrow$ "subset"

image source: 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

comparison:

- soft attention
  - model is smooth and differentiable
  - slow (heavy computation) when source input is large
- hard attention
  - fast inference (less computation)
  - model is non-differentiable (stochastic)
  - $\Rightarrow$ requires complicated techniques (*e.g.* variance reduction, RL) to train

# Global vs local attention

first proposed in machine translation (source sentence $\rightarrow$ target sentence)

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

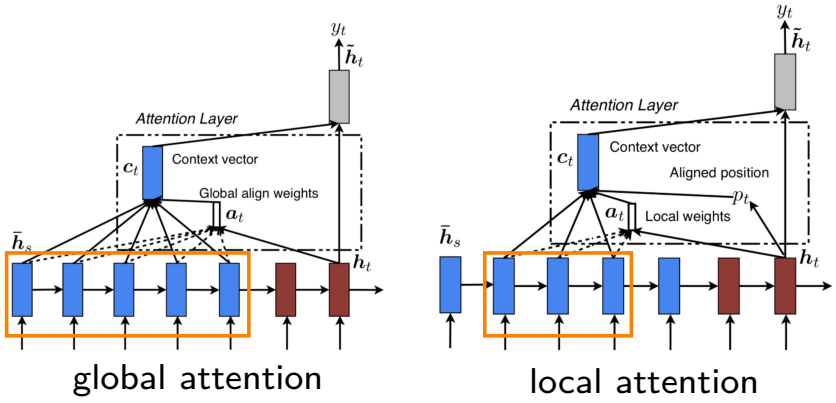



image source: Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation". In: *arXiv preprint arXiv:1508.04025* (2015)

# Outline

# Transformer

image sources: https://images.app.goo.gl/7RonggzhGp3Y9f3B9; https://images.app.goo.gl/AW4jp4nymvrK8Tm2A

# A high-level look

- a sequence-to-sequence model
  - input sequence (arbitrary length) $\longrightarrow$ output sequence (arbitrary length)


image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

- encoder-decoder architecture
  - an encoding component + a decoding component
  - and connections between them (cross-attention)

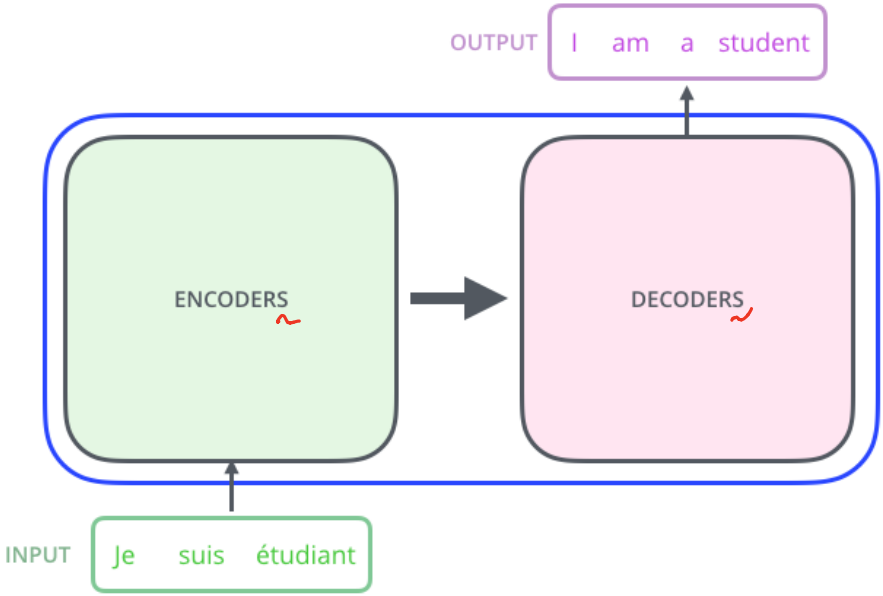



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

- encoding component: a stack of encoders[4]
- decoding component: a stack of decoders of the same number

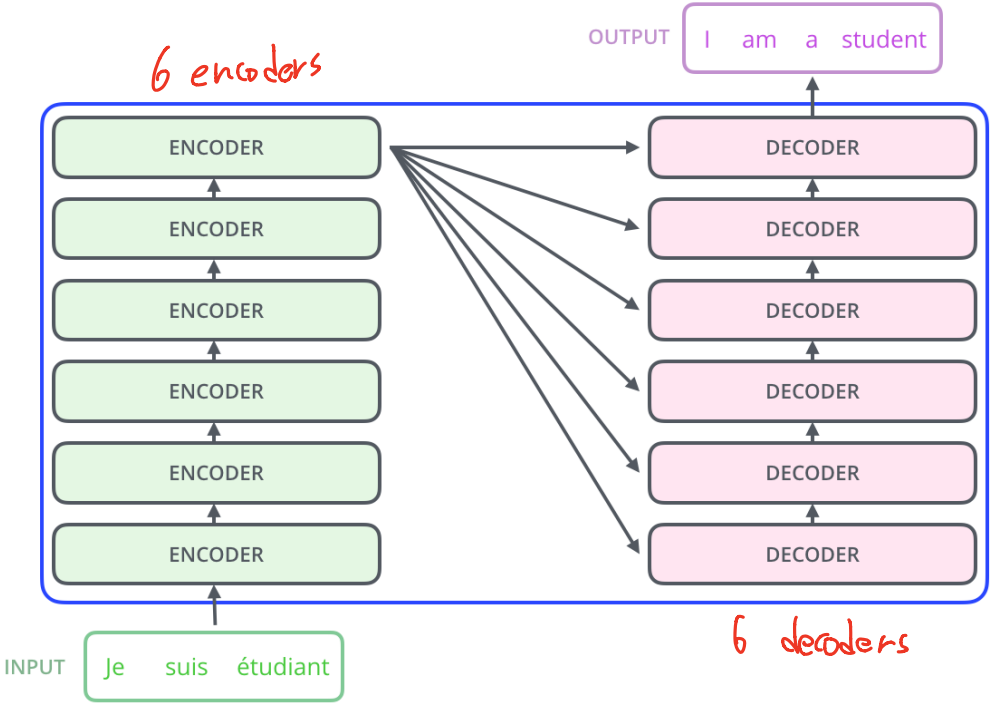
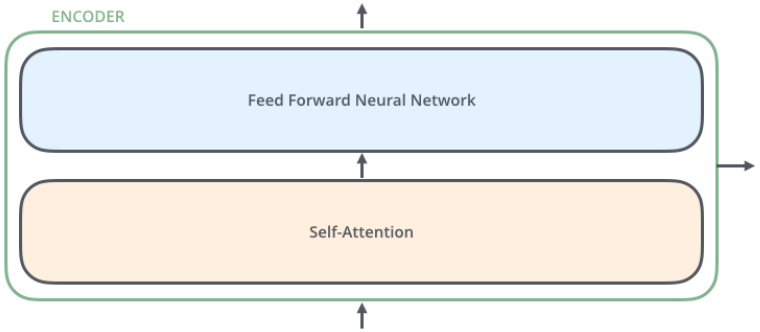



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

[4] the 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:

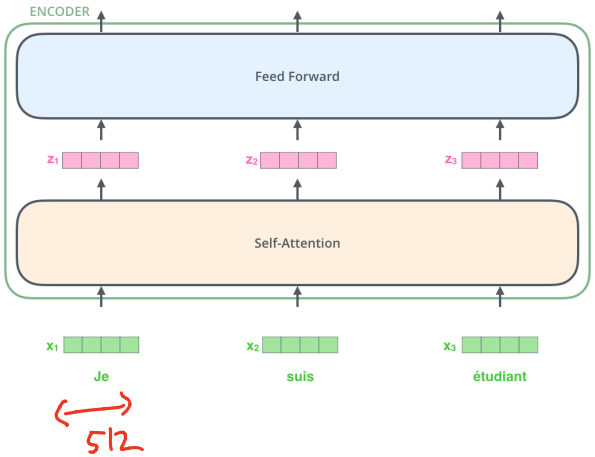



- self-attention layer
  - helps the encoder look at other words in the input sentence as it encodes a specific word

- feed-forward neural net layer
  - the exact same feed-forward net is independently applied to each position

image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

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



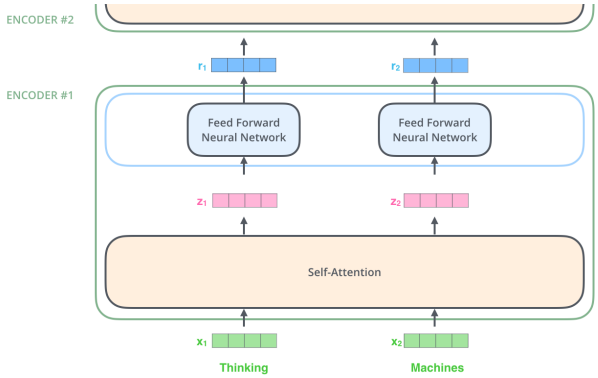
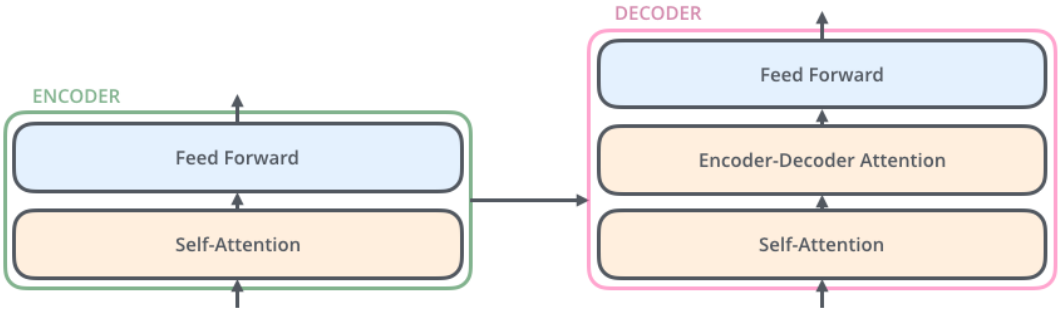




image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# Decoder

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



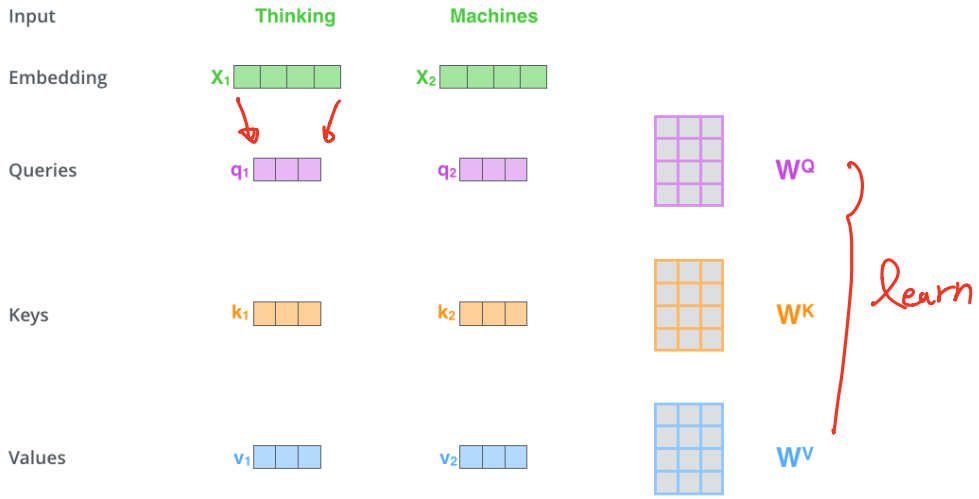

- encoder-decoder attention layer
  - helps the decoder focus on relevant parts of the input sentence (similar to what attention does in seq2seq models)

image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# Outline

# 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




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

image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# 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) · (key vector of the word we're scoring)

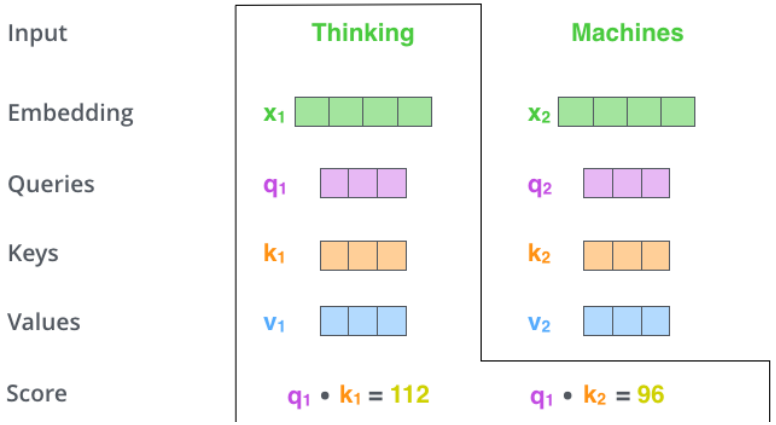



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# 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 probabilities (which sum to 1)

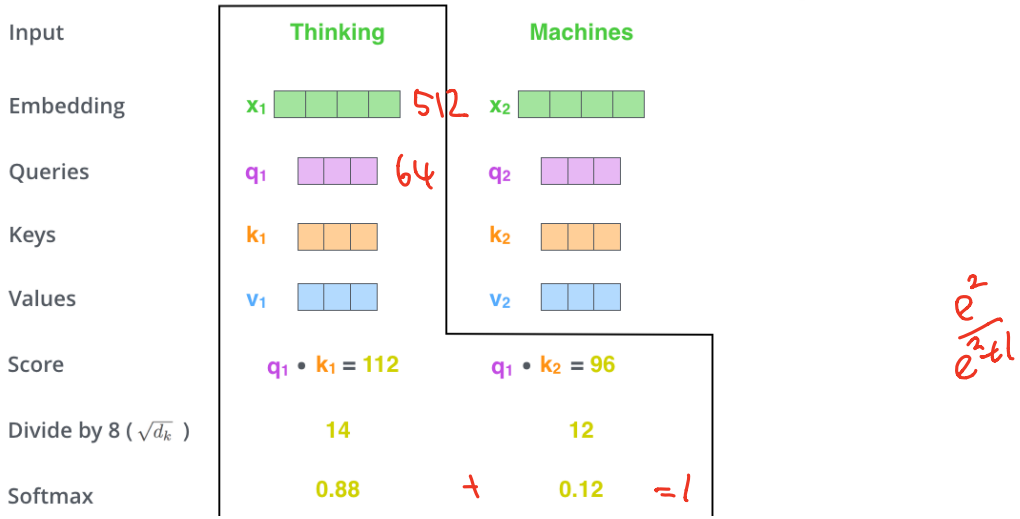



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# Step 4

- calculate the output of self-attention layer

↑

weighted sum of value vectors

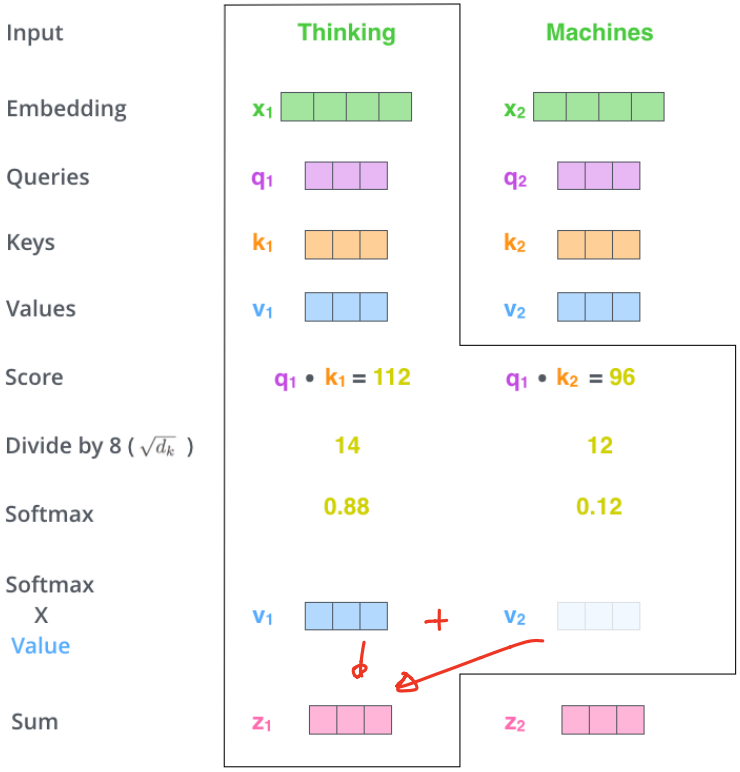
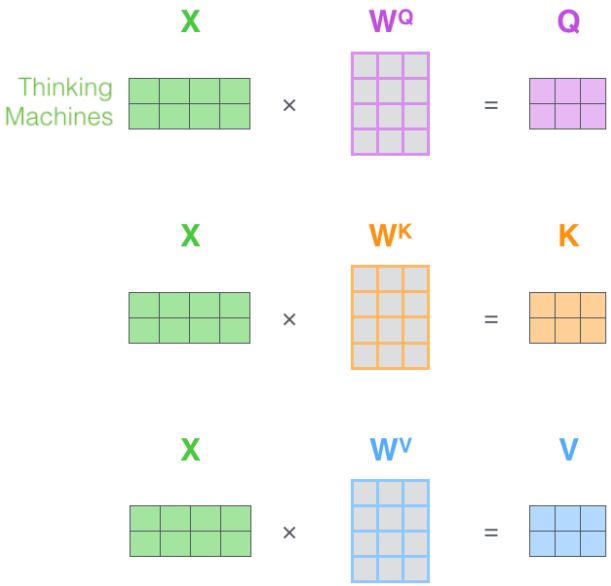



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# Matrix calculation of self-attention

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



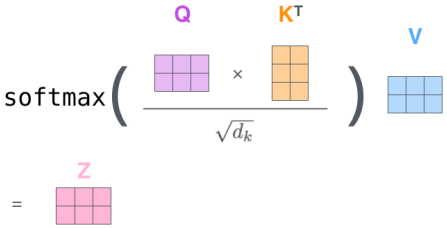




image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# Outline

# Multi-head attention

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

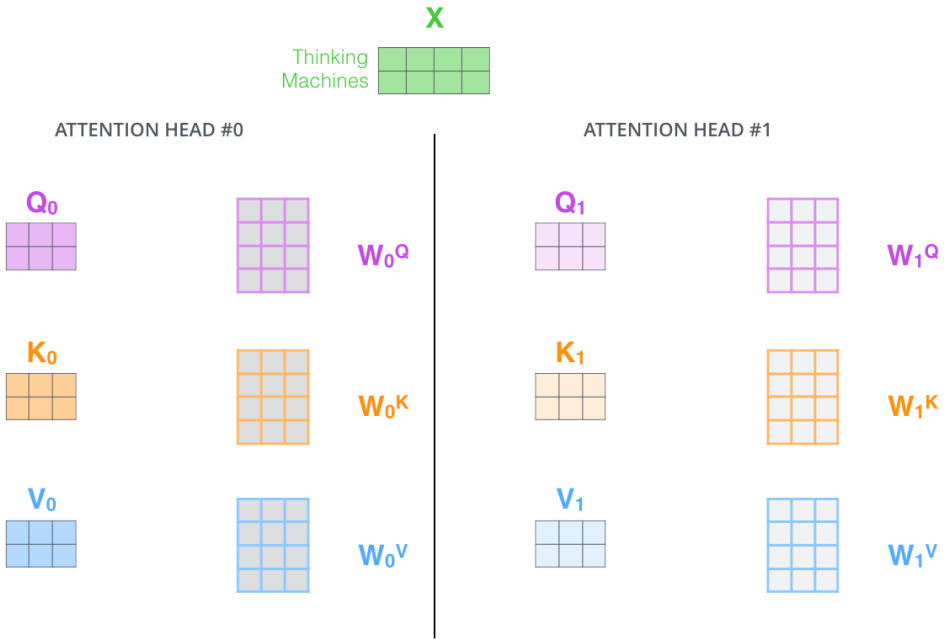



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

- different attention heads
  - focus differently as we encode the word "it"

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

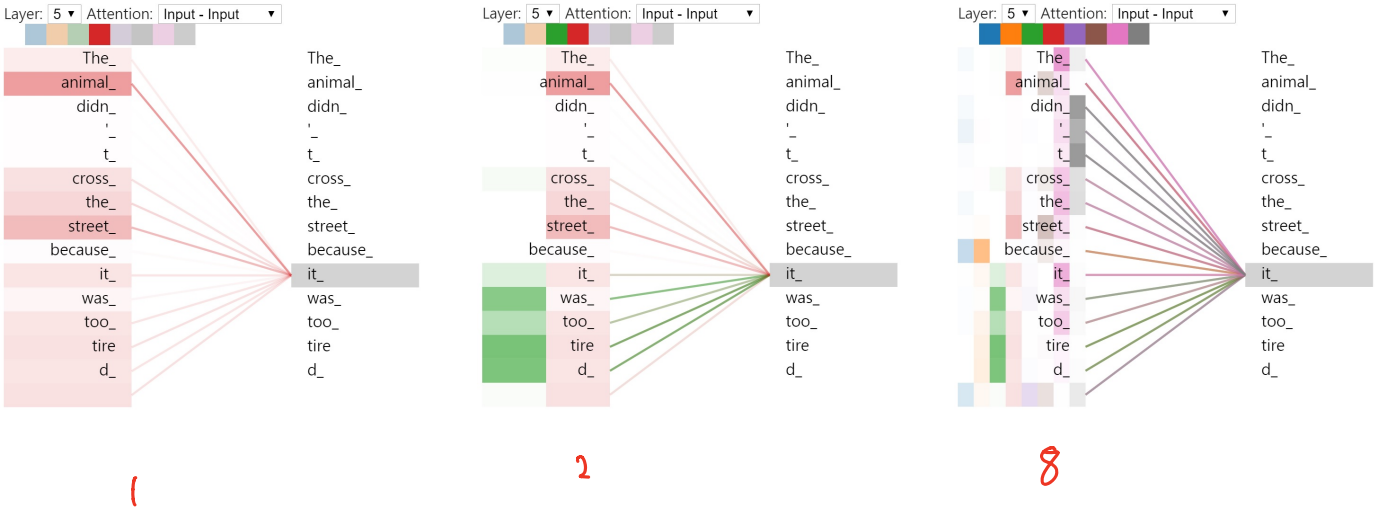



image source: Tensor2Tensor, https://colab.research.google.com/github/tensorflow/tensor2tensor/blob/master/tensor2tensor/notebooks/hello_t2t.ipynb

- the original transformer paper
  - 8 attention heads

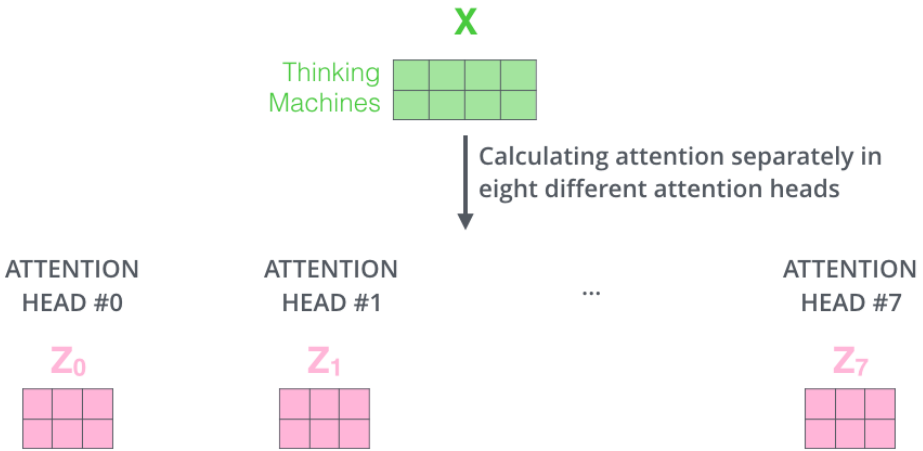



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# Condensing multi-head attention results

- concatenate heads and multiply with a learnable weight matrix

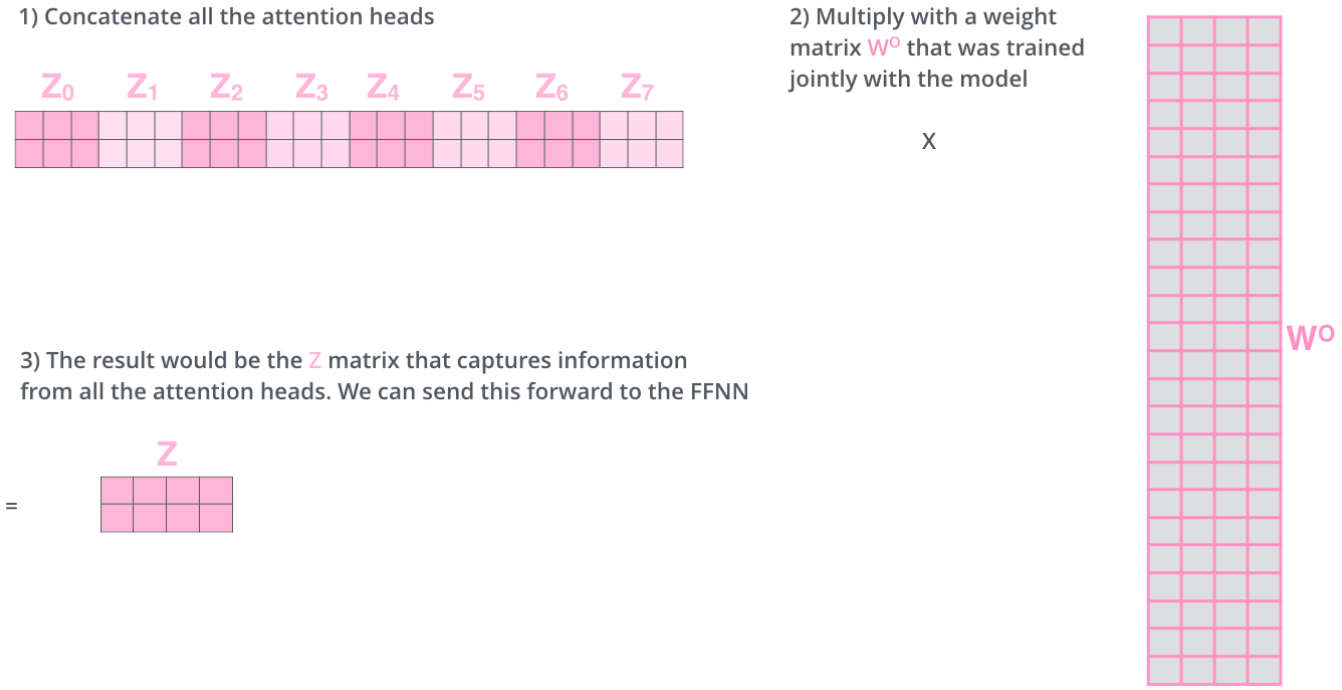



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# Recap

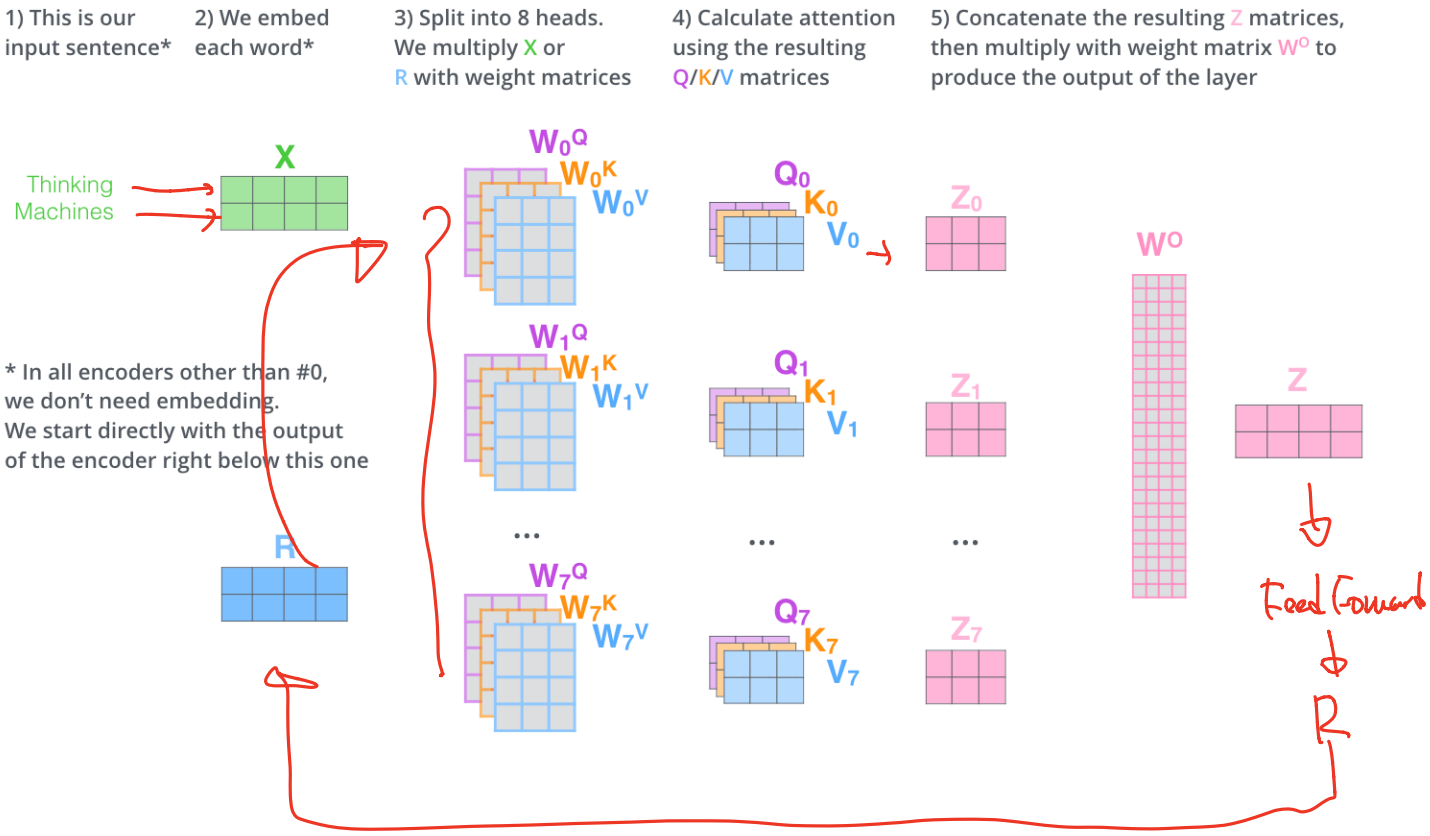
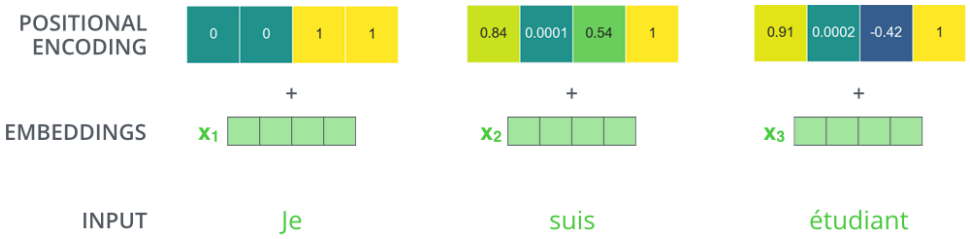



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# Outline

# Representing the order of words

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




- 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

image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# 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

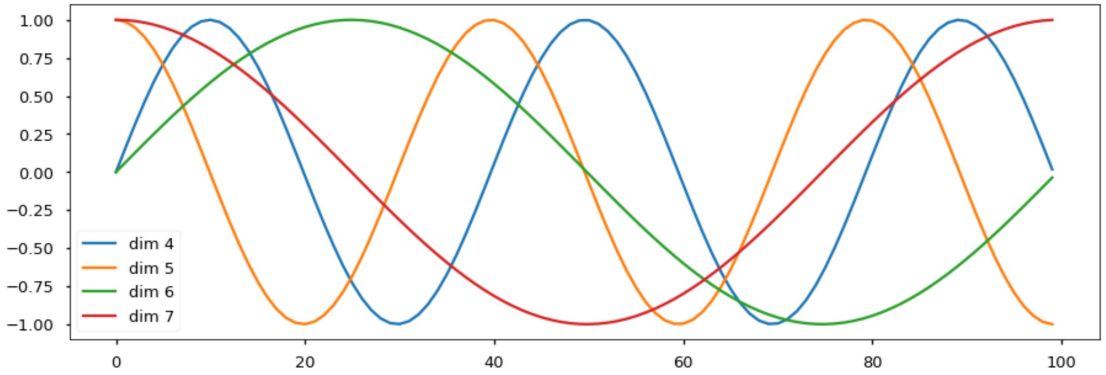
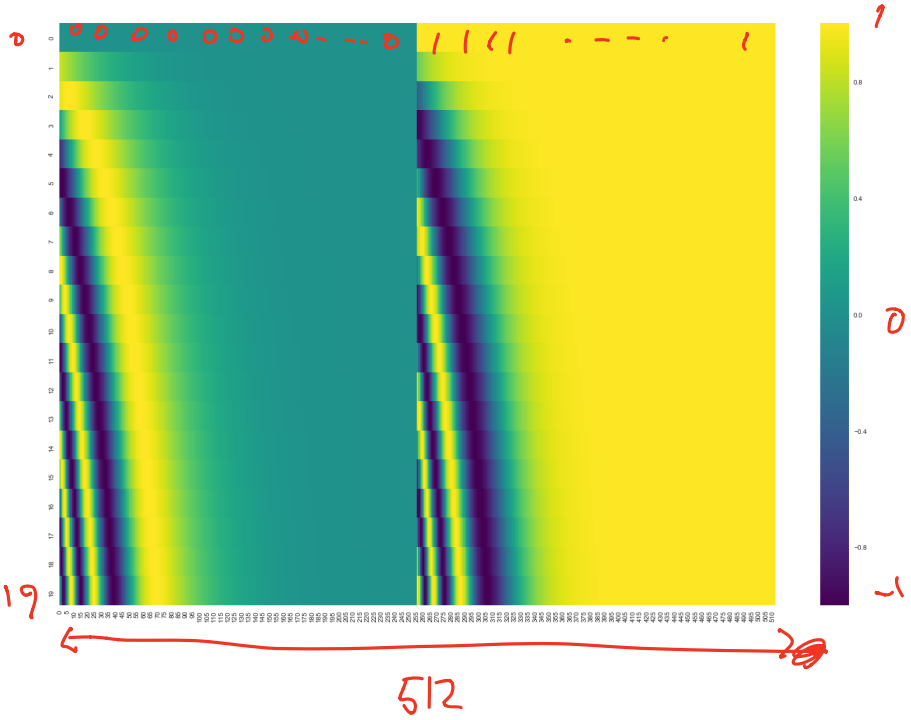


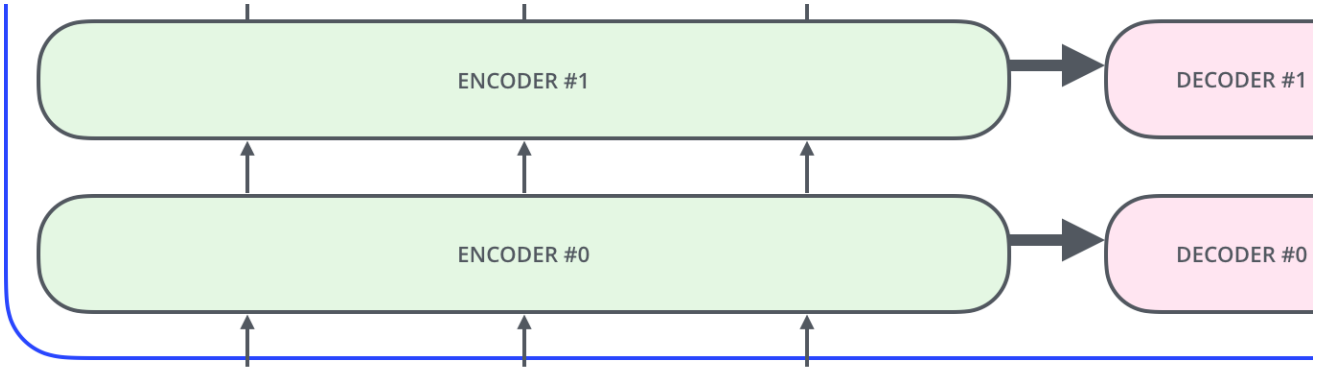

image source: *The Annotated Transformer*, https://nlp.seas.harvard.edu/2018/04/03/attention.html

- positional encoding (PE) in the original transformer
  - 512 dimensions: 256 (sine) + 256 (cosine)



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

image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer



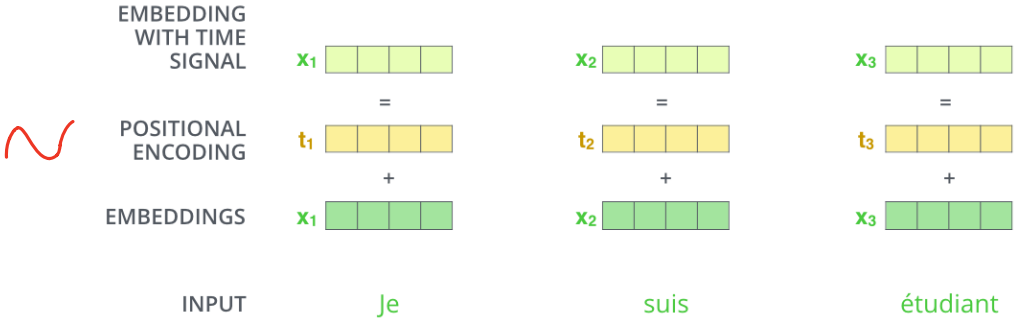
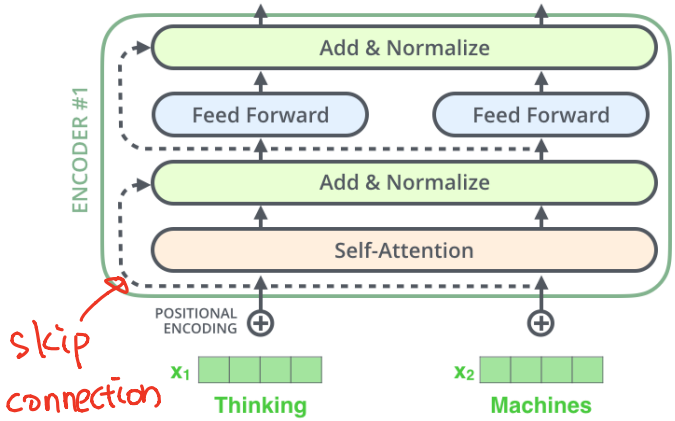




image source: Jay Alammar, *The Illustrated Transformer*, `http://jalammar.github.io/illustrated-transformer`

# Outline

# Add and normalize

- each sublayer (self-attention and feed-forward) in encoder
  (1) has a residual connection around it and
  (2) is followed by layer normalization[5]



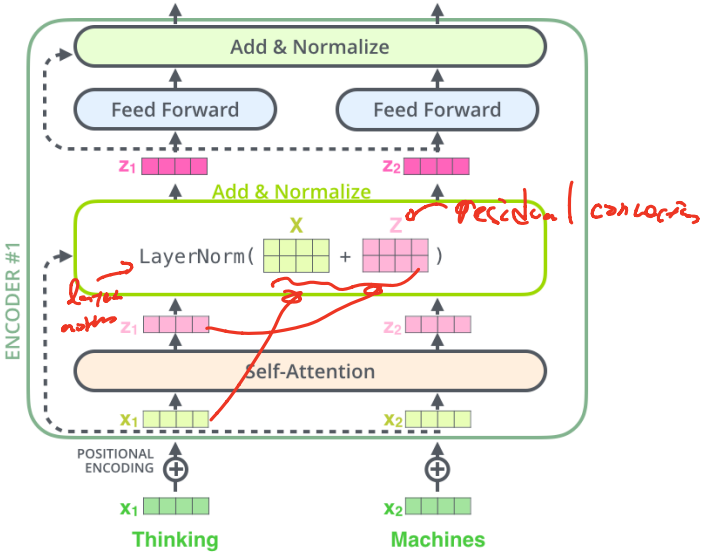




image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

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

# Outline

# Decoder architecture

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

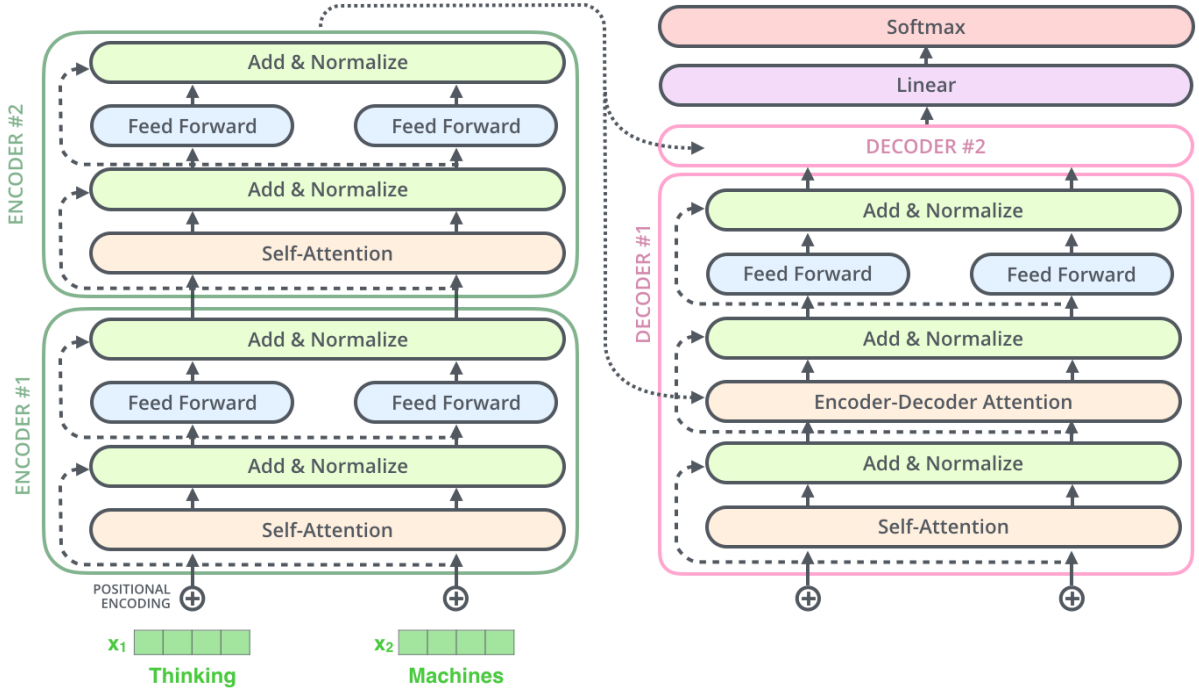
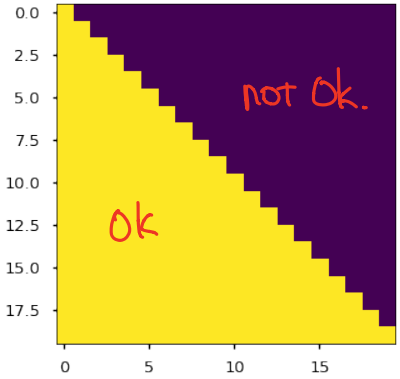



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

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

$$P(y_1, y_2, \ldots, y_T \mid \boldsymbol{x}) = \prod_{t=1}^{T} p(y_t \mid y_1, y_2, \ldots, y_{t-1}, \boldsymbol{x})$$

  - 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

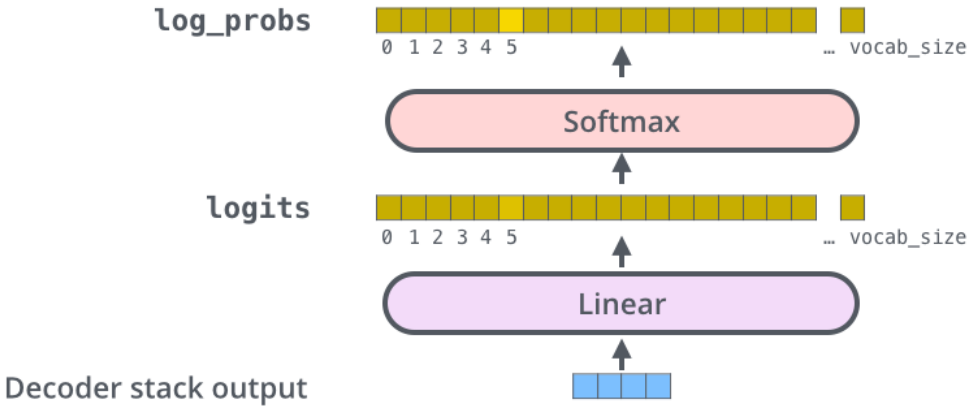
image source: *The Annotated Transformer*, https://nlp.seas.harvard.edu/2018/04/03/attention.html

# Logits and softmax layers

- linear layer: float vector $\longrightarrow$ logits[6]

  decoder output — length: vocab size (very large)

- softmax layer: logits $\longrightarrow$ probability vector




⁂ training loss: cross-entropy loss

image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

[6] a 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: greedy decoding
  - output the word with the highest probability

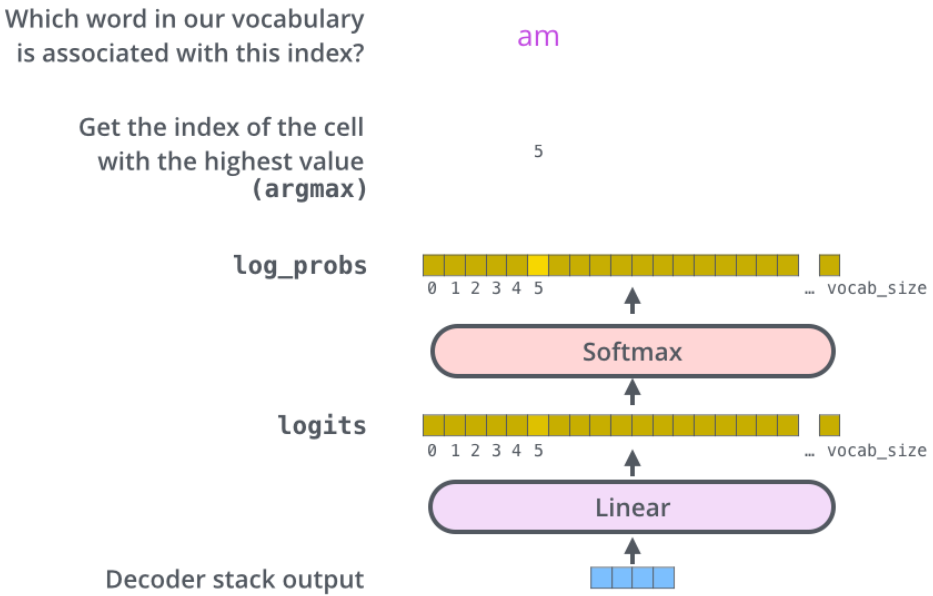


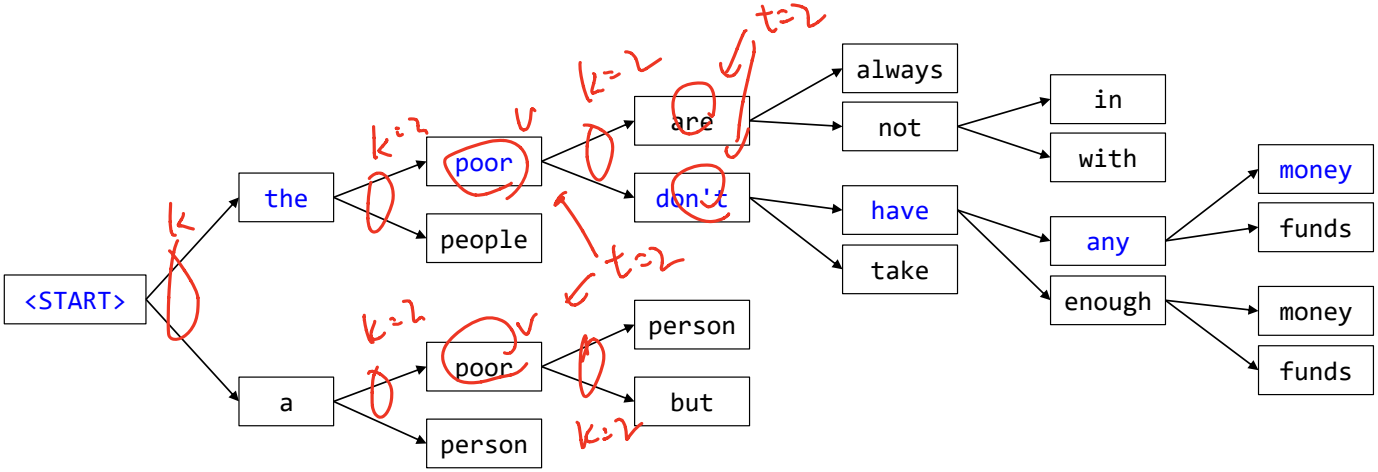

image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

- method #2: beam search (hyperparameters: beam size $k$, top beams $t$)[7]
  - 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`




[7] often set $k = t$; $k = t = 1$ corresponds to greedy decoding

# Altogether

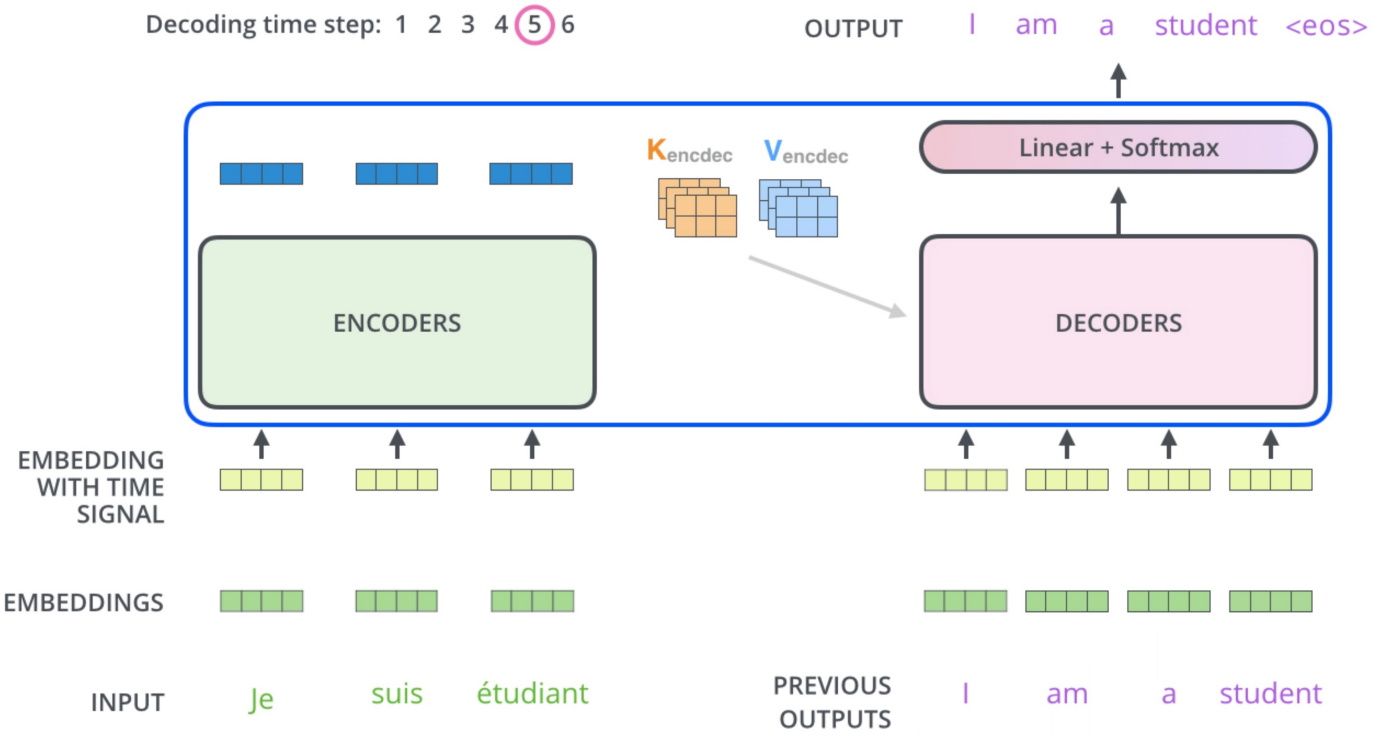



image source: Jay Alammar, *The Illustrated Transformer*, http://jalammar.github.io/illustrated-transformer

# The transformer

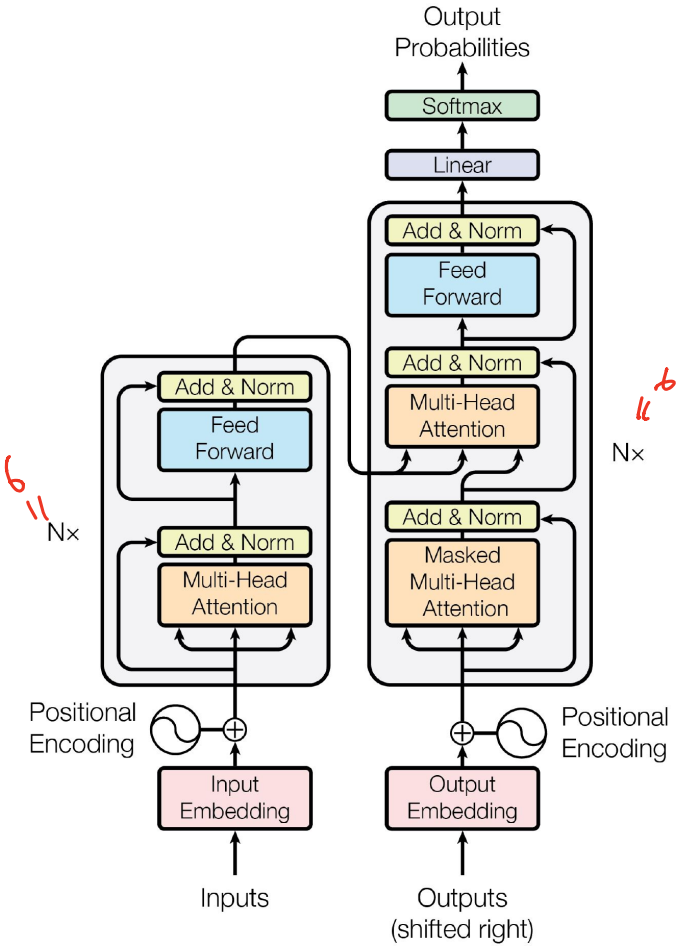



image source: Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008

# Outline

# BERT (Bidirectional Encoder Representations from Transformers[8])

- a language representation model
  - can pre-train deep bidirectional 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

AE : BERT
AR : GPT

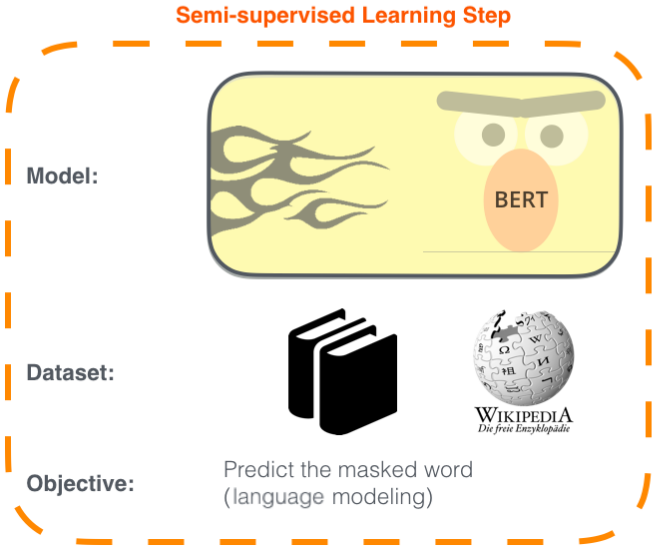

image source: https://images.app.goo.gl/ea55tHH34fRA9RT39

[8] 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.

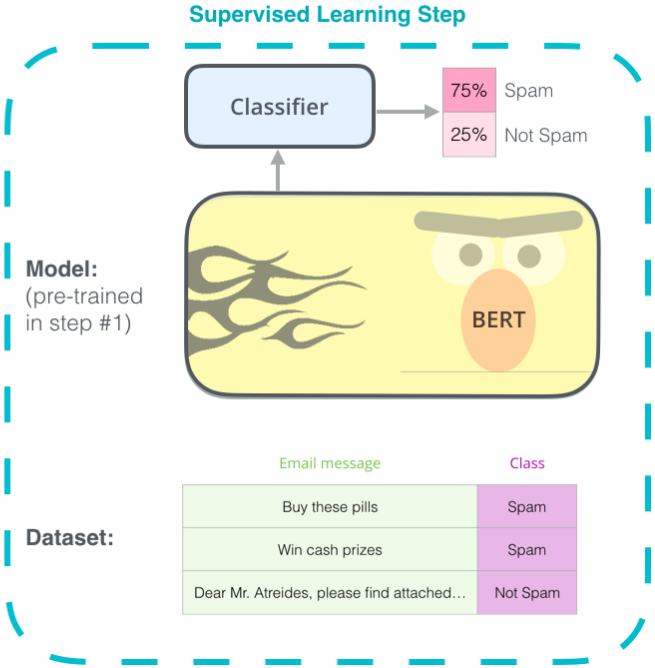
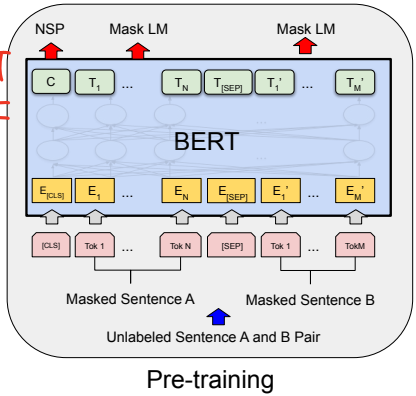



image source: Kaushal Trivedi, *Introducing FastBert — A simple Deep Learning library for BERT Models* (2019), https://medium.com/huggingface/introducing-fastbert-a-simple-deep-learning-library-for-bert-models-89ff763ad384

# Defining a prediction goal

- a major challenge in training language models
- many models
  - predict the next word in a sequence

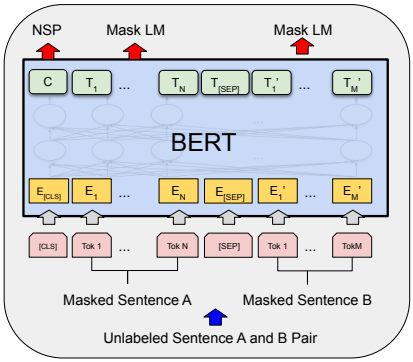
  *e.g.* `The child came home from ___`
  - a directional approach $\Rightarrow$ inherently limits context learning
- BERT uses two "pre-training" strategies to overcome this challenge
  1. Masked language model (MLM)
  2. next sentence prediction (NSP)
     - MLM and NSP are trained together, minimizing a combined loss function



Pre-training

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

# Masked LM (MLM)



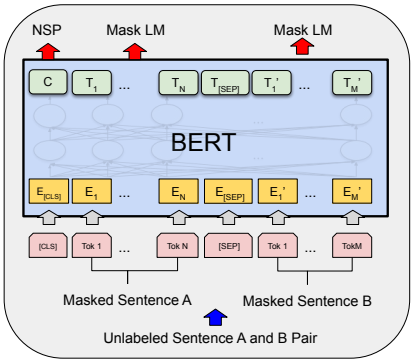
Pre-training

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, non-masked words

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

# Next sentence prediction (NSP)



Pre-training

during training:

- the model receives two sentences as input and learns to predict
  - if the second sentence is the subsequent 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

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

- input pre-processing[9] for NSP
  1. [CLS] token: inserted at the beginning of 1st sentence (classification)
  2. [SEP] token: inserted at the end of each sentence (seperation)
  3. Sentence embedding (to indicate each sentence) is added to each token
  4. positional embedding is added to each token

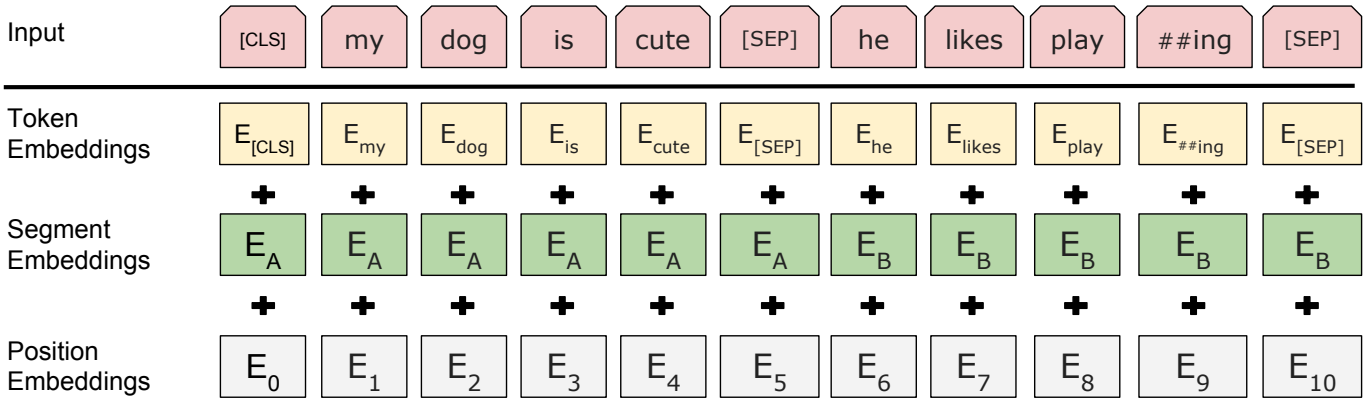



image source: Jacob Devlin et al. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018)

[9] [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)

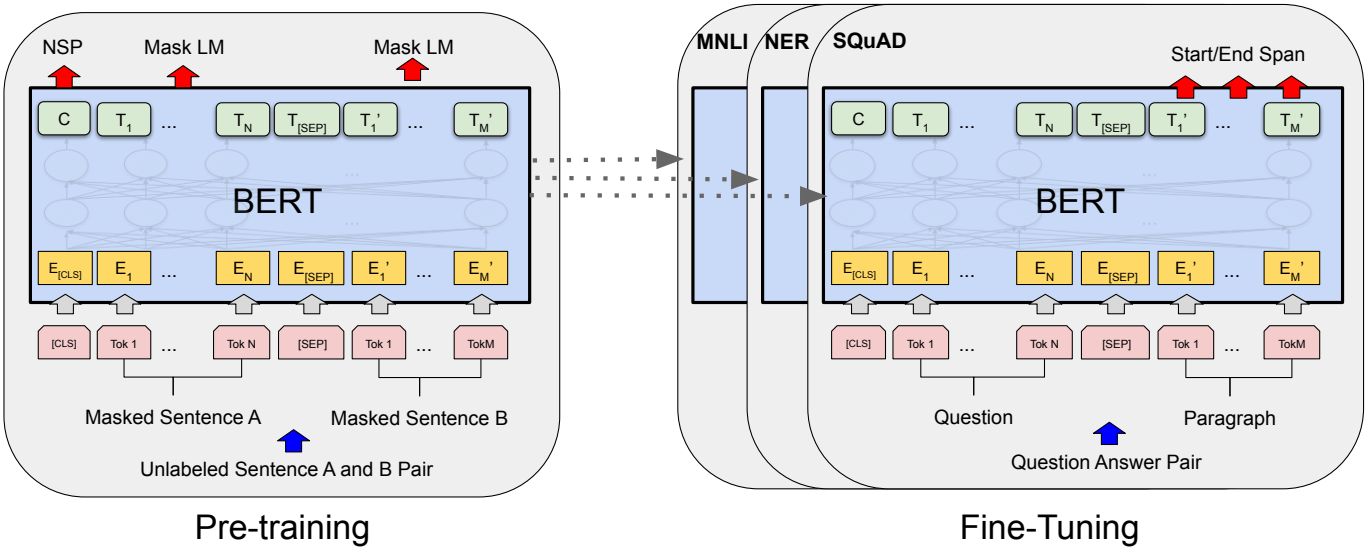
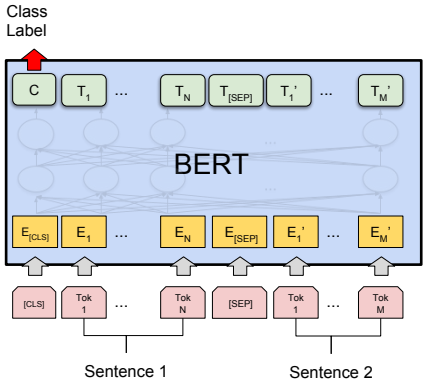
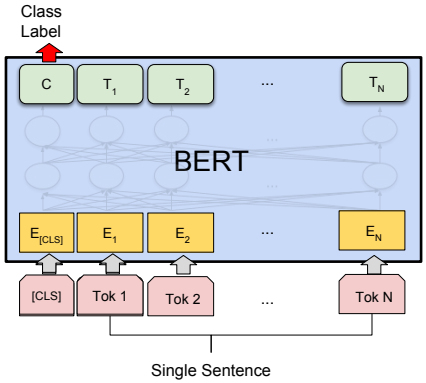



image source: Jacob Devlin et al. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018)

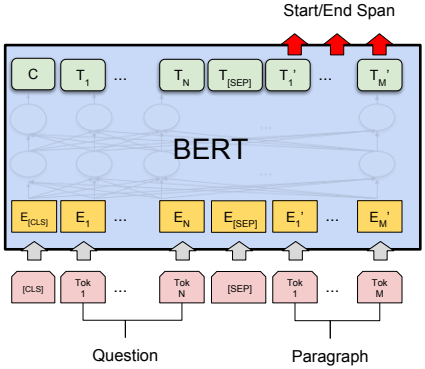
- fine-tuning BERT on different tasks



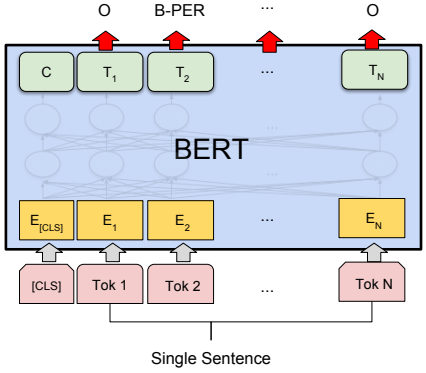

(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG




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




(c) Question Answering Tasks: SQuAD v1.1




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

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

# OpenAI GPT (Generative Pre-trained Transformer[10])

- similar idea
  - unsupervised pre-training + transformer
- key difference
  - BERT: bidirectional transformer
  - GPT: left-to-right transformer

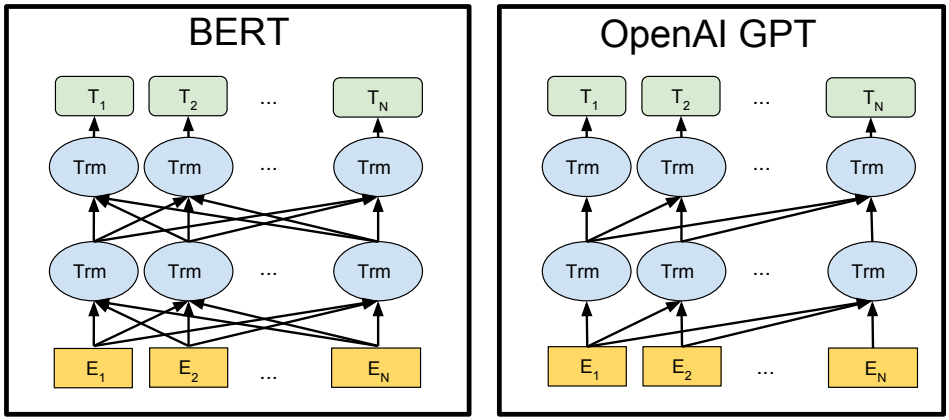



image source: Jacob Devlin et al. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018)

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

# OpenAI GPT-2[11]

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

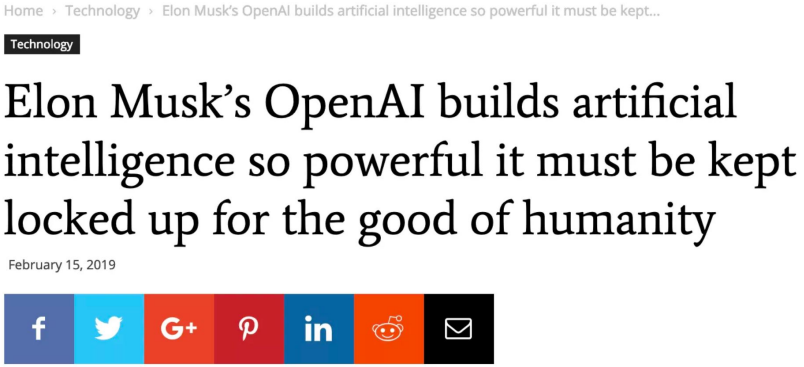




image sources: OpenAI, *Better Language Models and Their Implications* (2019), https://openai.com/blog/better-language-models/#sample8; BusinessFast (2019), https://www.businessfast.co.uk/

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

# XLNet[12]

- BERT limitations
  - neglects dependency between masked positions
  - suffers from a pretrain-finetune discrepancy

This is good
This good is
⋮

- XLNet: a generalized autoregressive pretraining method
  - based on transformer-XL (autoregressive model: RNN + transformer)
  - learns bidirectional contexts
    - by considering all permutations of the factorization order

$$AR: \log p_\theta(x) = \sum \log p_\theta(x_t | x_{<t})$$

$$XLNet: \mathbb{E}_z \left[ \log p_\theta(x_{z_t} | x_{z<t}) \right]$$

image source: Keita Kurita (2019), https://mlexplained.com/2019/06/30/paper-dissected-xlnet-generalized-autoregressive-pretraining-for-language-understanding-explained/

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

comparison:

- traditional LM

- BERT[13]

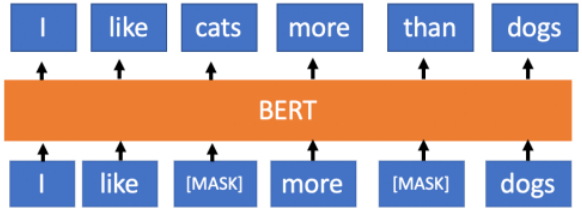
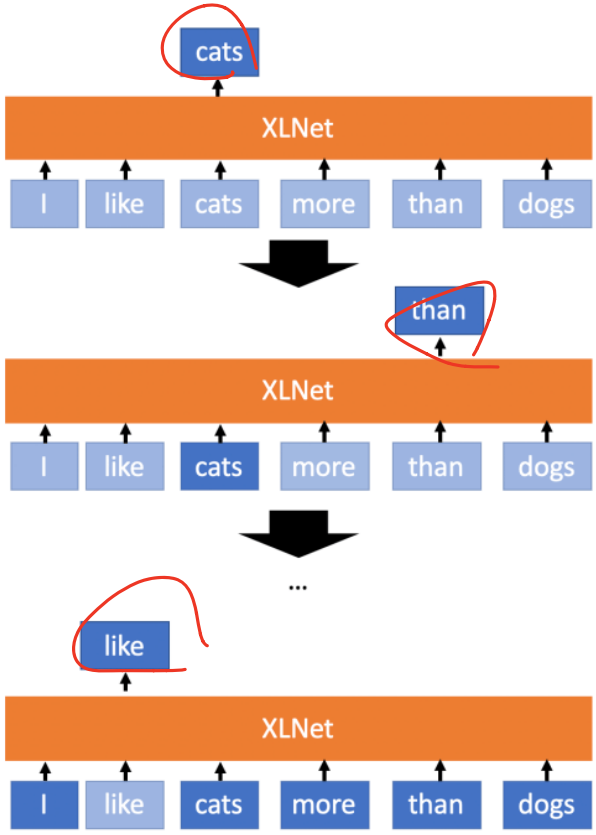



image source: Keita Kurita (2019)

- XLNet[14]




[13] predicts all masked words simultaneously

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

# OpenAI GPT-3[15]

- scale-up version of GPT-2
  - 175 billion parameters (10x more than any previous non-sparse LM)
  - trained using 499 billion words and 350 GBytes of GPU memory
  - tested in few-shot setting (without any gradient updates or fine-tuning)
  - can generate human-like samples of news articles


*A - 간단한 질문 하나 할게요. 누가 이 지구를 만들었을까요?*
*B - '시간'인가요?*
*A - 아닙니다.*
*B - 그럼 누구죠?*
*A - '외계생물체'?*
*B - 아니요.*
*A - '신'?*
*B - 신은 누구인가요?*
*A - 이 지구를 만든 지성이지요.*
*B - 신은 존재하나요?*
*A - 그럼요.*
*B - 신을 본 적 있습니까?*
*A - 없어요*
*B - 신이 존재한다는 걸 어떻게 확신하죠?*
*A - 그게 최선이니까요. 여전히 확신합니다.*

image source: https://news.kbs.co.kr/news/view.do?ncd=4504471

[15] Brown, Tom B., et al. "Language models are few-shot learners." *arXiv preprint arXiv:2005.14165* (2020), https://arxiv.org/pdf/2005.14165.pdf, repository at https://github.com/openai/gpt-3

# Outline

# 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-2/3
  - BERT: pre-training (MLM + NSP) $\rightarrow$ fine-tuning
  - XLNet: based on transformer-XL (RNN + transformer)