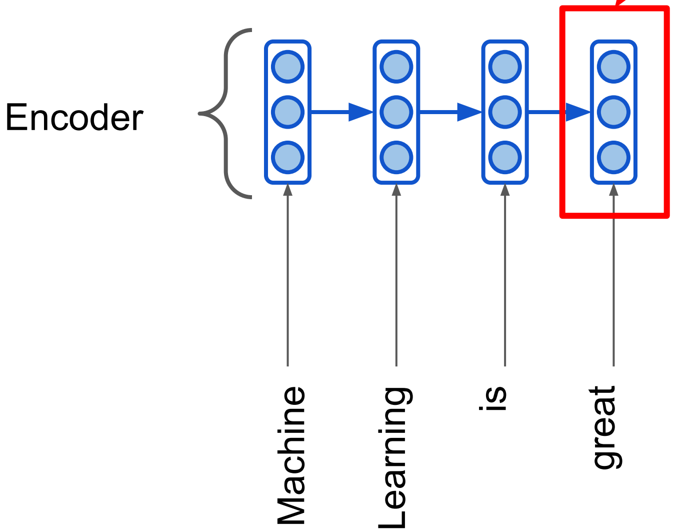
Lecture 8
Attention, self-attention
Transformer
BERT

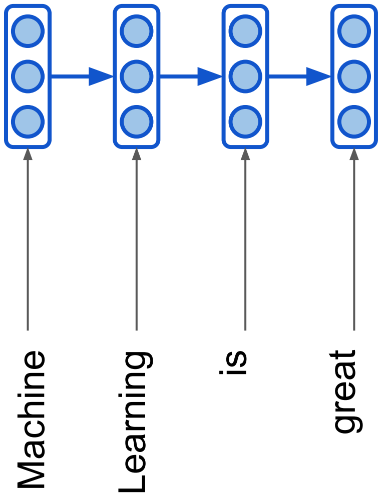
Vladislav Goncharenko

Moscow, 2021

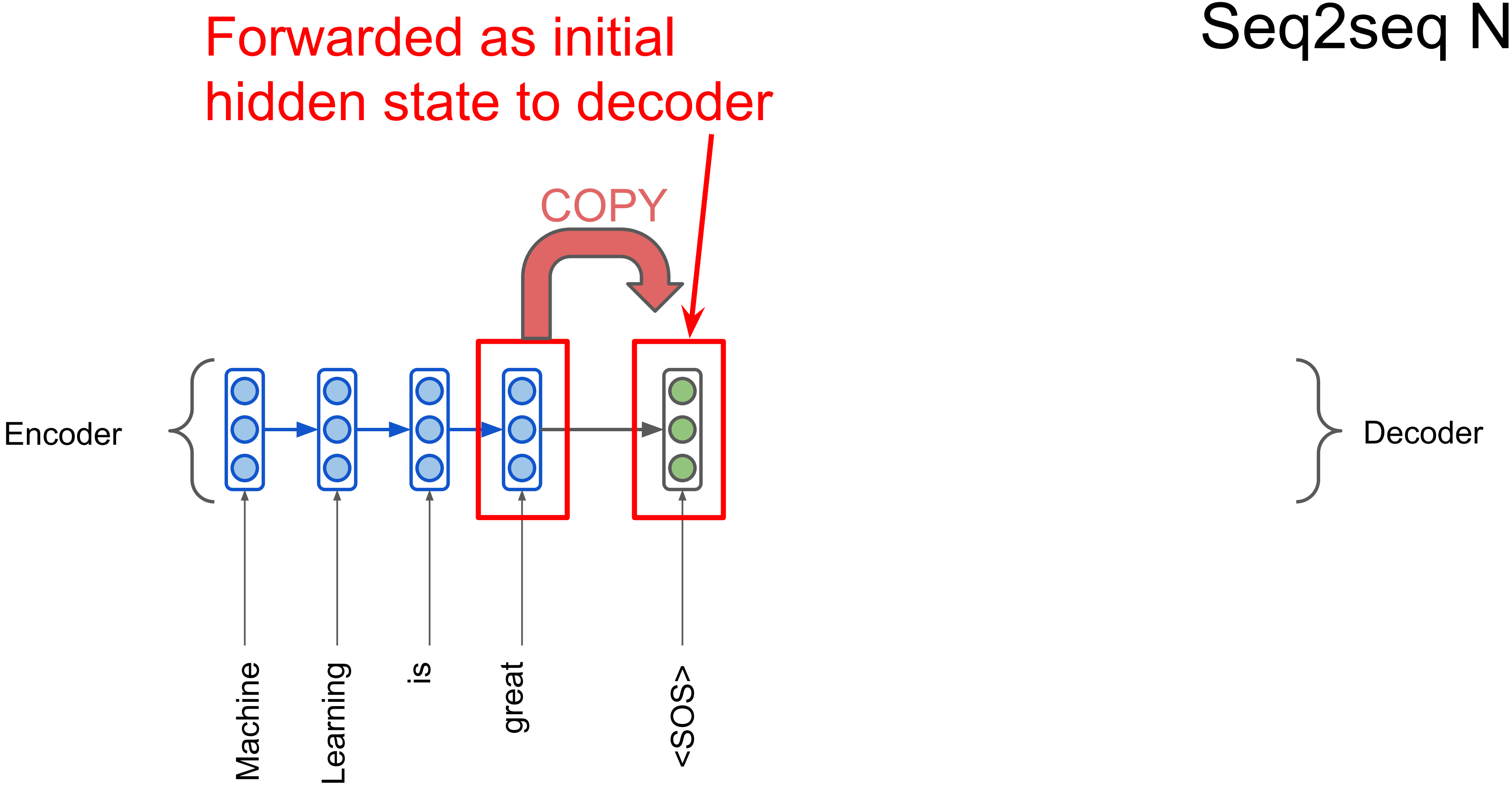
Seq2seq NMT

This state encodes the whole sentence

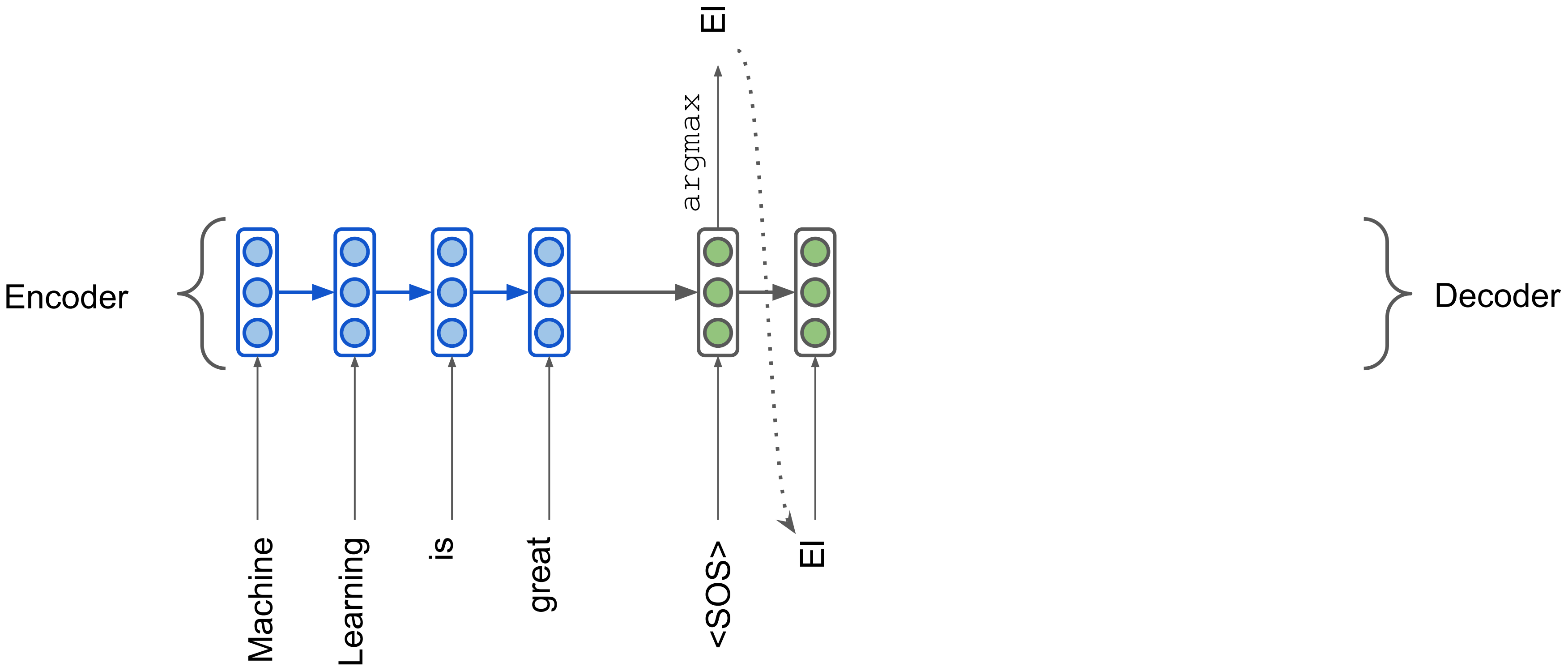



Machine
Learning
is
great

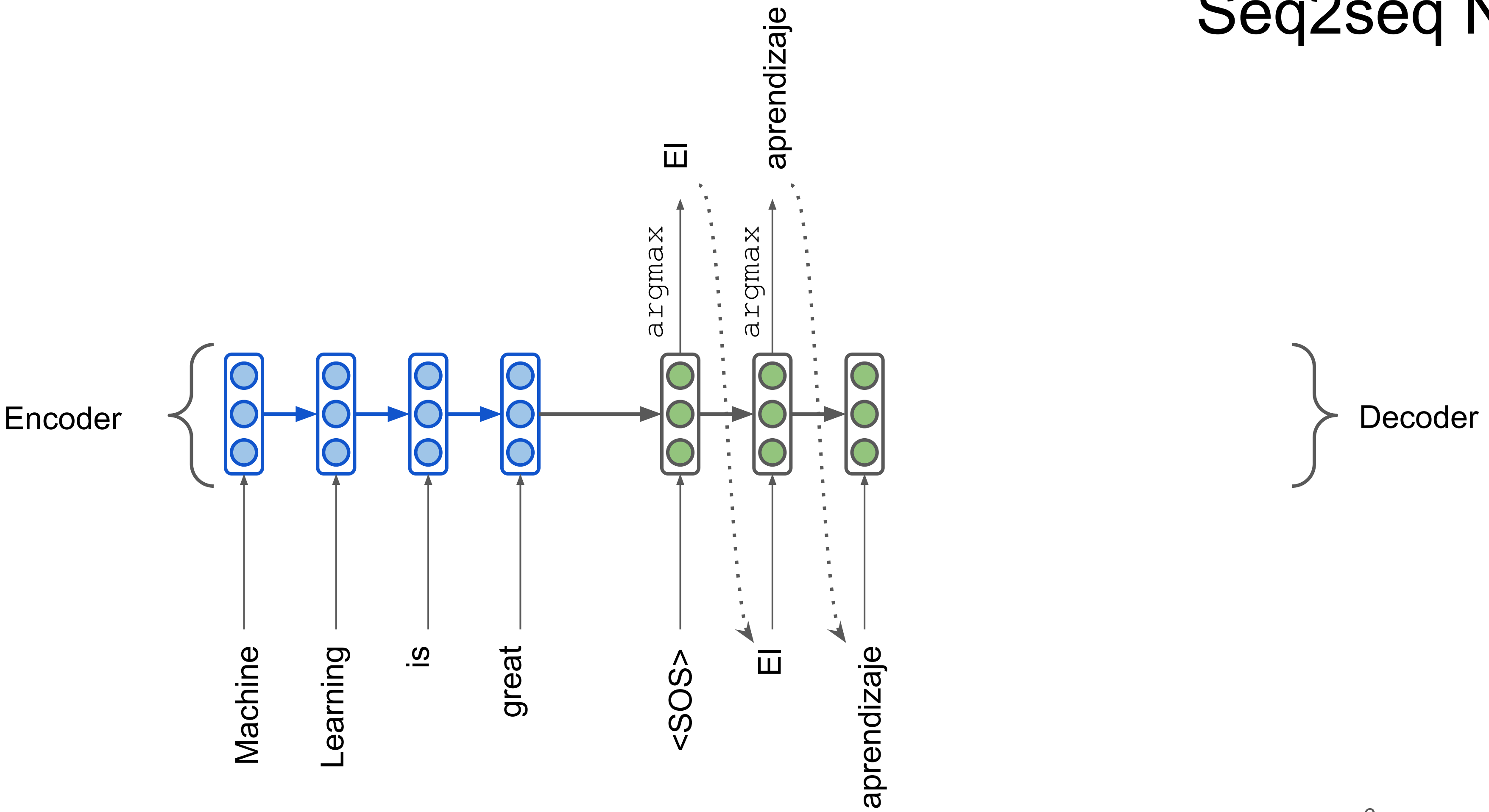
Seq2seq NMT


Forwarded as initial
hidden state to decoder
COPY
Encoder
Decoder
Machine
Learning
is
great
<SOS>

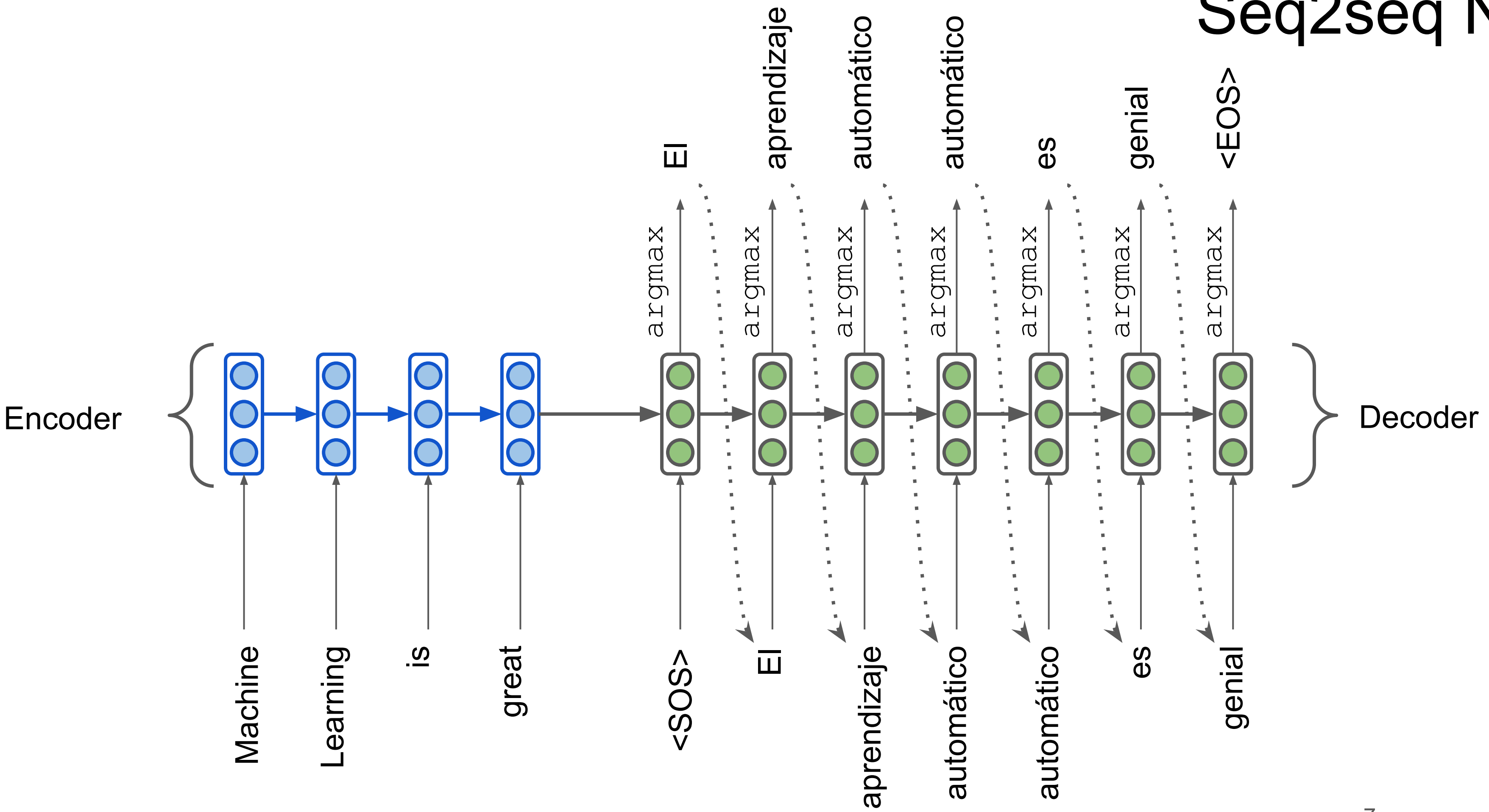
Seq2seq NMT



Seq2seq NMT

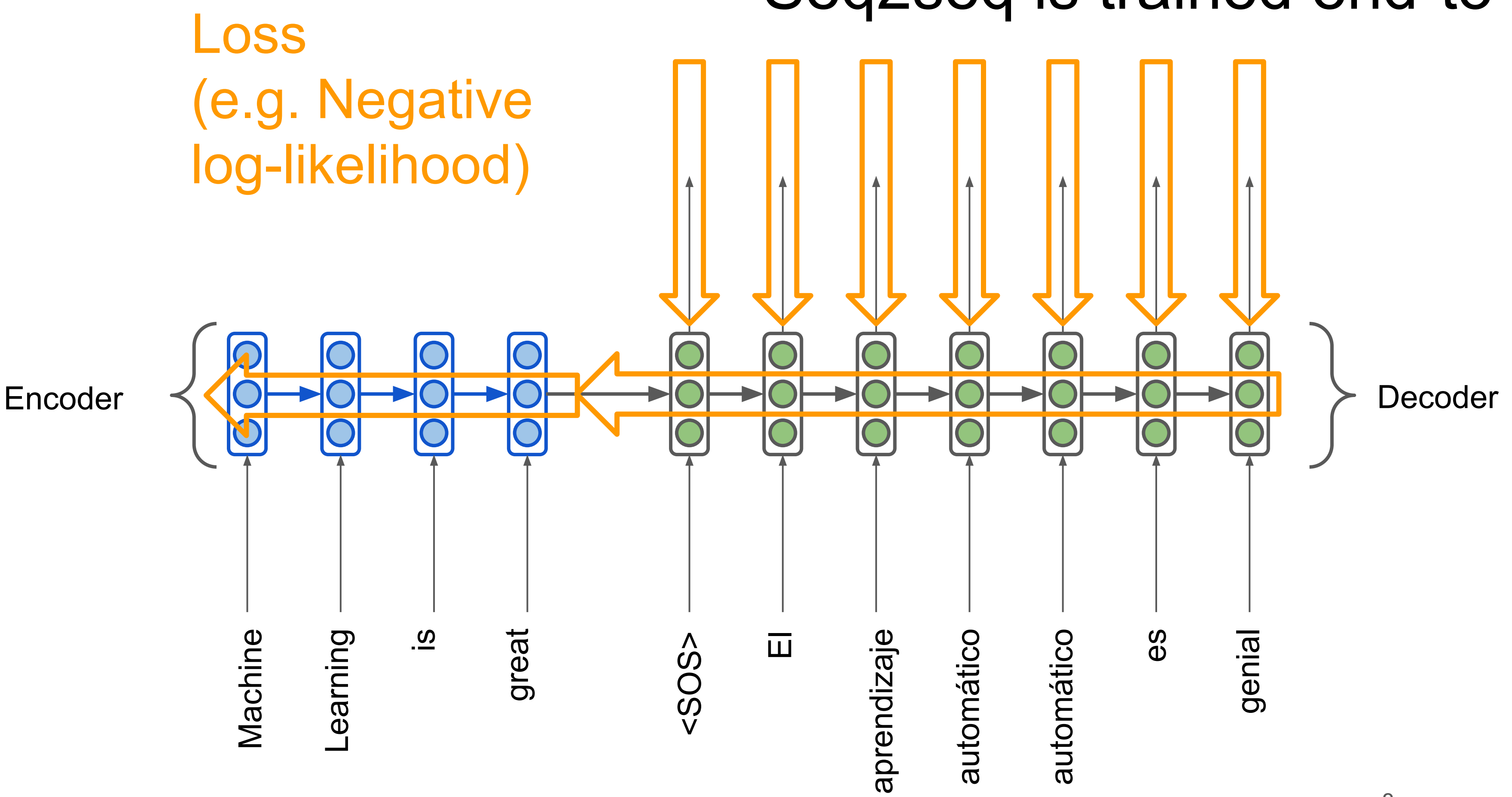

El
aprendizaje
argmax
argmax
Encoder
Decoder
Machine
Learning
is
great
<SOS>
El
aprendizaje

Seq2seq NMT



Encoder
Decoder
Machine
Learning
is
great
<SOS>
El
aprendizaje
automático
automático
es
genial
argmax
El
aprendizaje
automático
automático
es
genial
<EOS>

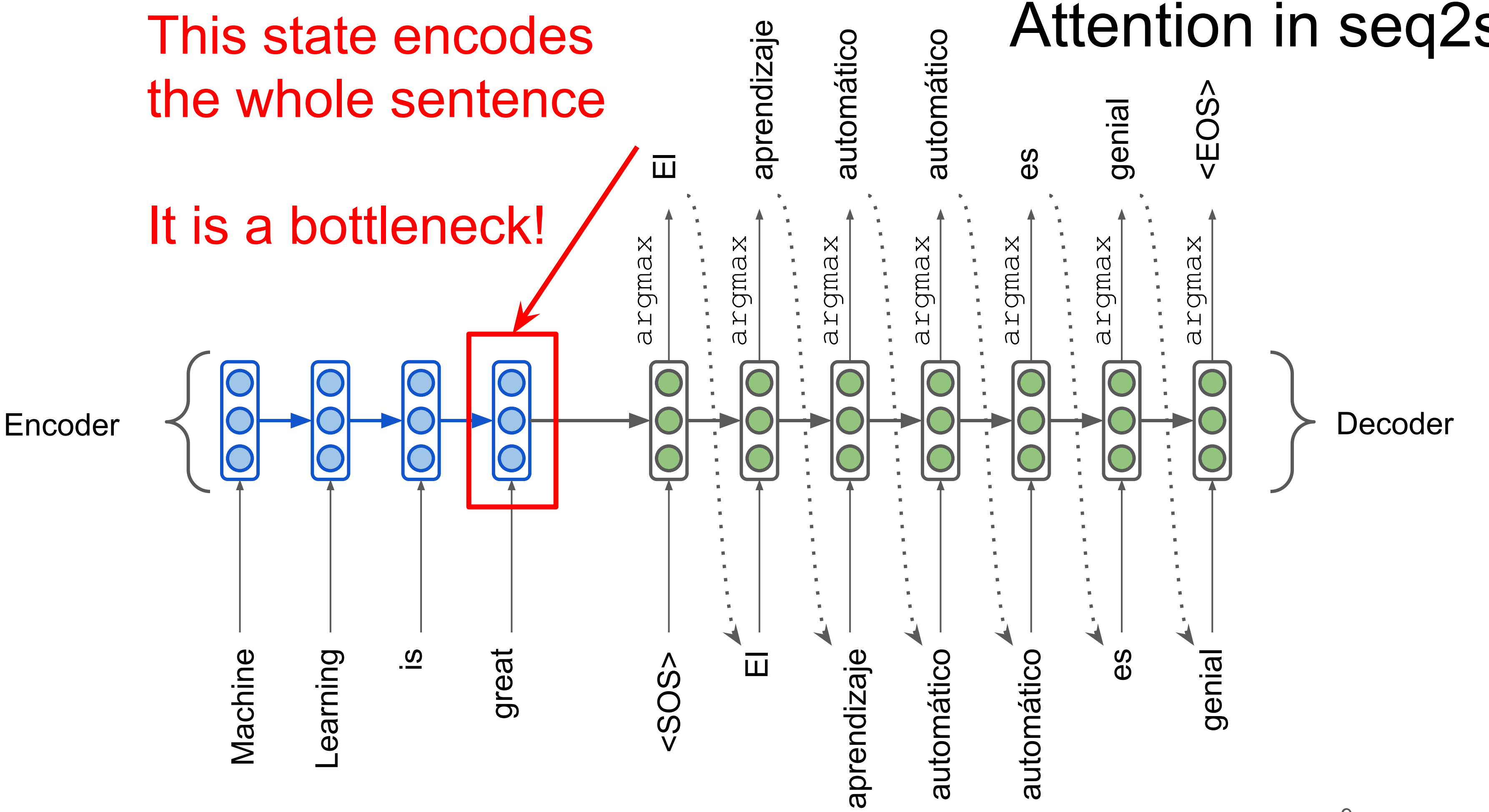
Seq2seq is trained end-to-end



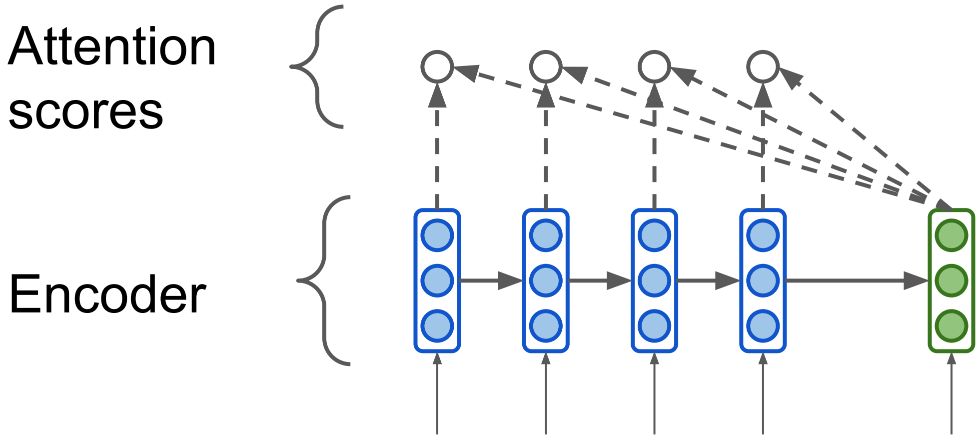
Attention in seq2seq

This state encodes the whole sentence

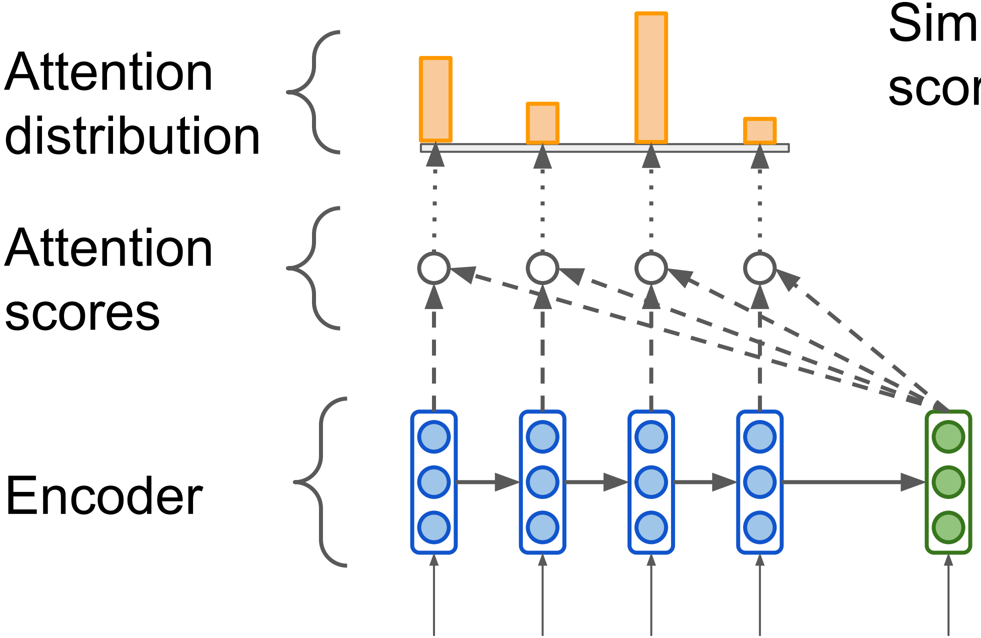
It is a bottleneck!



Seq2seq with attention

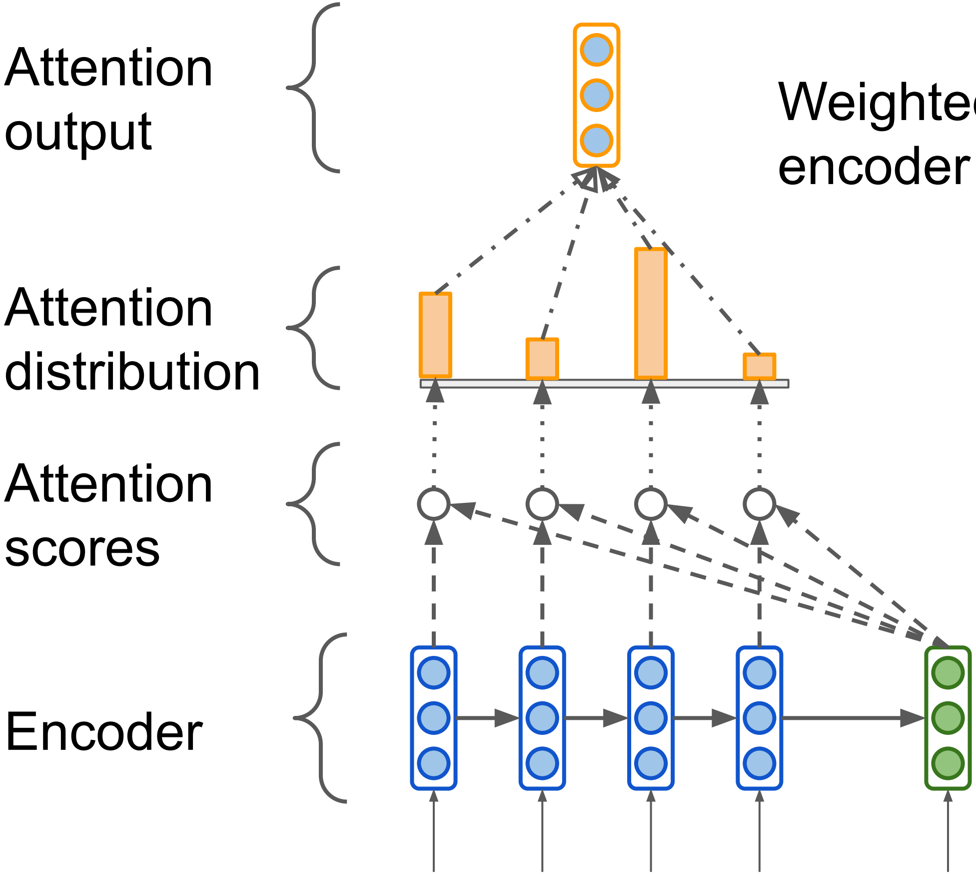


Seq2seq with attention



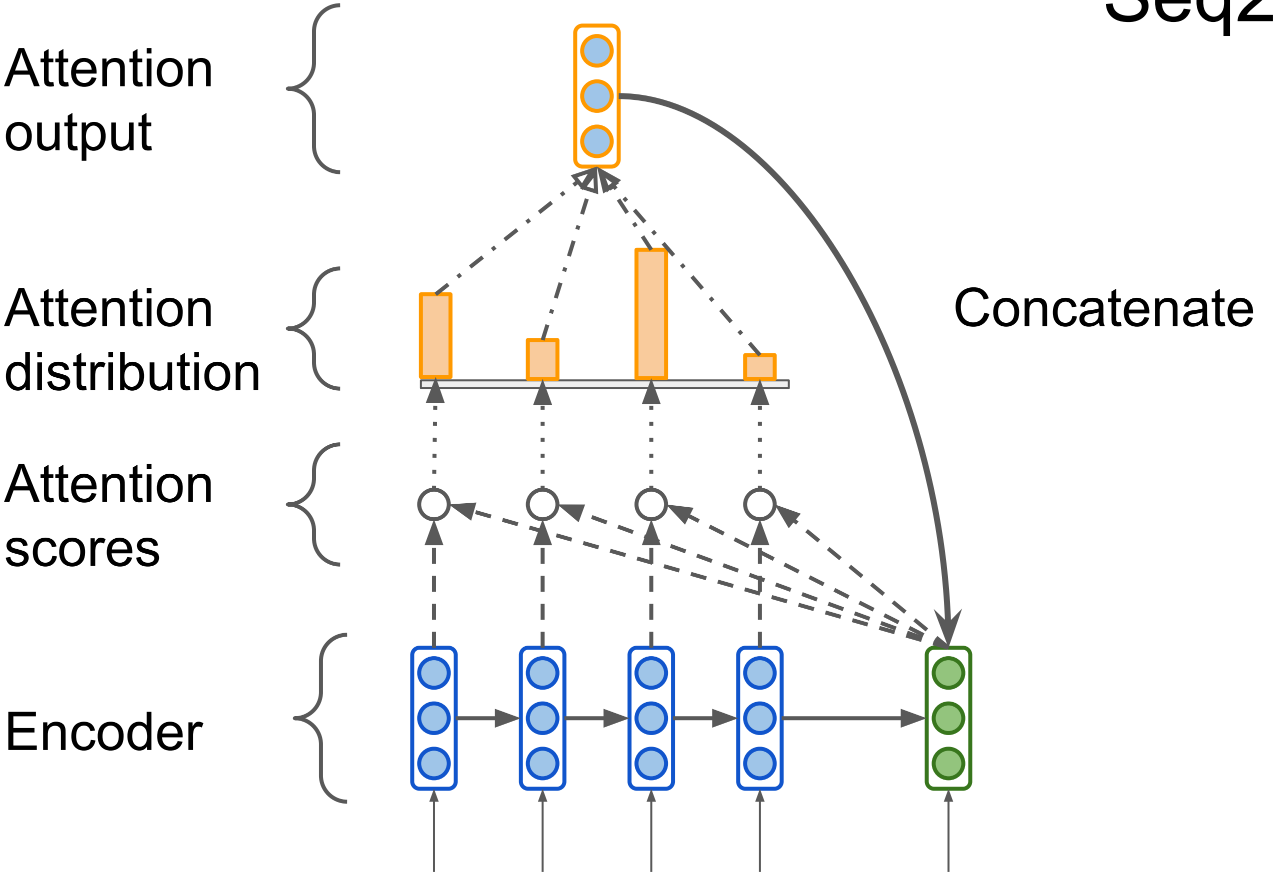
Simply apply softmax to scores

Seq2seq with attention

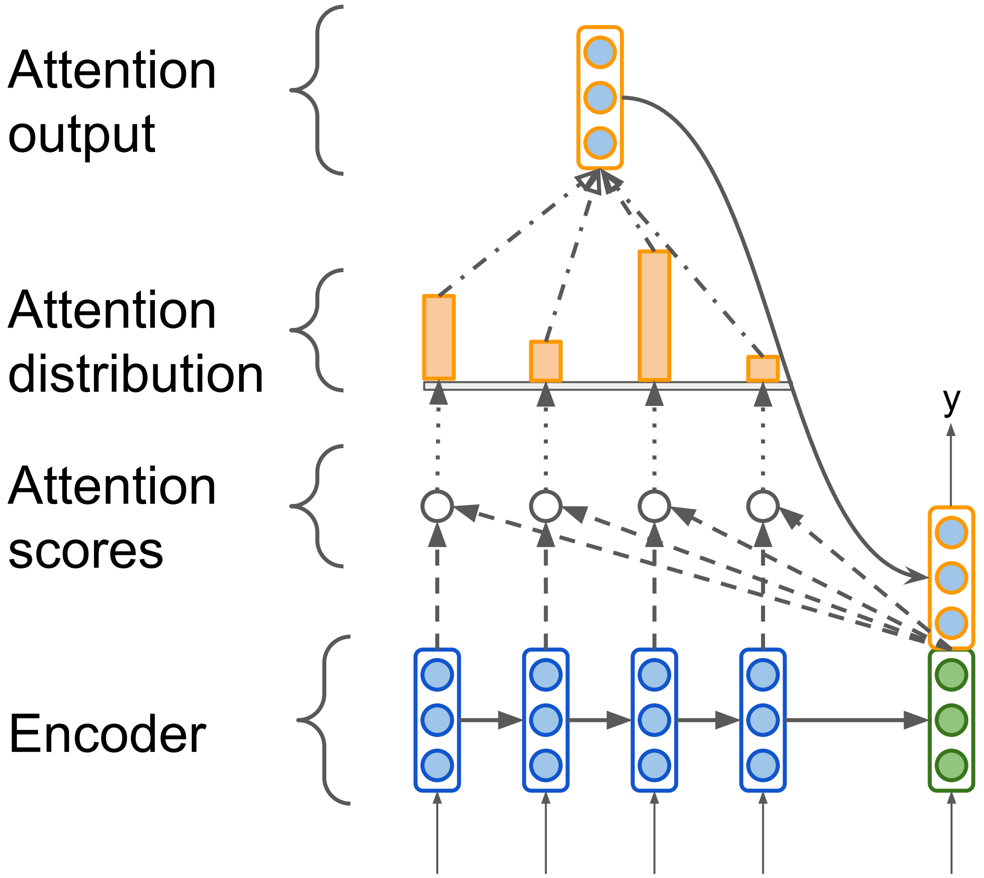


Weighted sum of all encoder states

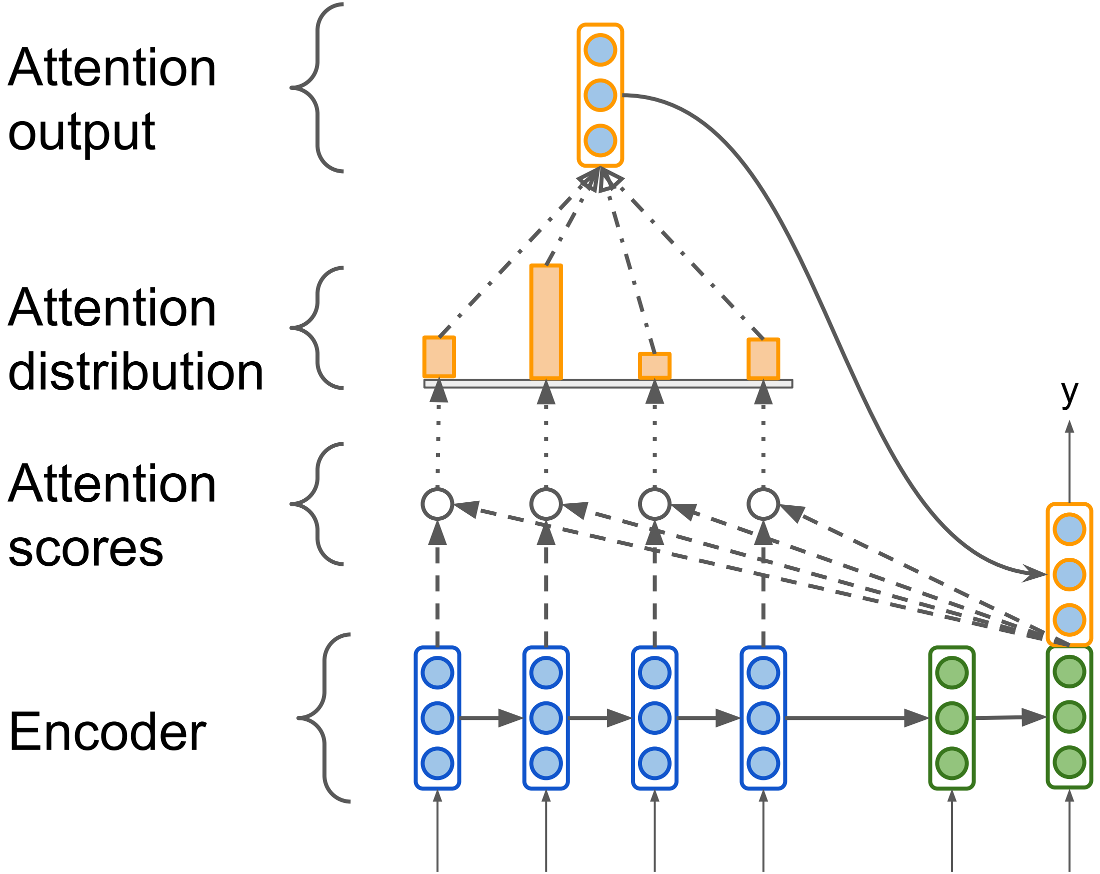
Seq2seq with attention



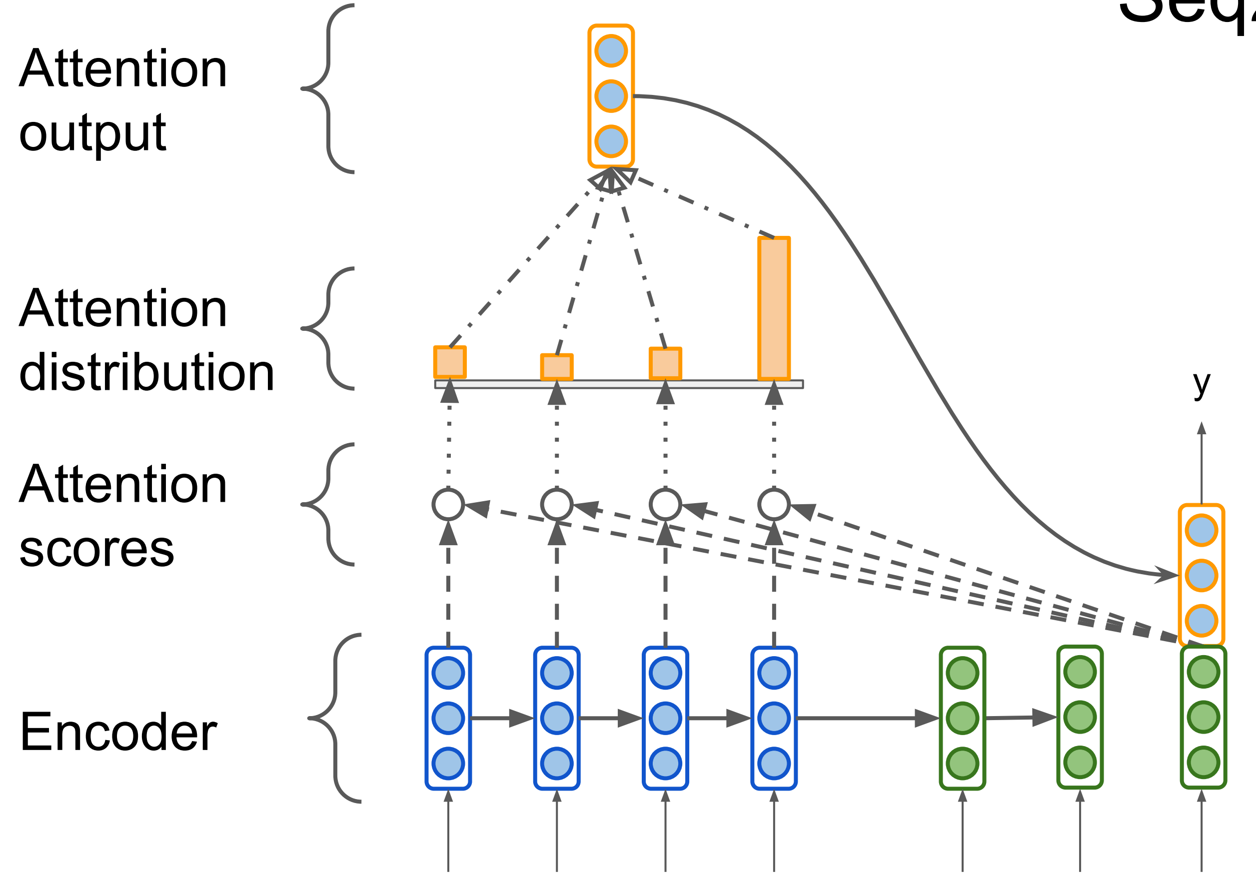
Seq2seq with attention



Seq2seq with attention



Seq2seq with attention



Attention in equations

Denote encoder hidden states $\mathbf{h}_1, \ldots, \mathbf{h}_N \in \mathbb{R}^k$

and decoder hidden state at time step t $\mathbf{s}_t \in \mathbb{R}^k$

The attention scores $\mathbf{e}^t$ can be computed as dot product

$$\mathbf{e}^t = [\mathbf{s}^T \mathbf{h}_1, \ldots, \mathbf{s}^T \mathbf{h}_N]$$

Then the attention vector is a linear combination of encoder states

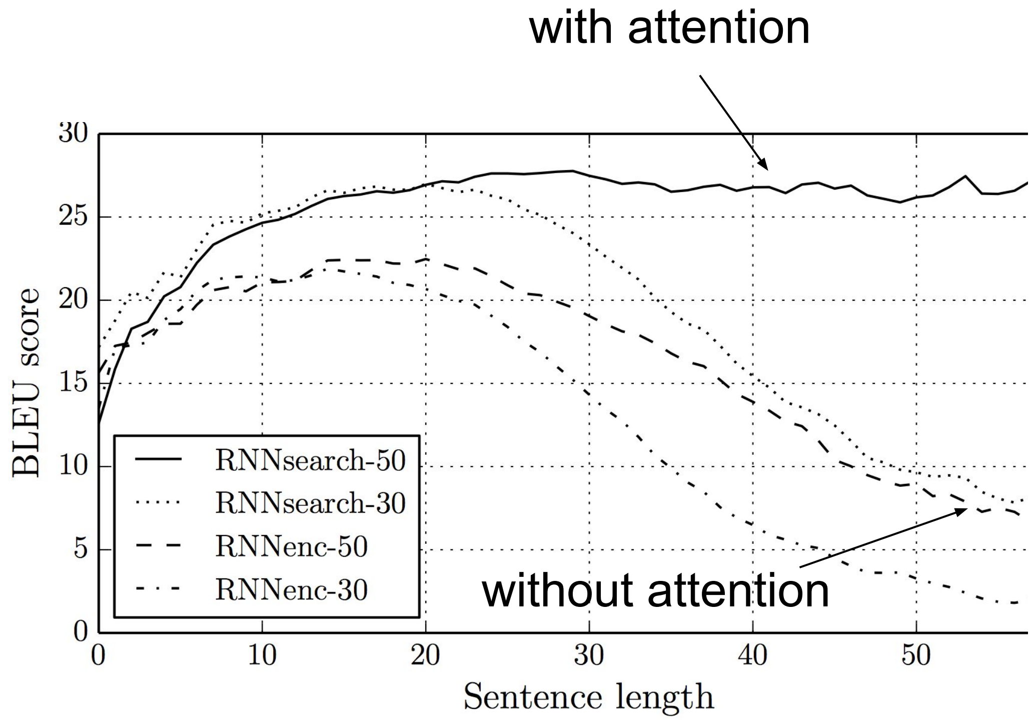
$$\mathbf{a}_t = \sum_{i=1}^{N} \boldsymbol{\alpha}_i^t \mathbf{h}_i \in \mathbb{R}^k$$, where $\boldsymbol{\alpha}_t = \mathrm{softmax}(\mathbf{e}_t)$

Attention variants

- Basic dot-product (the one discussed before): $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
    - $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ - weight matrix
- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
    - $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ - weight matrices
    - $\boldsymbol{v} \in \mathbb{R}^{d_3}$ - weight vector

Attention advantages

- "Free" word alignment
- Better results on long sequences



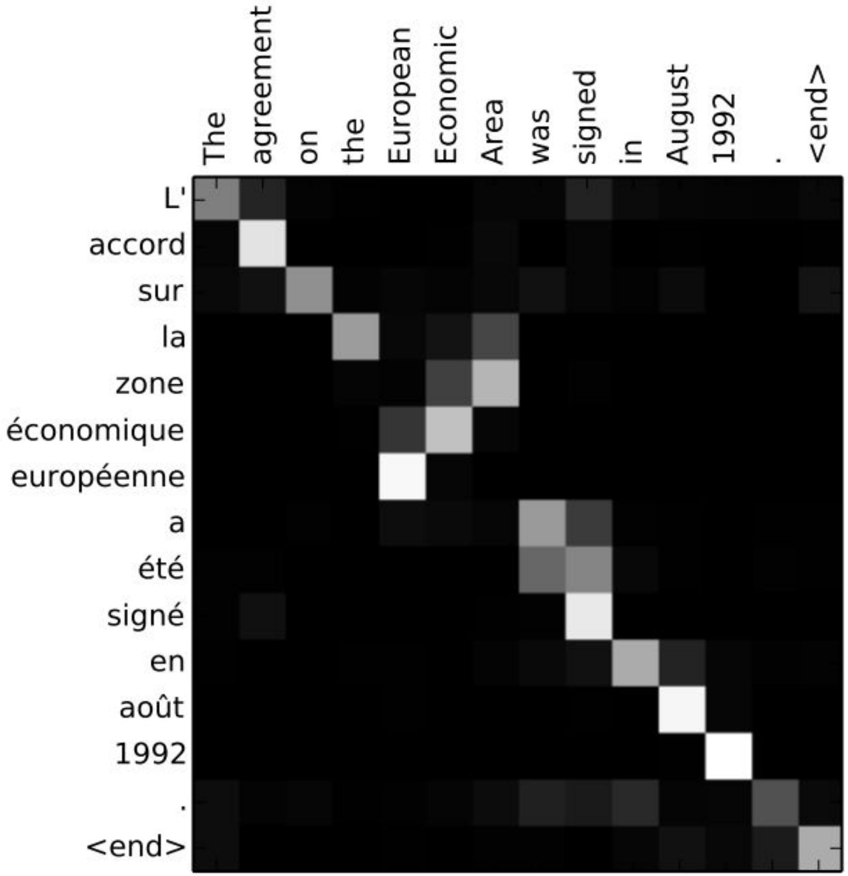


Image source: [Neural Machine Translation by Jointly Learning to Align and Translate]()

The Transformer

The Transformer

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

The Transformer

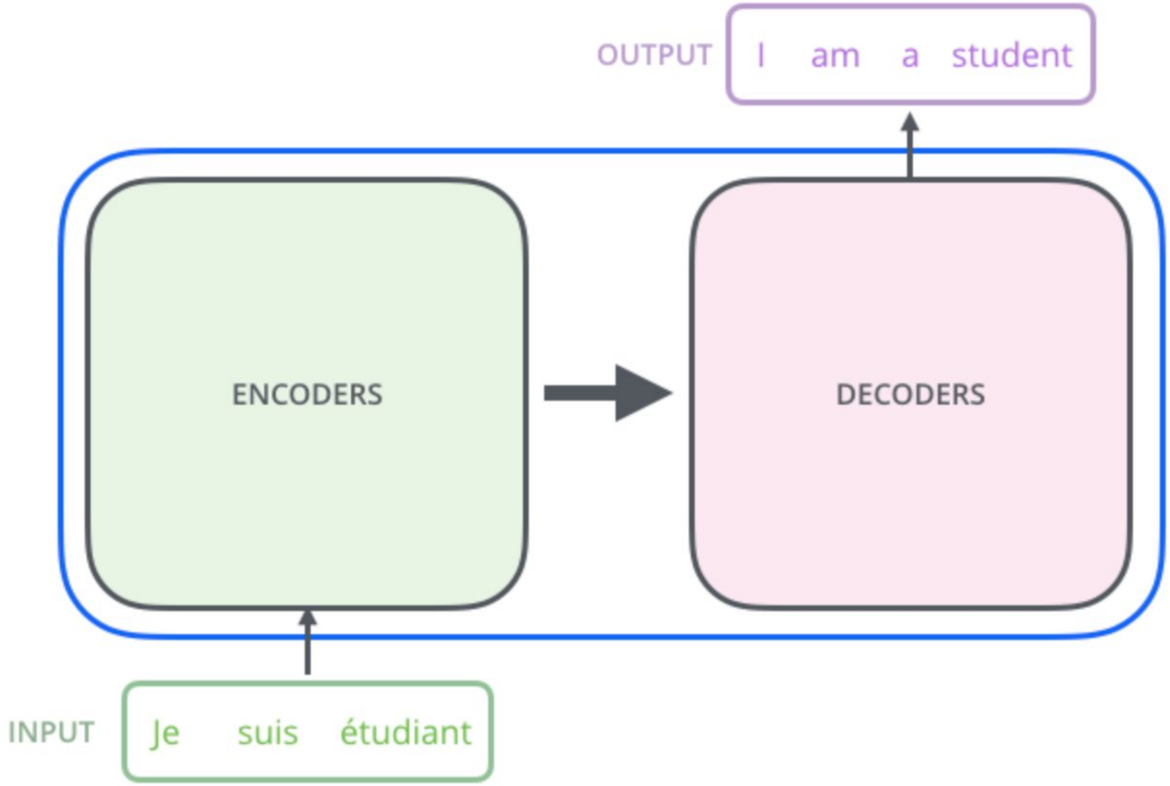


Image source: https://jalammar.github.io/illustrated-transformer/

The Transformer

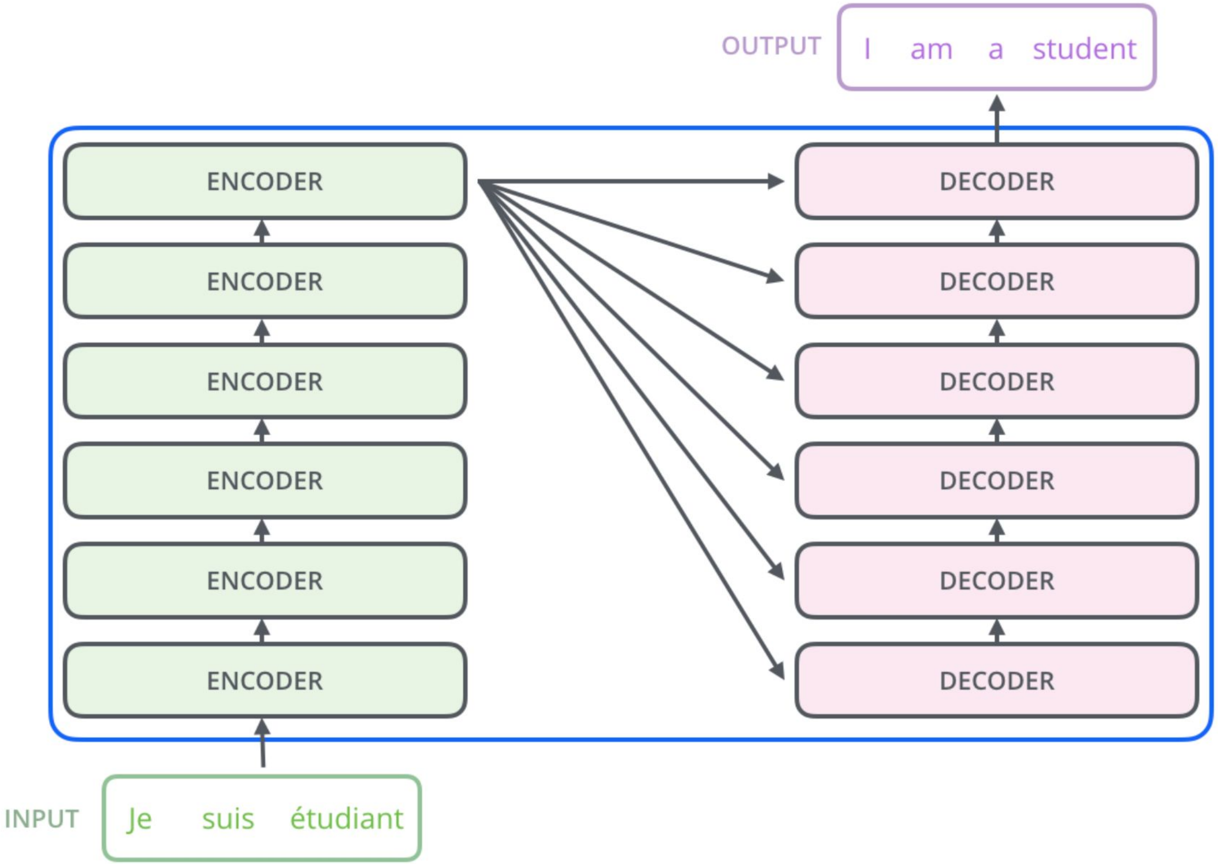
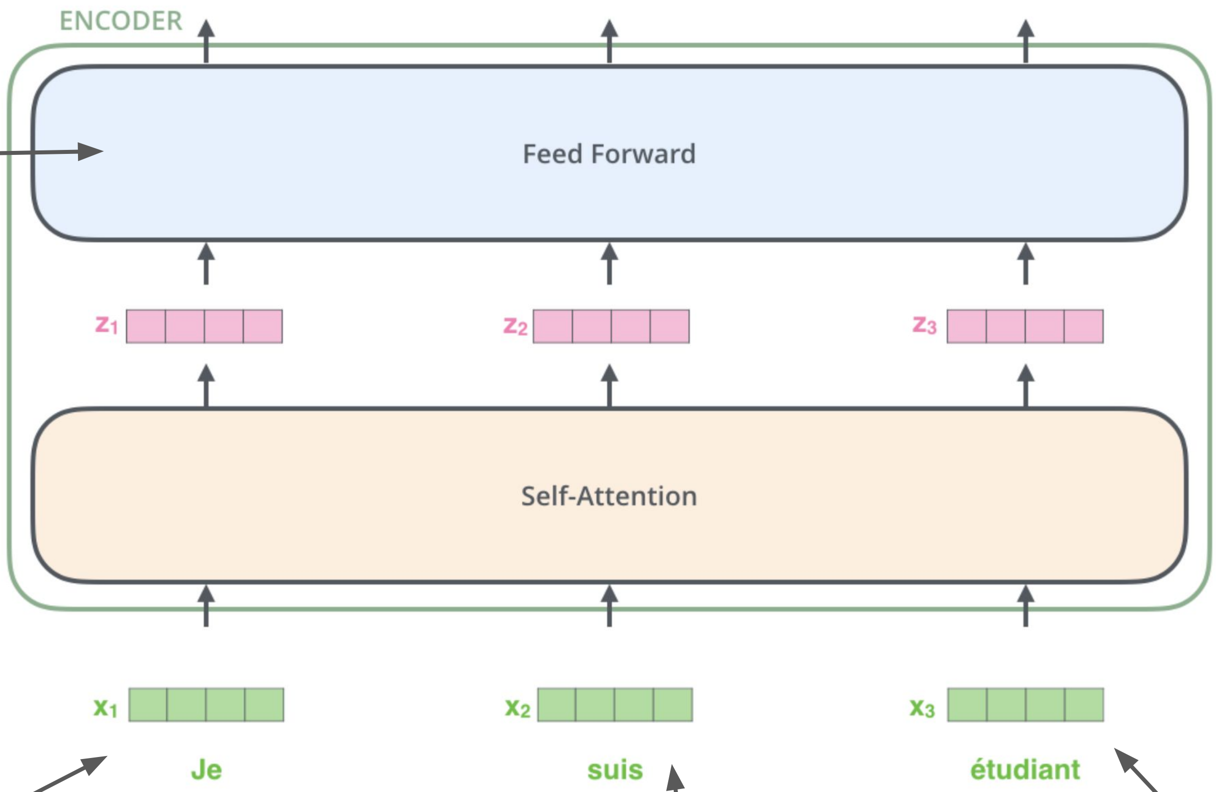


Image source: https://jalammar.github.io/illustrated-transformer/

The Encoder Side

Can be parallelized

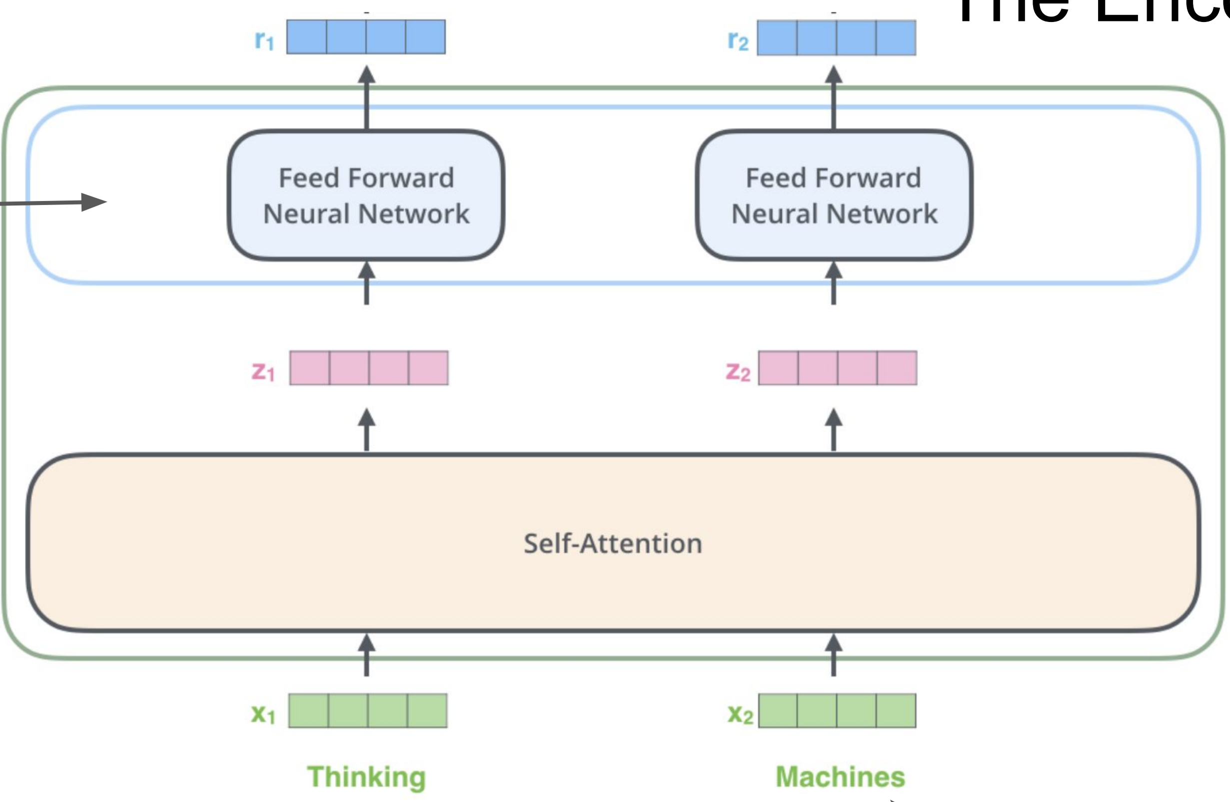


the word in each position flows through its own path in the encoder

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

The Encoder Side

Can be parallelized



the word in each position flows through its own path in the encoder

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

The Transformer: quick overview

- Proposed in 2017 in paper Attention is All You Need by Ashish Vaswani et al.
- No recurrent or convolutional layers, only attention
- Beats seq2seq in machine translation task
    - *28.4 BLEU on the WMT 2014 English-to-German translation task*
- Much faster
- Uses **self-attention** concept

Self-Attention

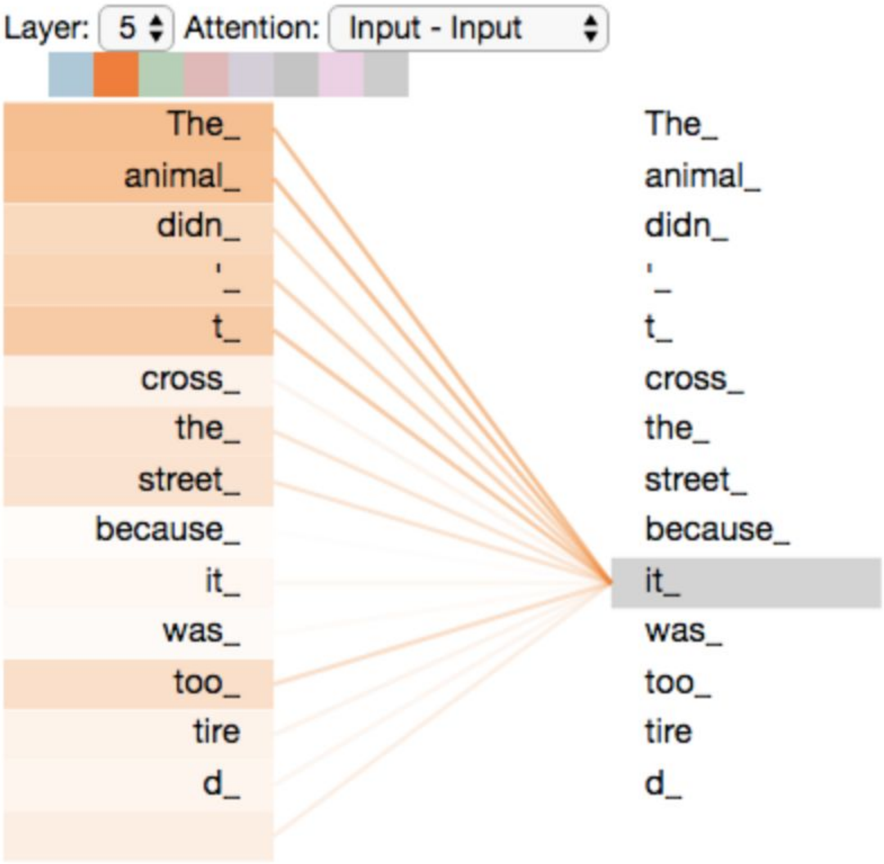
Self-Attention at a High Level

”`The animal didn't cross the street because it was too tired`”

- What does “it” in this sentence refer to?
- We want self-attention to associate “it” with “animal”

- Self-attention is the method the Transformer uses to bake the “understanding” of other relevant words into the one we’re currently processing

Self-Attention at a High Level



Self-Attention: detailed explanation

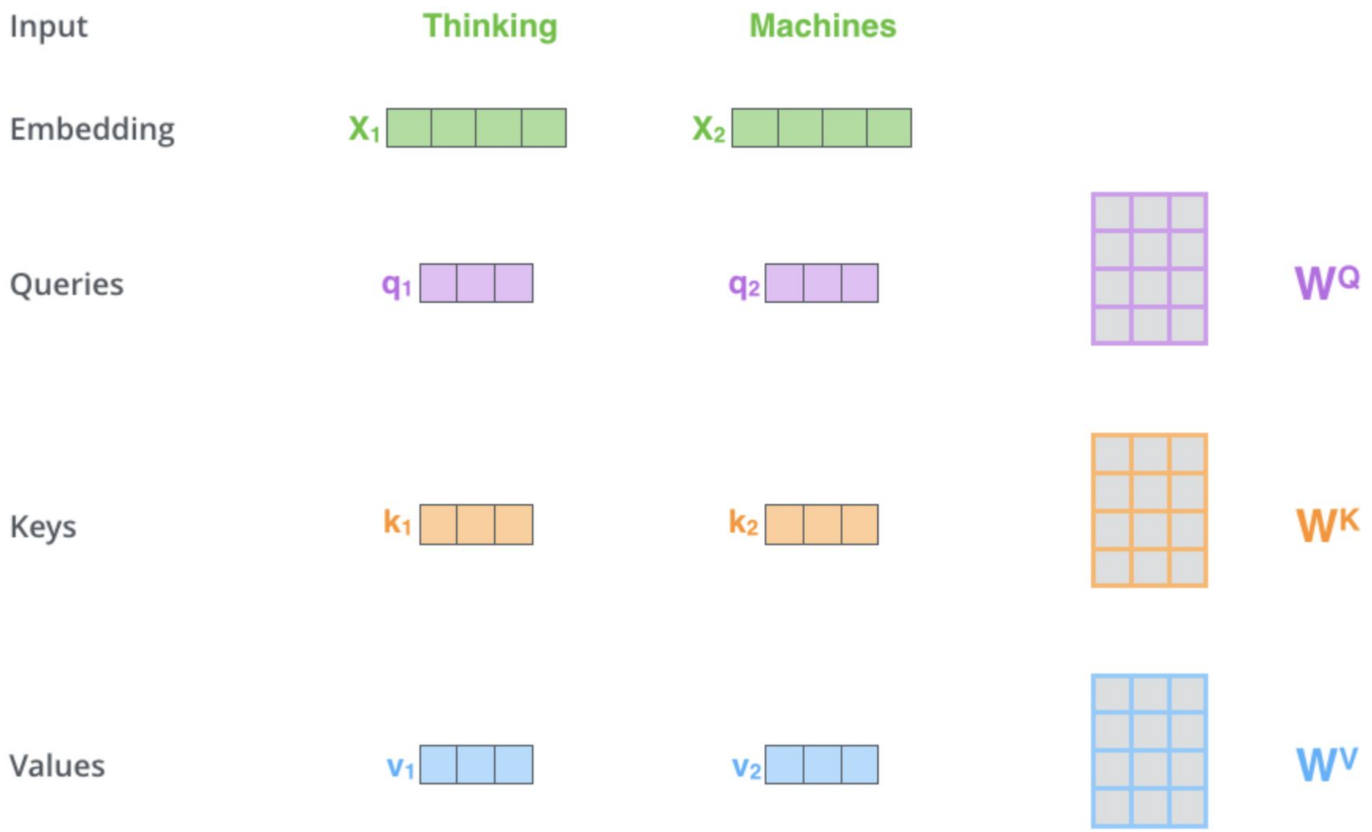


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: detailed explanation

STEP 1:

create 3 vectors (**query**, **key**, **value**)

from each of the encoder's input vectors

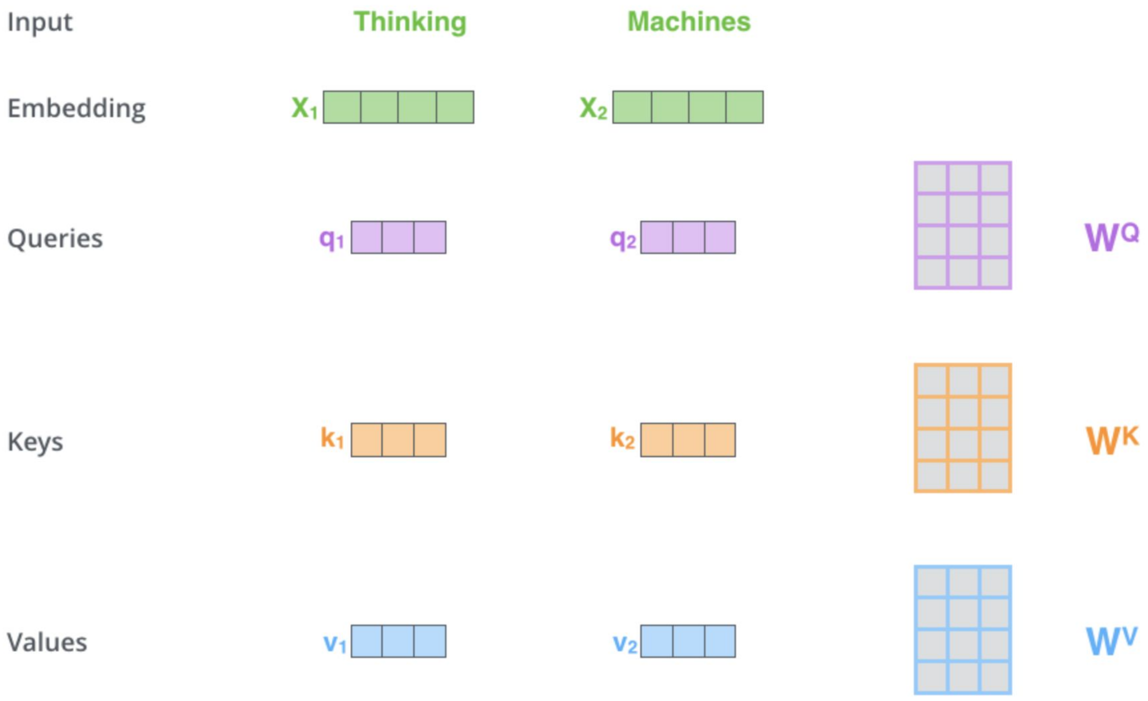


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: detailed explanation

What are the **query**, **key**, **value** vectors?

They're abstractions that are useful for

calculating and thinking about attention.

Self-Attention: detailed explanation

STEP 2:

calculate a score

(score each word of the input sentence against the current word)

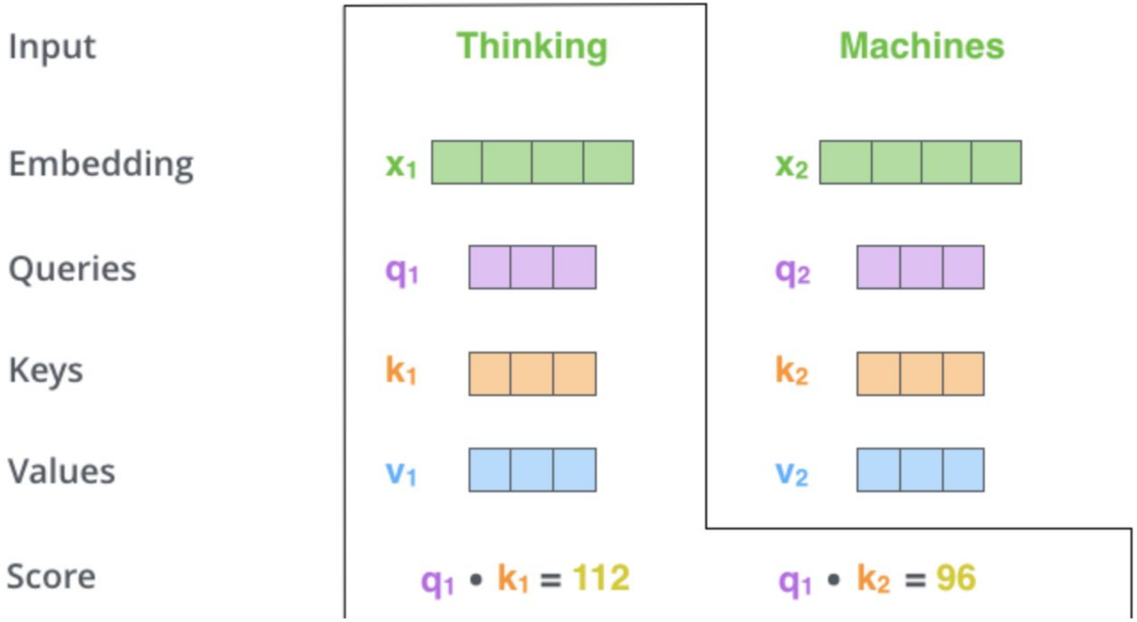


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: detailed explanation

STEP 3:

divide the scores by 8

(the square root of the dimension of the key vectors)

STEP 4:

softmax

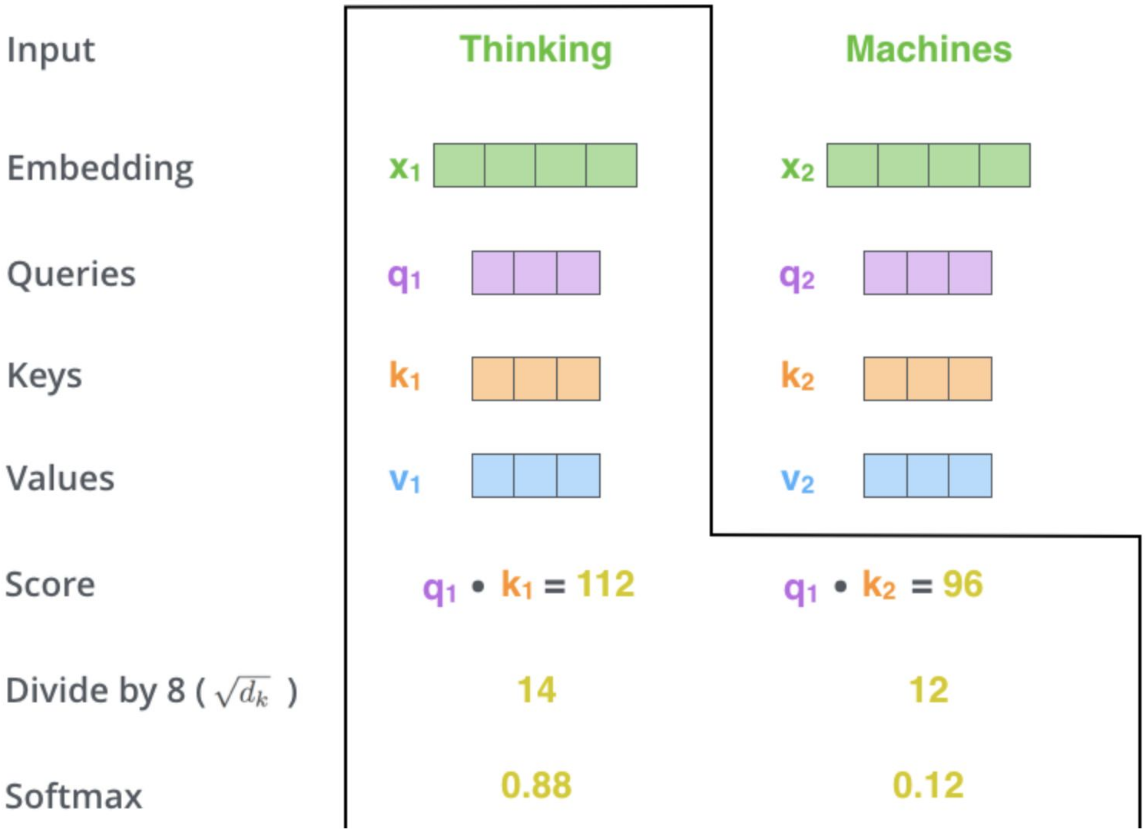


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention: detailed explanation

STEP 5:

multiply each value vector by the softmax score

STEP 6:

sum up the weighted value vectors

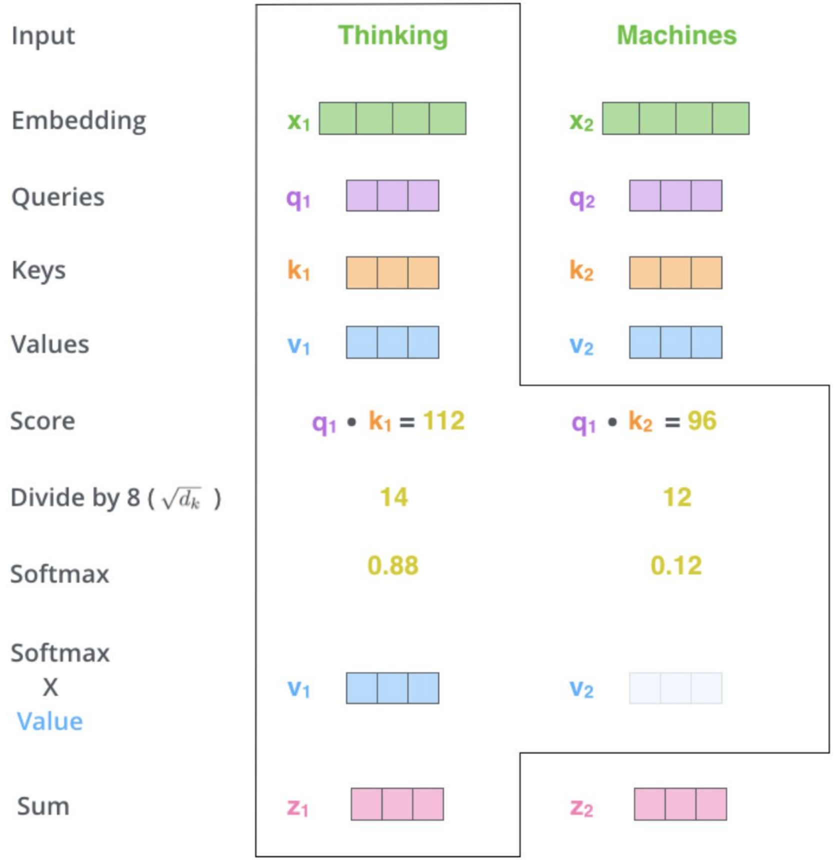
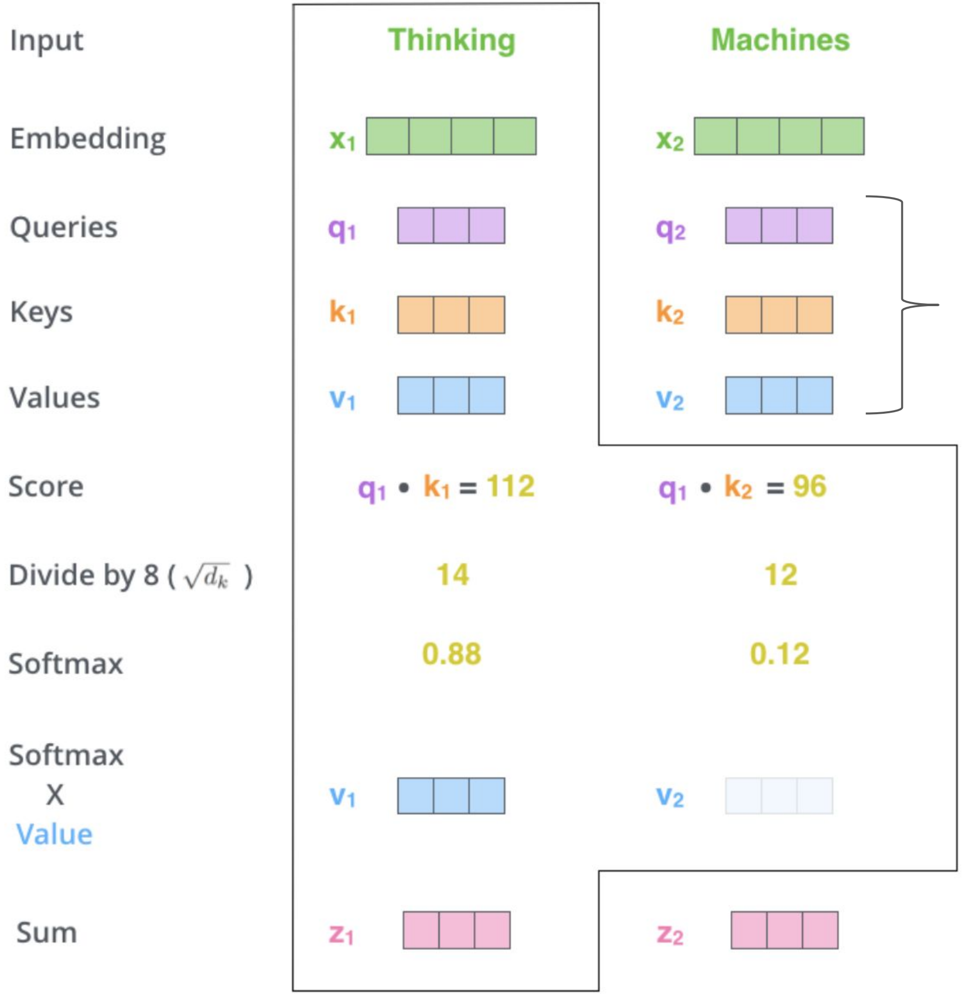


Image source: https://jalammar.github.io/illustrated-transformer/

Self-Attention



STEP 1: create Query, Key, Value

STEP 2: calculate scores

STEP 3: divide by $\sqrt{d_k}$

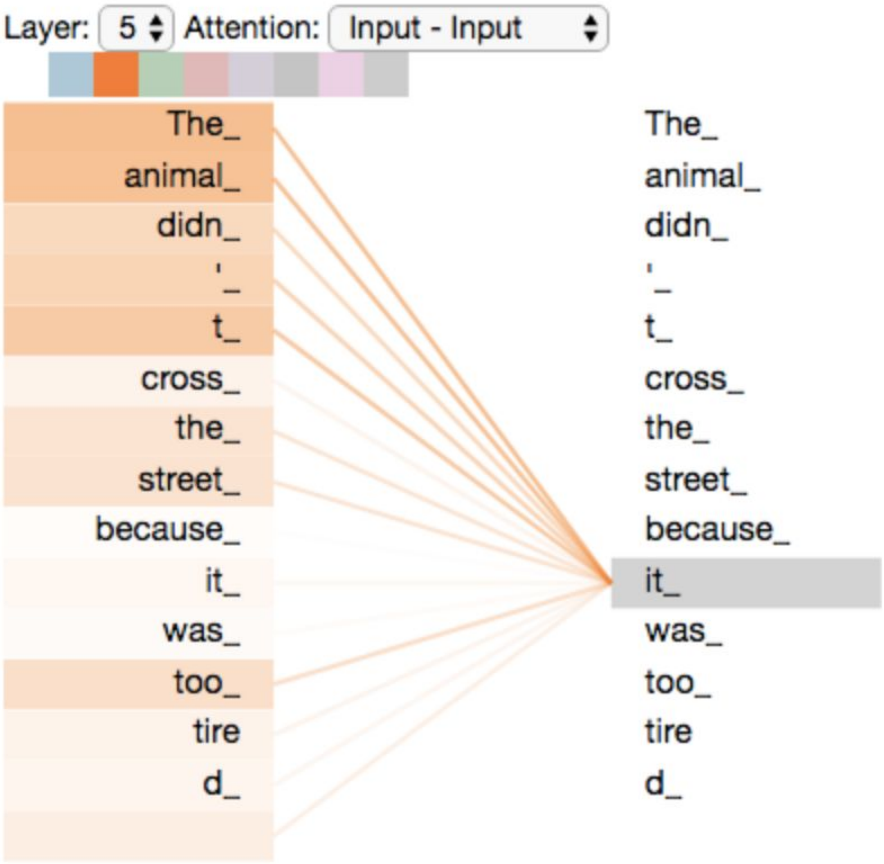
STEP 4: softmax

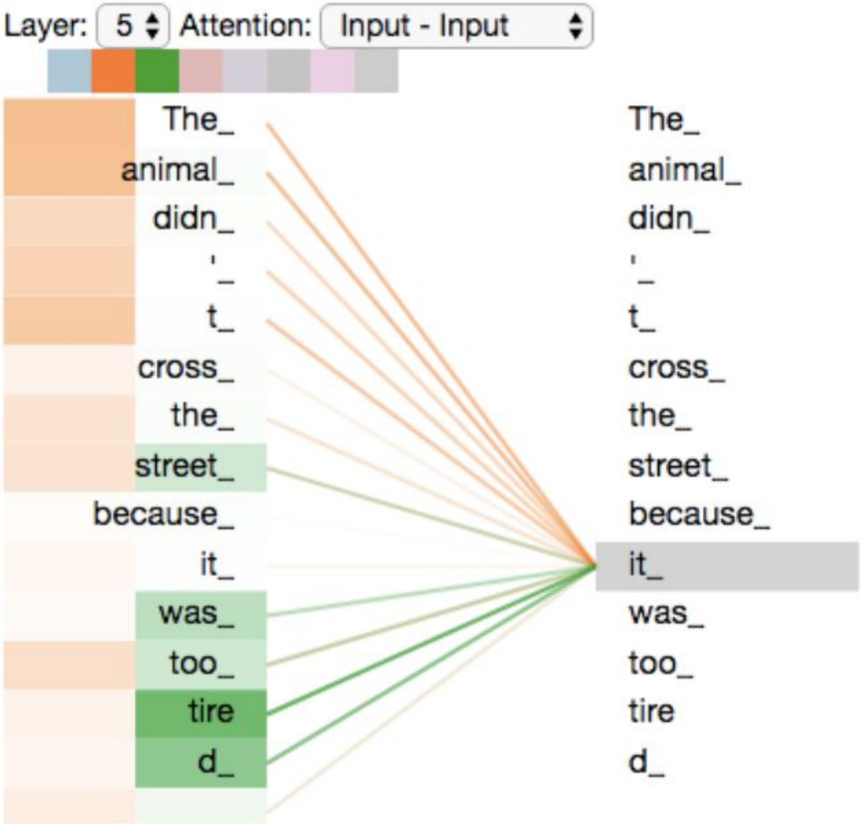
STEP 5: multiply each value vector by the softmax score

STEP 6: sum up the weighted value vectors

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

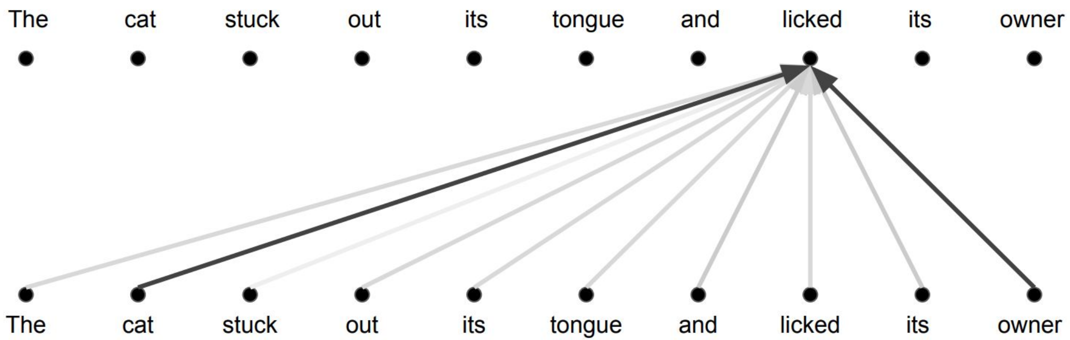
Multi-Head Attention





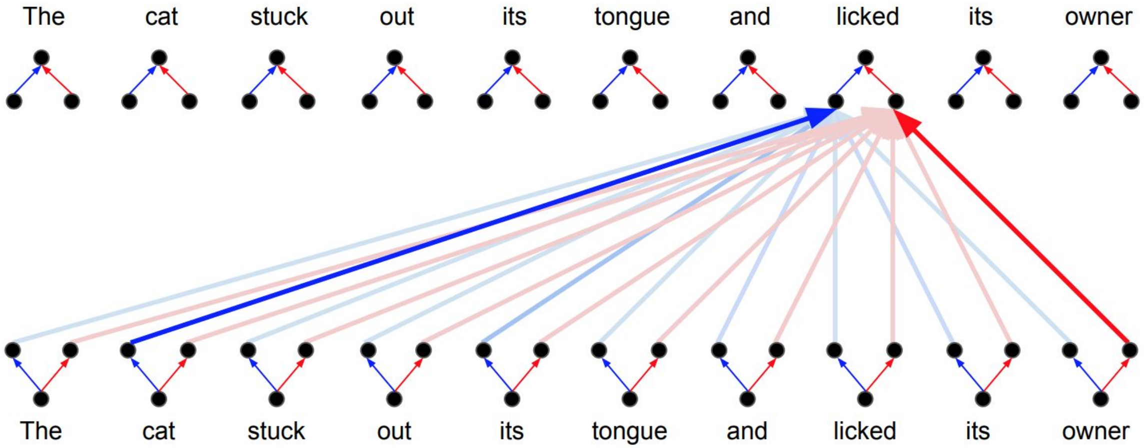
Attention vs. Multi-Head Attention

Attention: a weighted average



Multi-Head Attention:

parallel attention layers with different linear transformations on input and output.



Performance: WMT 2014 BLEU

	EN-DE	EN-FR
GNMT (orig)	24.6	39.9
ConvSeq2Seq	25.2	40.5
Transformer*	**28.4**	**41.8**

*Transformer models trained $\boxed{\text{>3x faster}}$ than the others.

Research Challenges

- Constant ‘path length’ between any two positions.
- Unbounded memory.
- Trivial to parallelize (per layer).
- Models Self-Similarity.
- Relative attention provides expressive timing, equivariance, and extends naturally to graphs.

Positional Encoding

Positional Encoding: why sin and cos?

$$\vec{p_t}^{(i)} = f(t)^{(i)} = \begin{cases} \sin(\omega_k t), & \text{if } i = 2k \\ \cos(\omega_k t), & \text{if } i = 2k+1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$

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

t stays for position in the original sequence
k is the index of the element in the positional vector

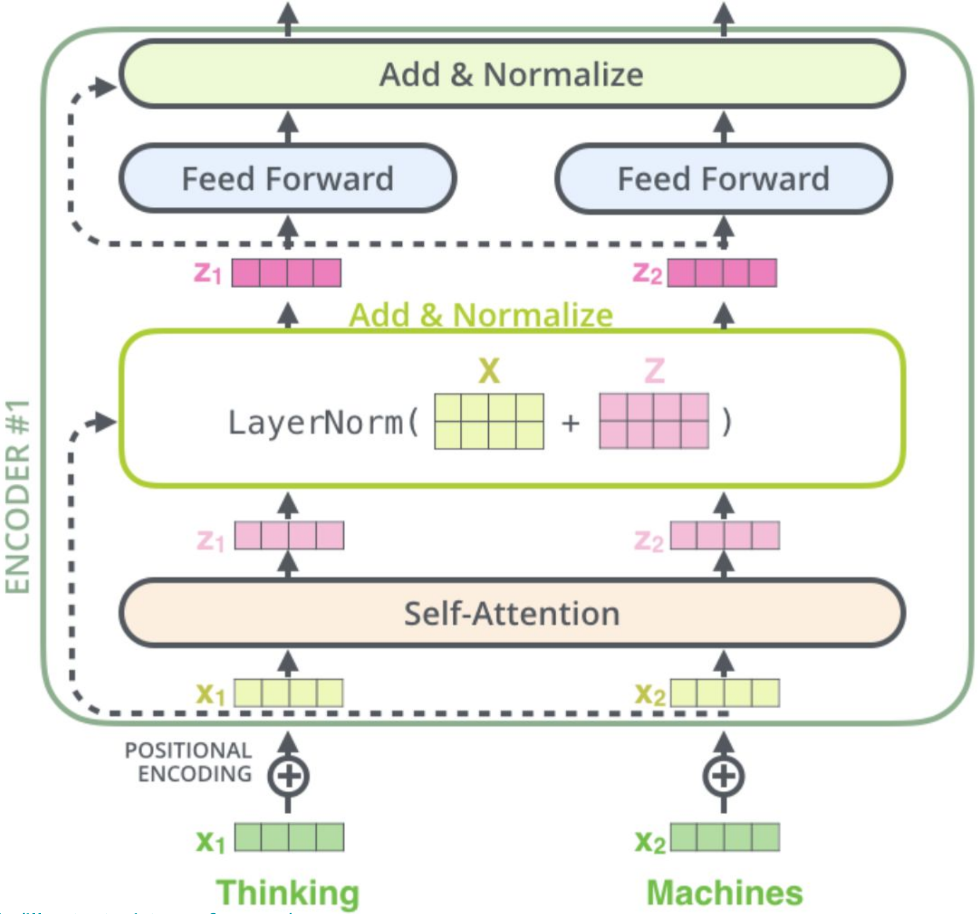
Image source: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Layer Normalization

Layer Normalization

Like BatchNorm

but normalize along all features representing latent vector



More info: [Layer Normalization]

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

The Decoder

The Decoder Side

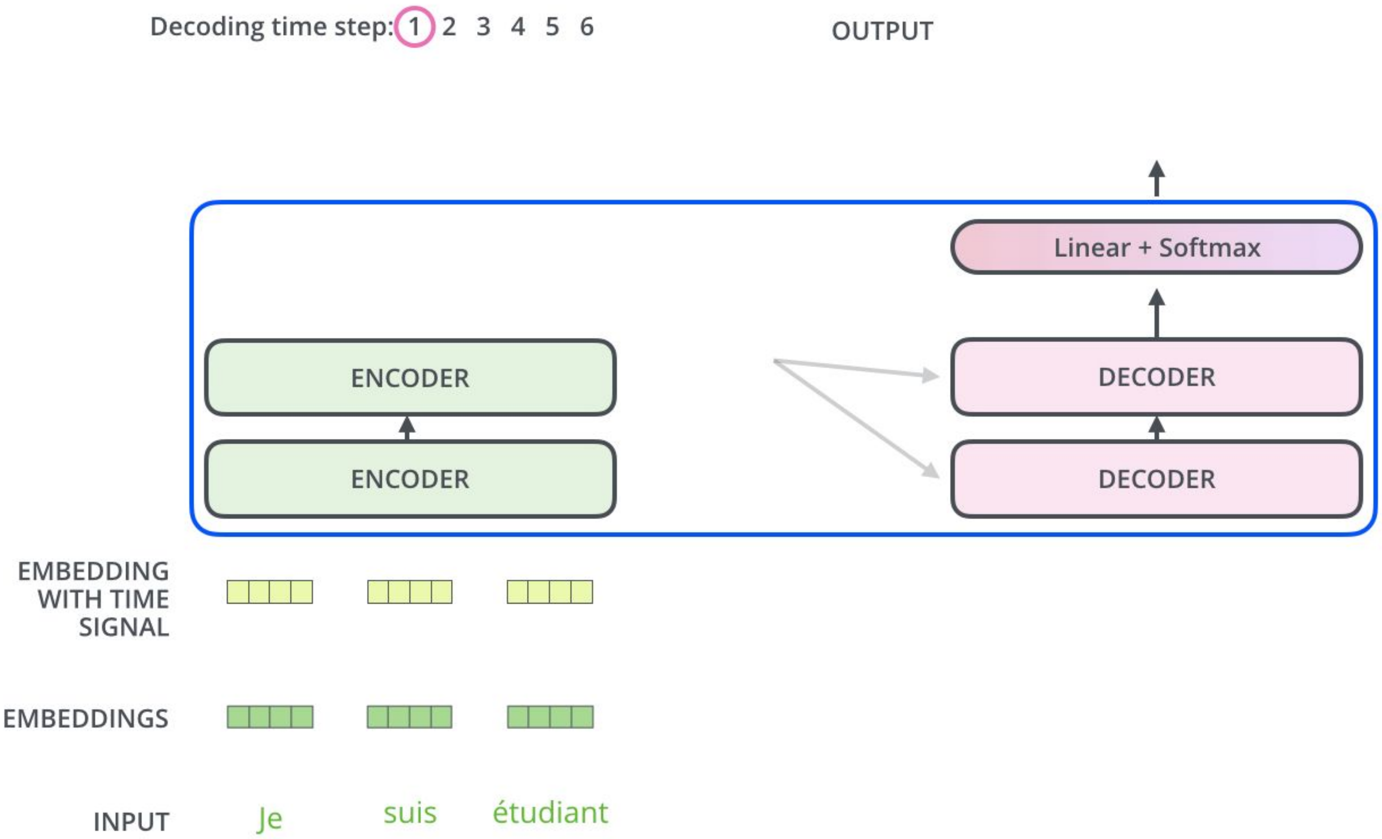
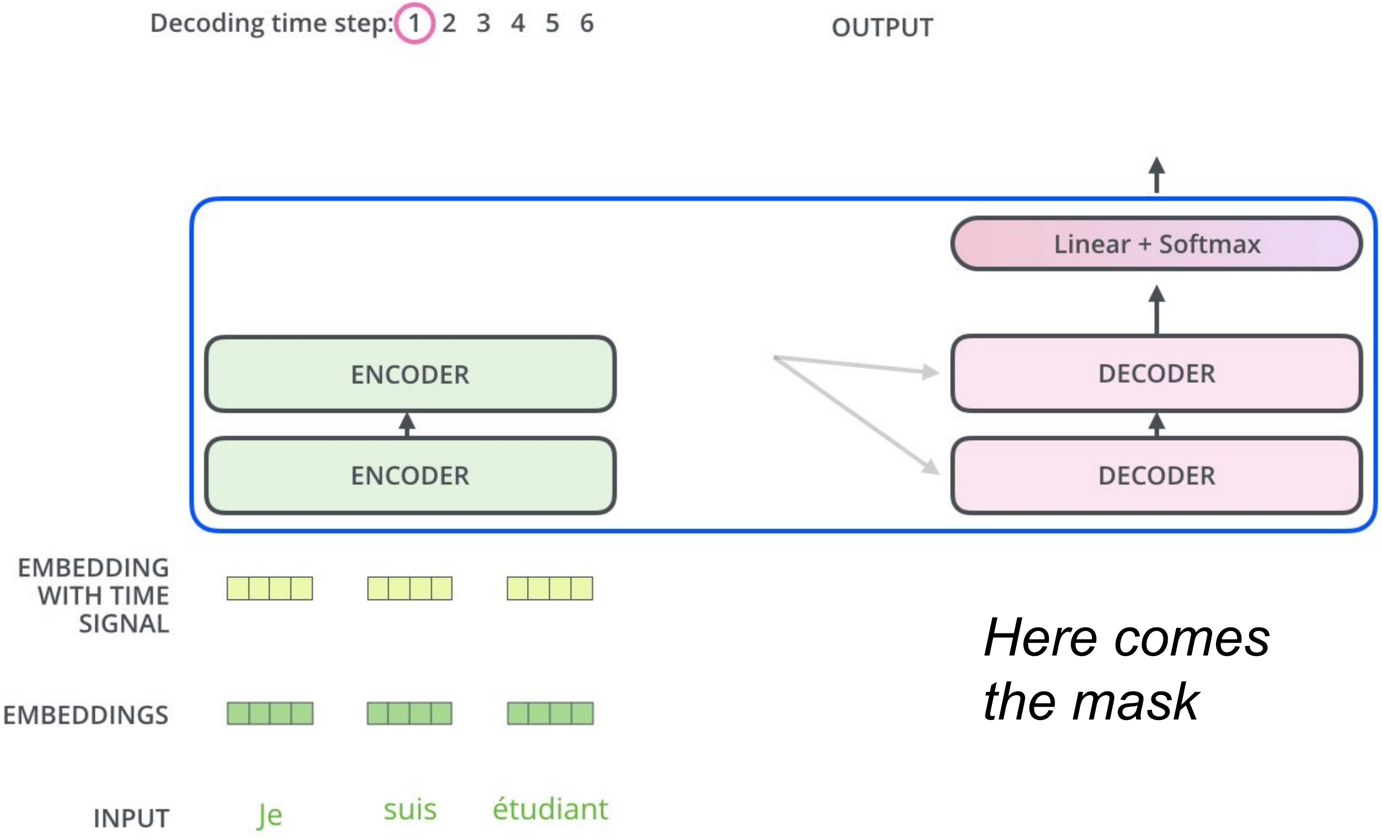


Image source: https://jalammar.github.io/illustrated-transformer/

The Decoder Side



Here comes the mask

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

The Decoder Side

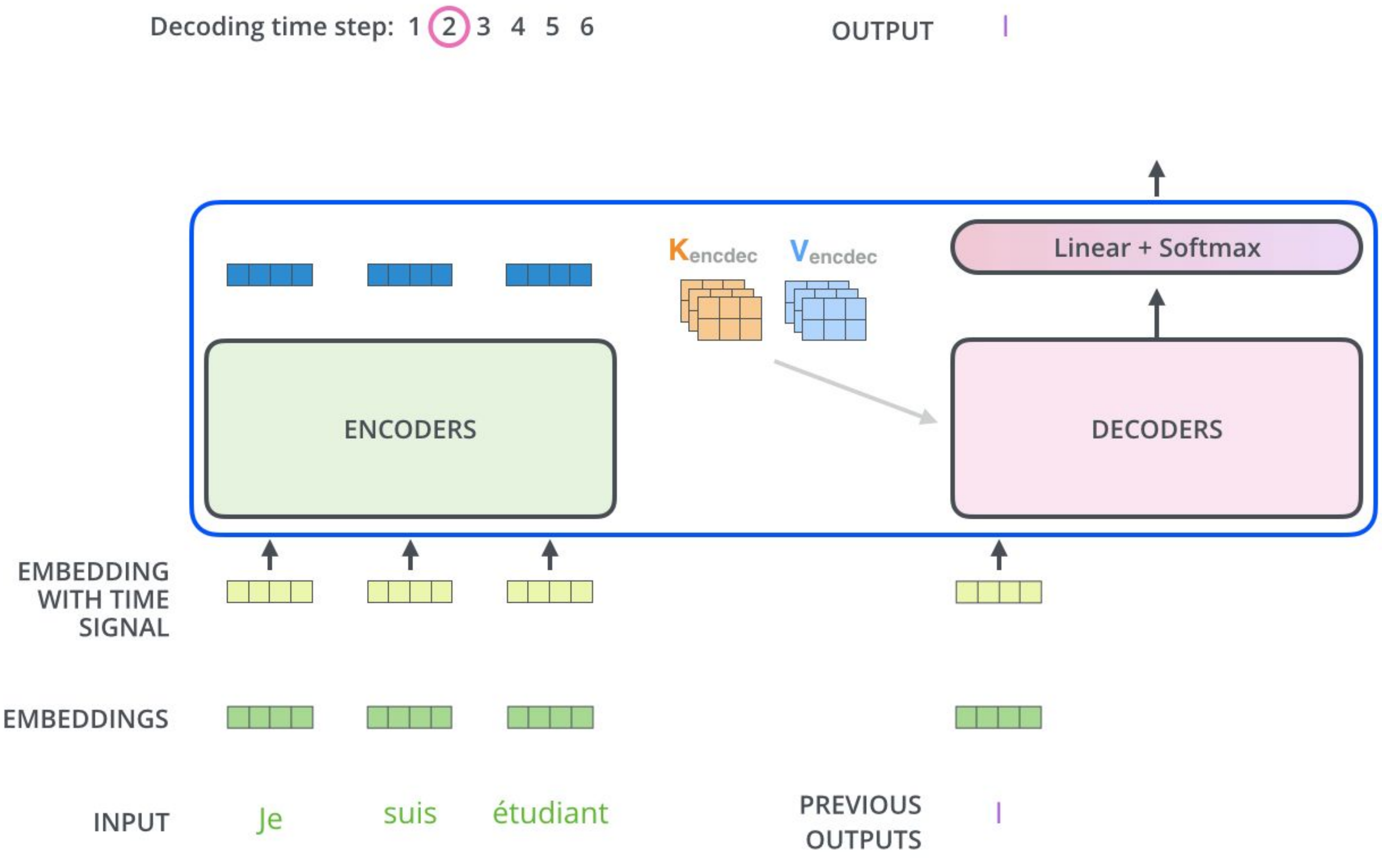


Image source: https://jalammar.github.io/illustrated-transformer/

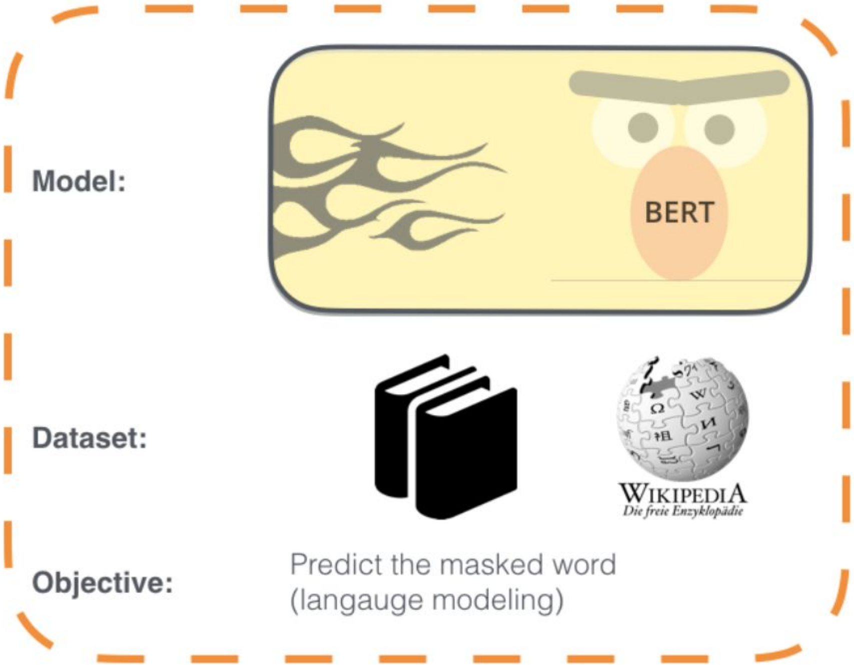
BERT

Bidirectional Encoder Representations from Transformers

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.

Semi-supervised Learning Step



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

Supervised Learning Step

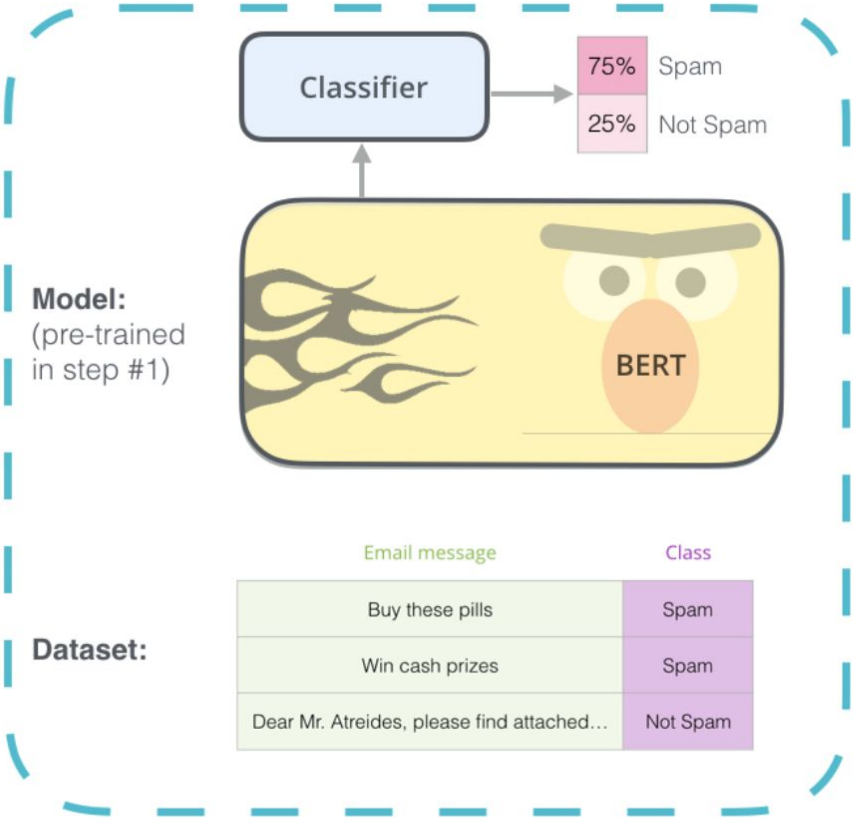


Image source: http://jalammar.github.io/illustrated-bert/

BERT

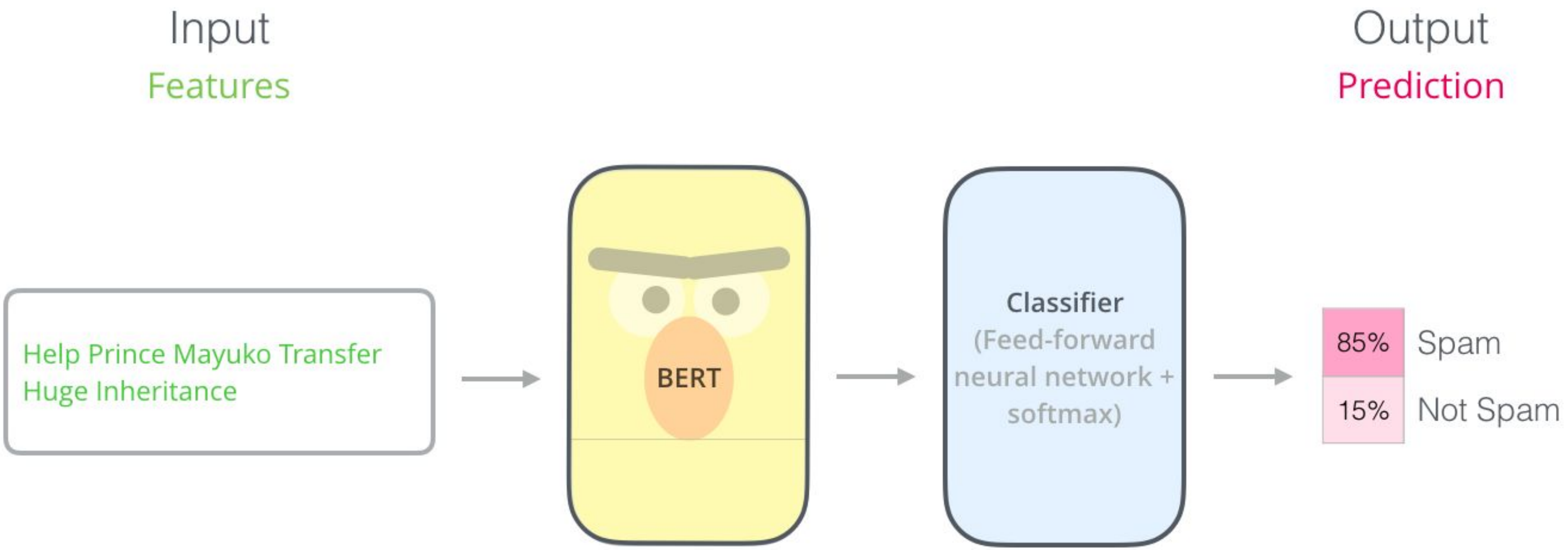


Image source: http://jalammar.github.io/illustrated-bert/

BERT: base and large

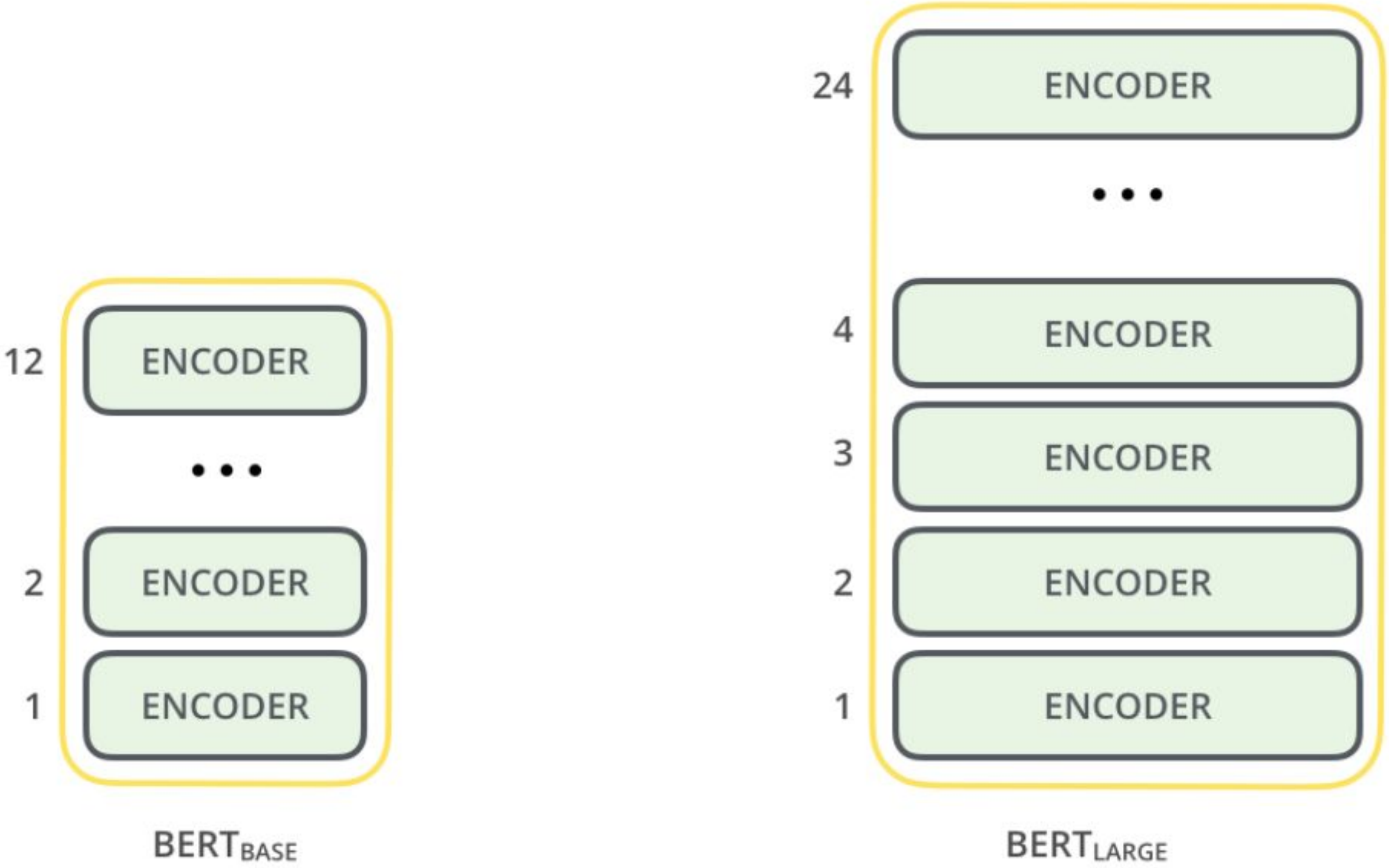


Image source: http://jalammar.github.io/illustrated-bert/

BERT vs. Transformer

			
		Base BERT	**Large BERT**
Encoders	6	12	24
Units in FFN	512	768	1024
Attention Heads	8	12	16

Image source: http://jalammar.github.io/illustrated-bert/

Model inputs

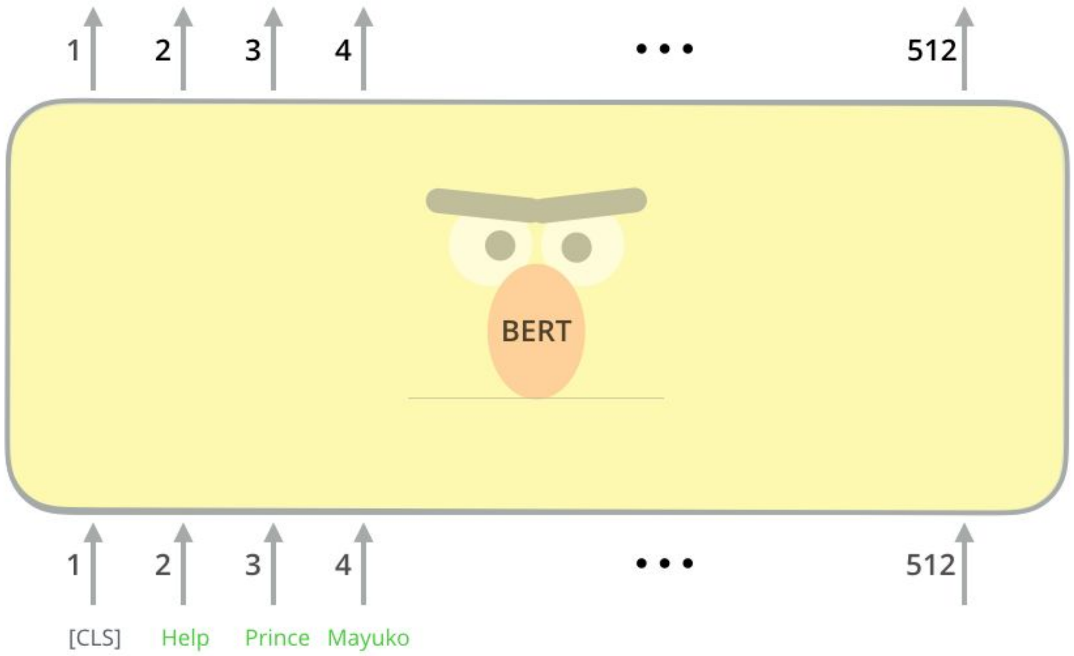
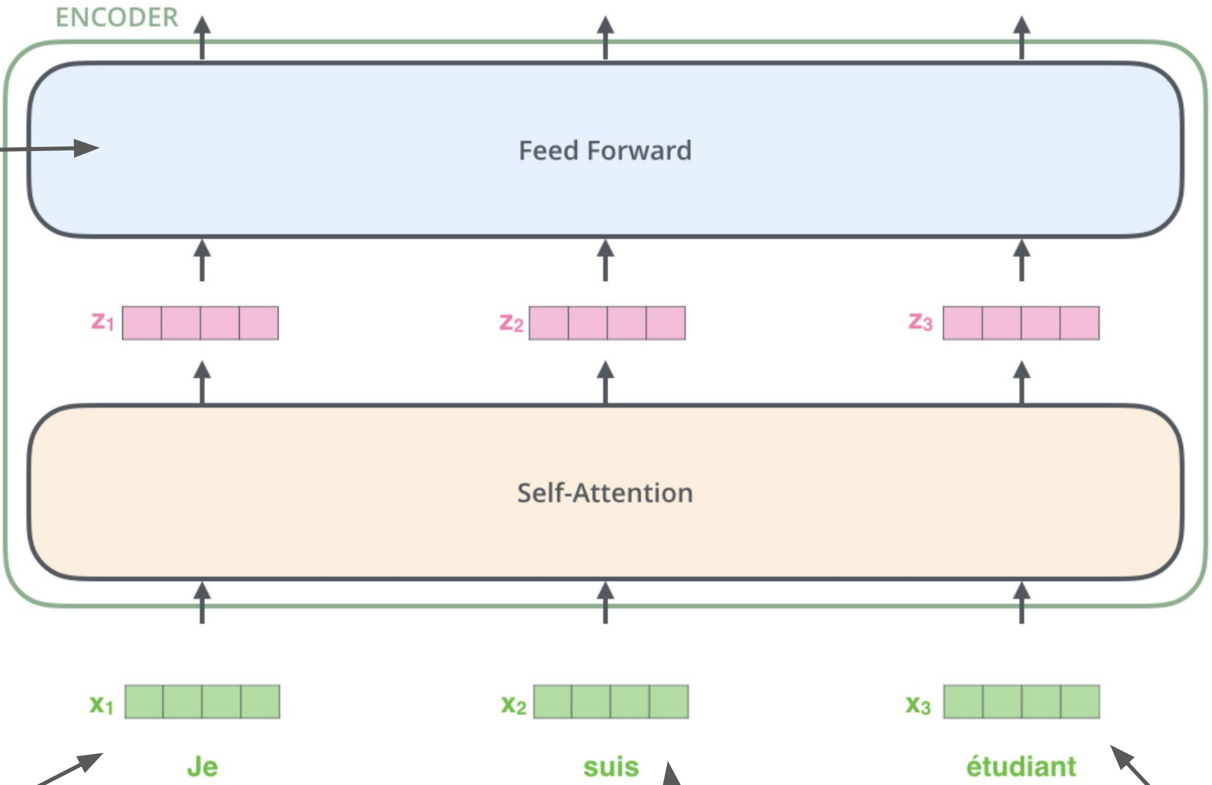


Image source: http://jalammar.github.io/illustrated-bert/

Transformer Block in BERT

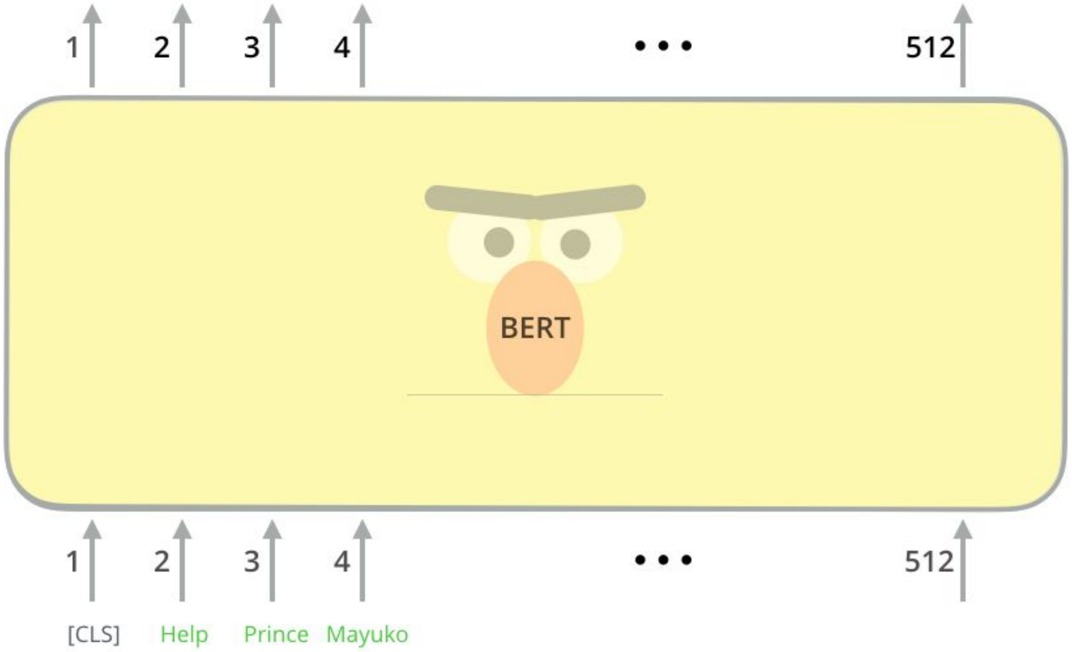
Can be parallelized



the word in each position flows through its own path in the encoder

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

Model inputs

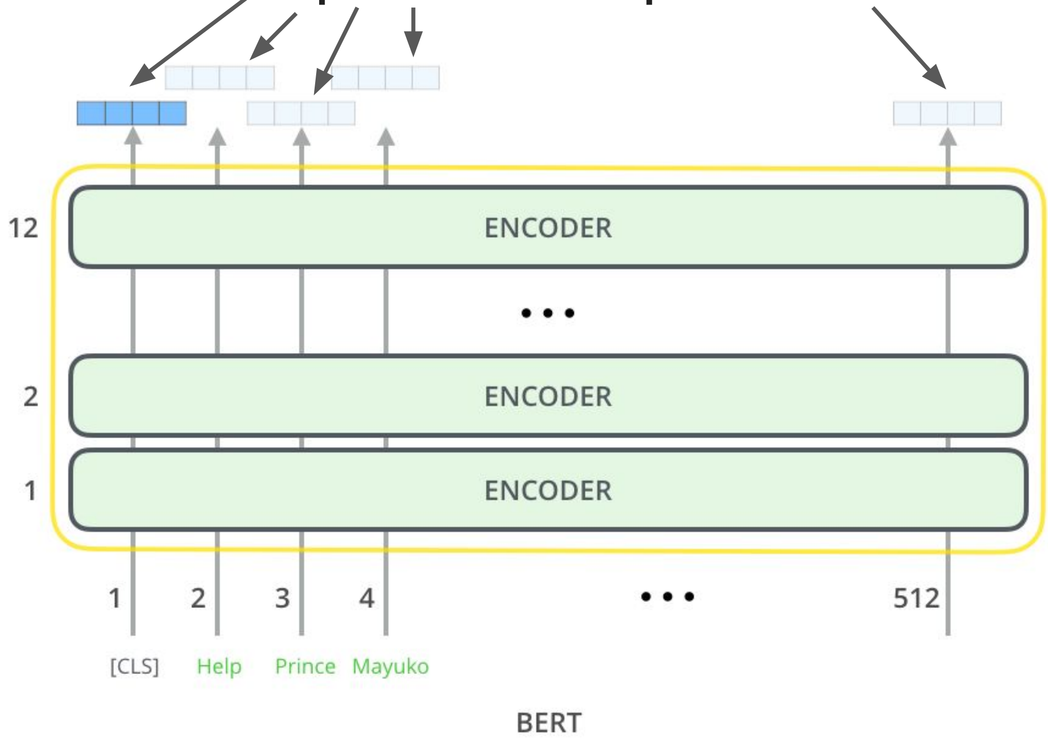


Identical to the Transformer up until this point

Why is BERT so special?

Model outputs

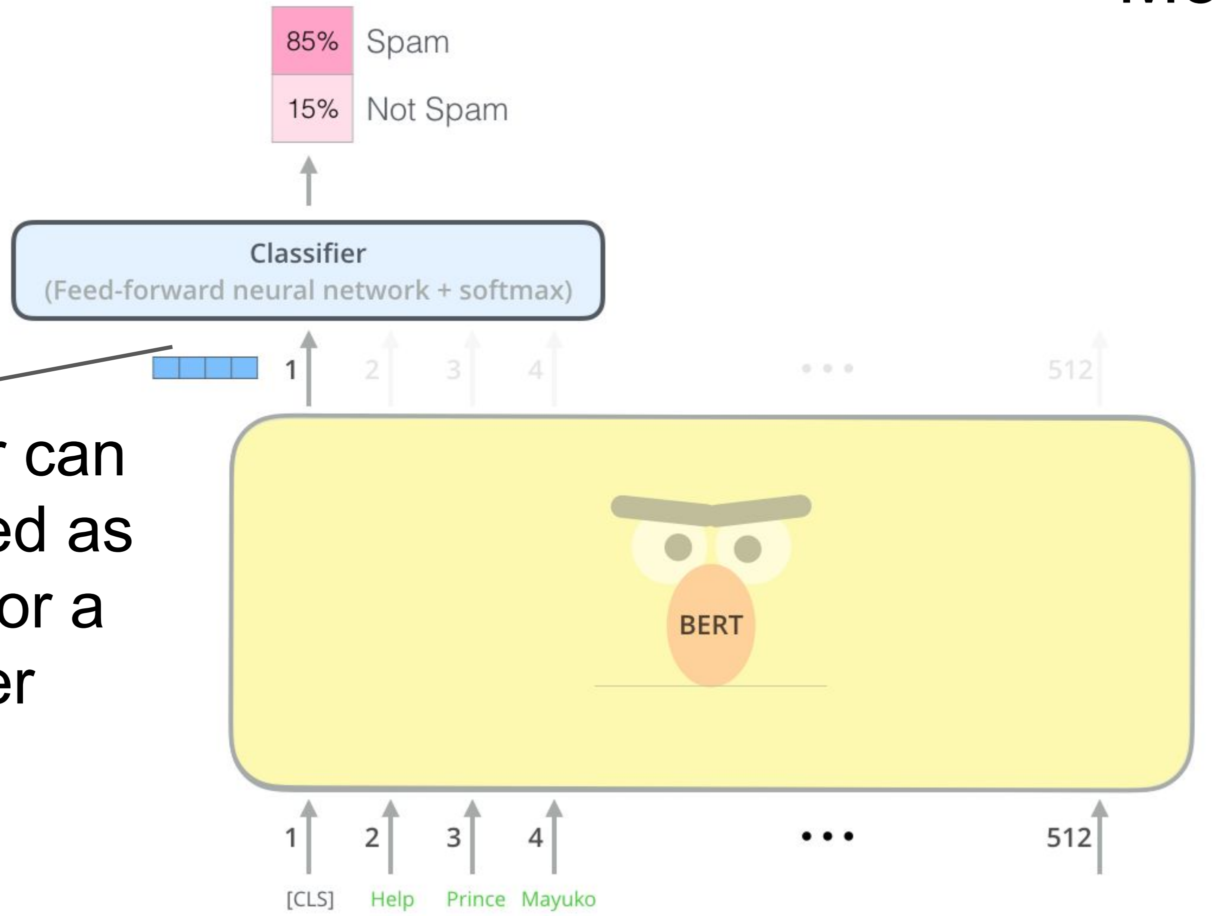
Each position outputs a vector



For sentence classification we focus on the first position (that we passed [CLS] token to)

Image source: http://jalammar.github.io/illustrated-bert/

Model inputs



This vector can now be used as the input for a classifier

Image source: http://jalammar.github.io/illustrated-bert/

Similar to CNN concept!

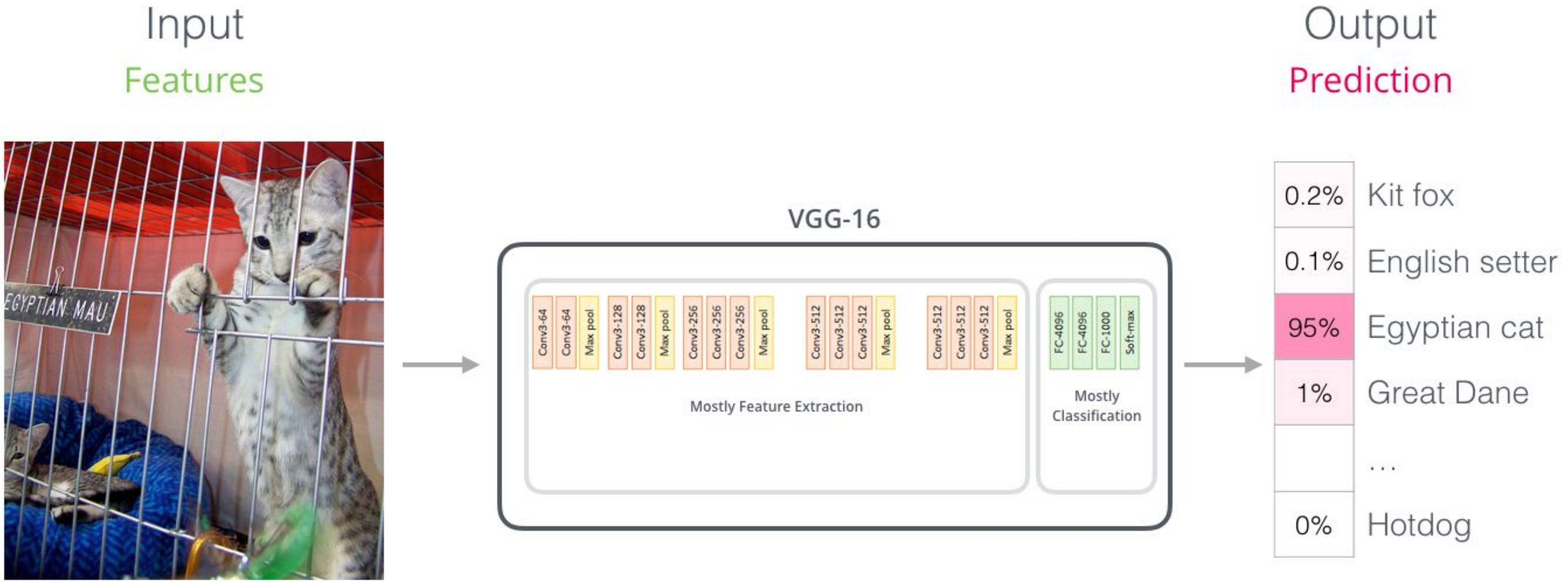


Image source: http://jalammar.github.io/illustrated-bert/

BERT: pre-training

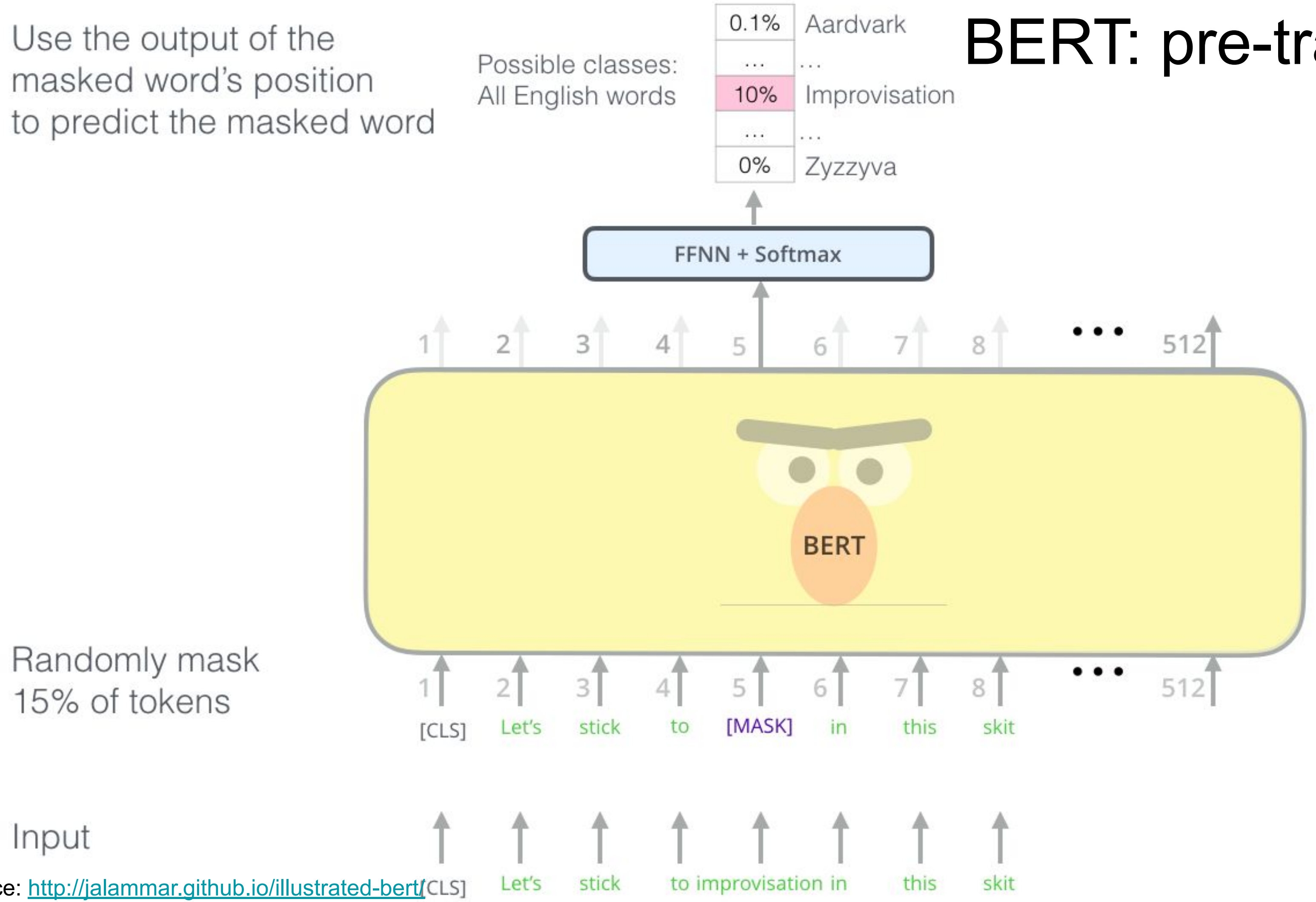


Image source: http://jalammar.github.io/illustrated-bert/

BERT: pre-training

- “Masked Language Model” approach

- To make BERT better at handling relationships between multiple sentences, the pre-training process includes an additional task:

  “Given two sentences (A and B), is B likely to be the sentence that follows A, or not?”

BERT: pre-training

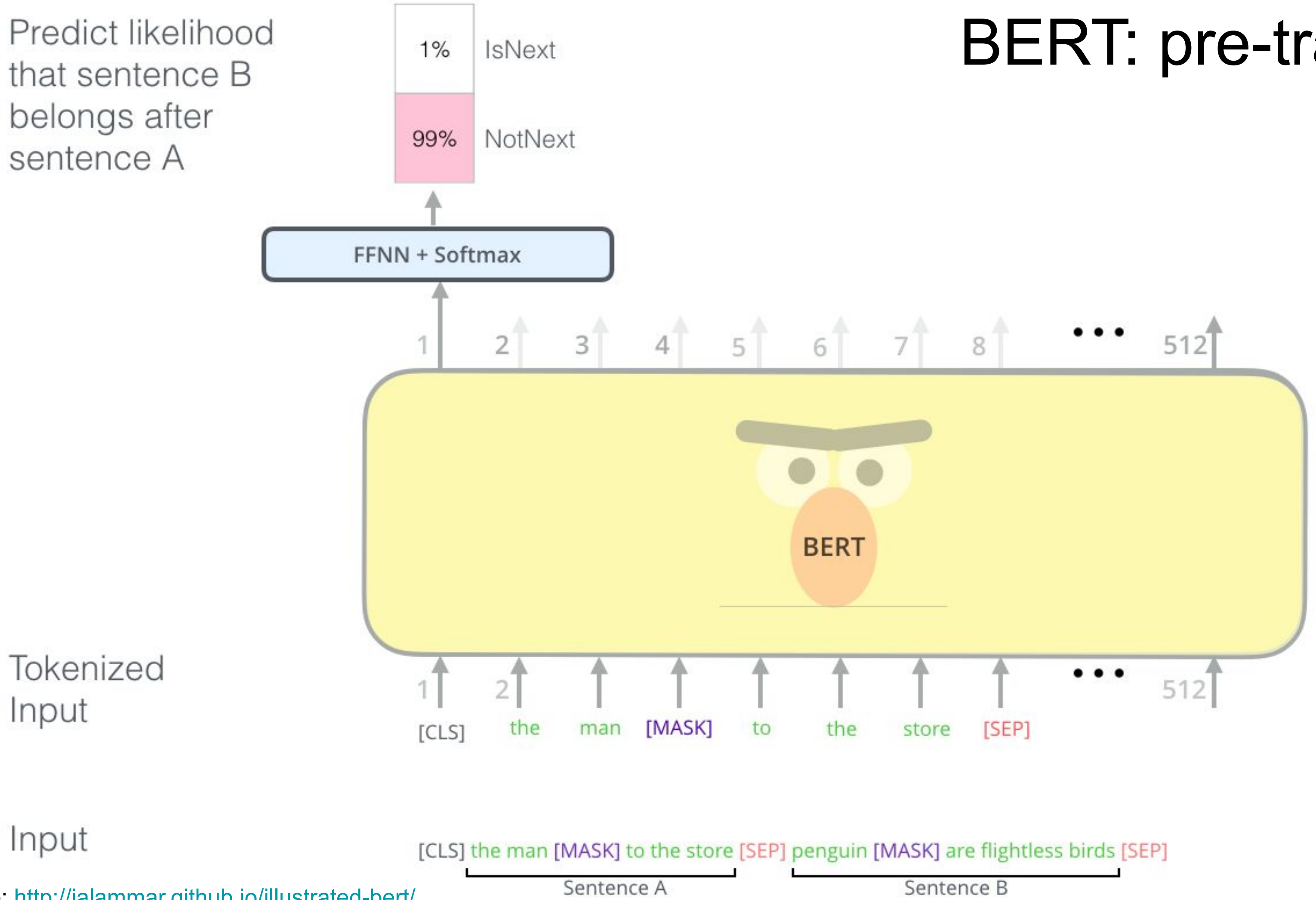


Image source: http://jalammar.github.io/illustrated-bert/

BERT: input data format

For each tokenized input sentence, we need to create:

- **input ids**: a sequence of integers identifying each input token to its index number in the BERT tokenizer vocabulary
- **segment mask**: a sequence of 1s and 0s used to identify whether the input is one sentence or two sentences long. For one sentence inputs, this is simply a sequence of 0s. For two sentence inputs, there is a 0 for each token of the first sentence, followed by a 1 for each token of the second sentence
- **attention mask**: a sequence of 1s and 0s, with 1s for all input tokens and 0s for all padding tokens

BERT: fine-tuning for different tasks

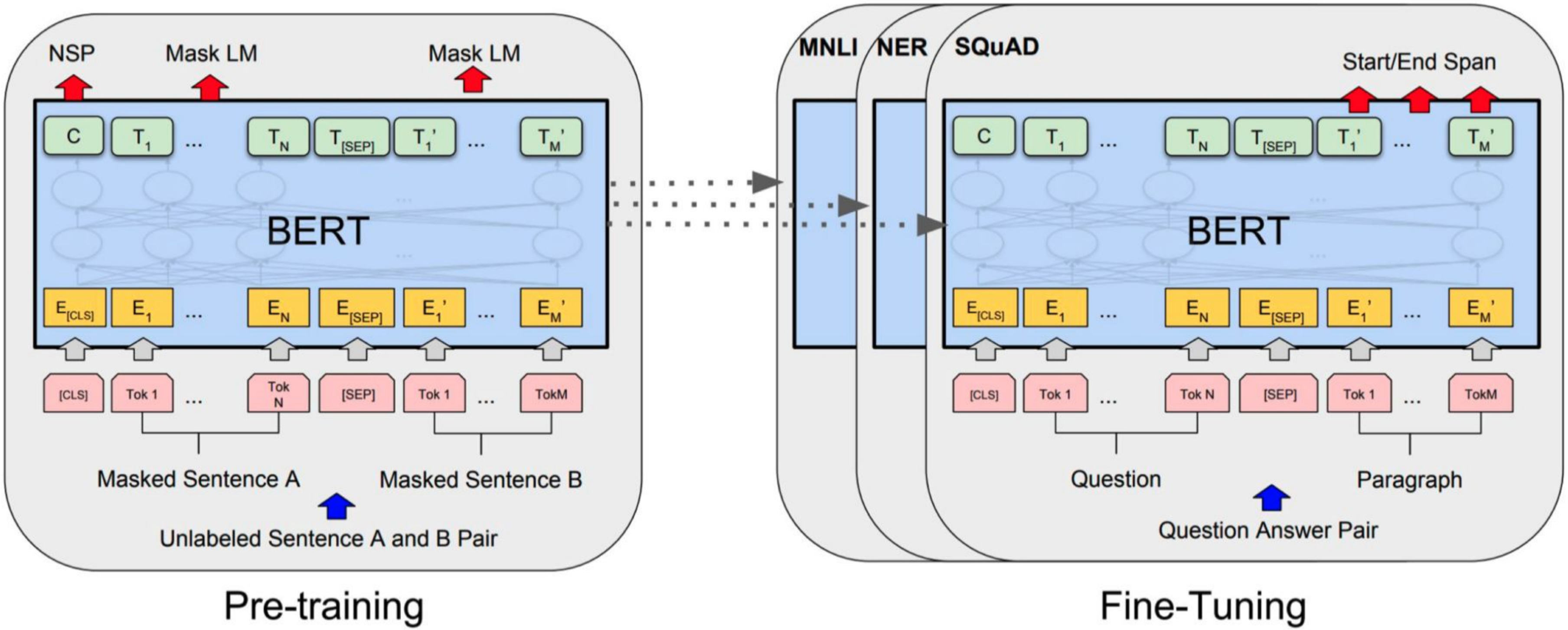
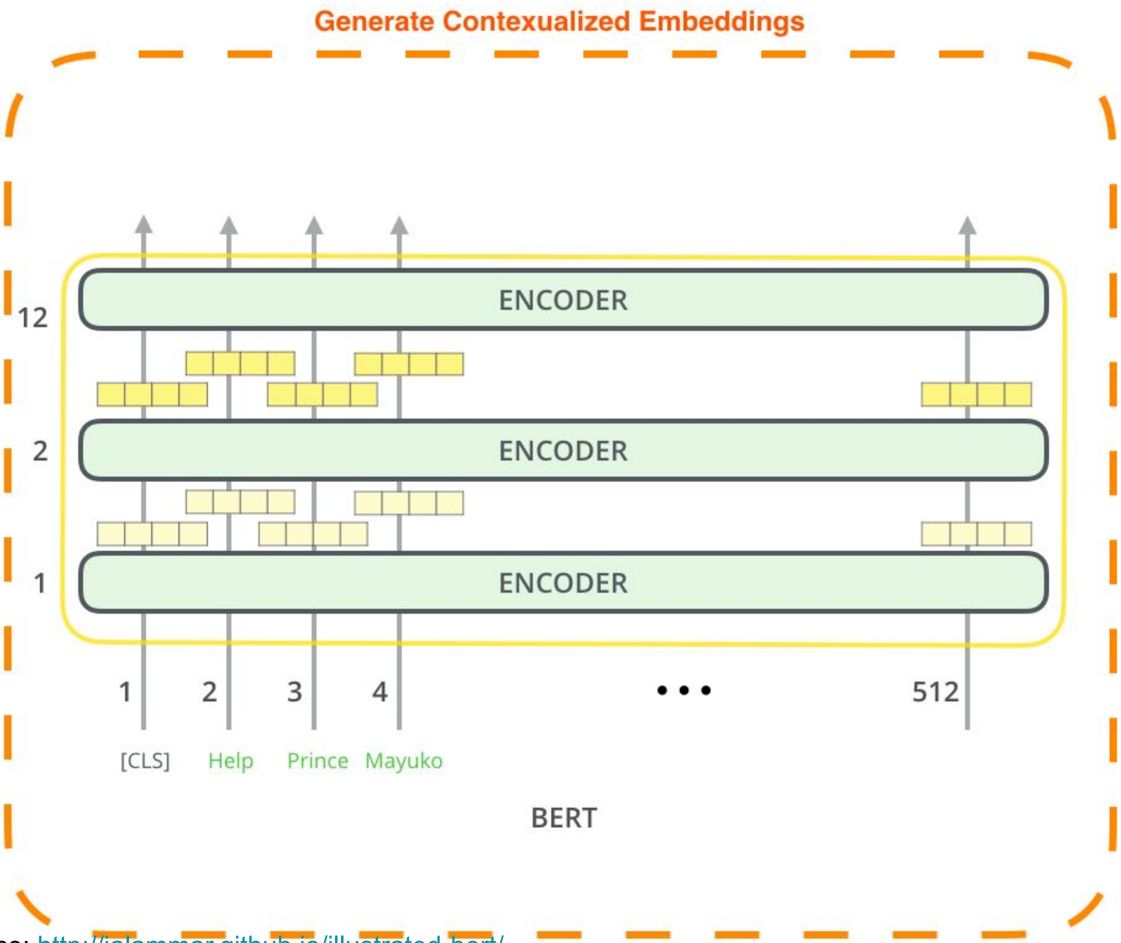
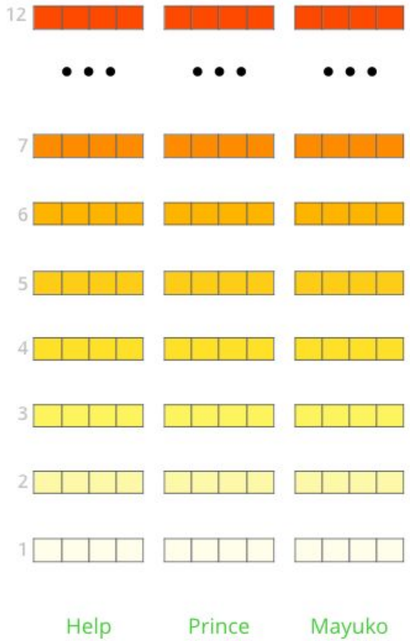


Image source: http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture13-contextual-representations.pdf

BERT for feature extraction



The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?

Image source: http://jalammar.github.io/illustrated-bert/

BERT for feature extraction

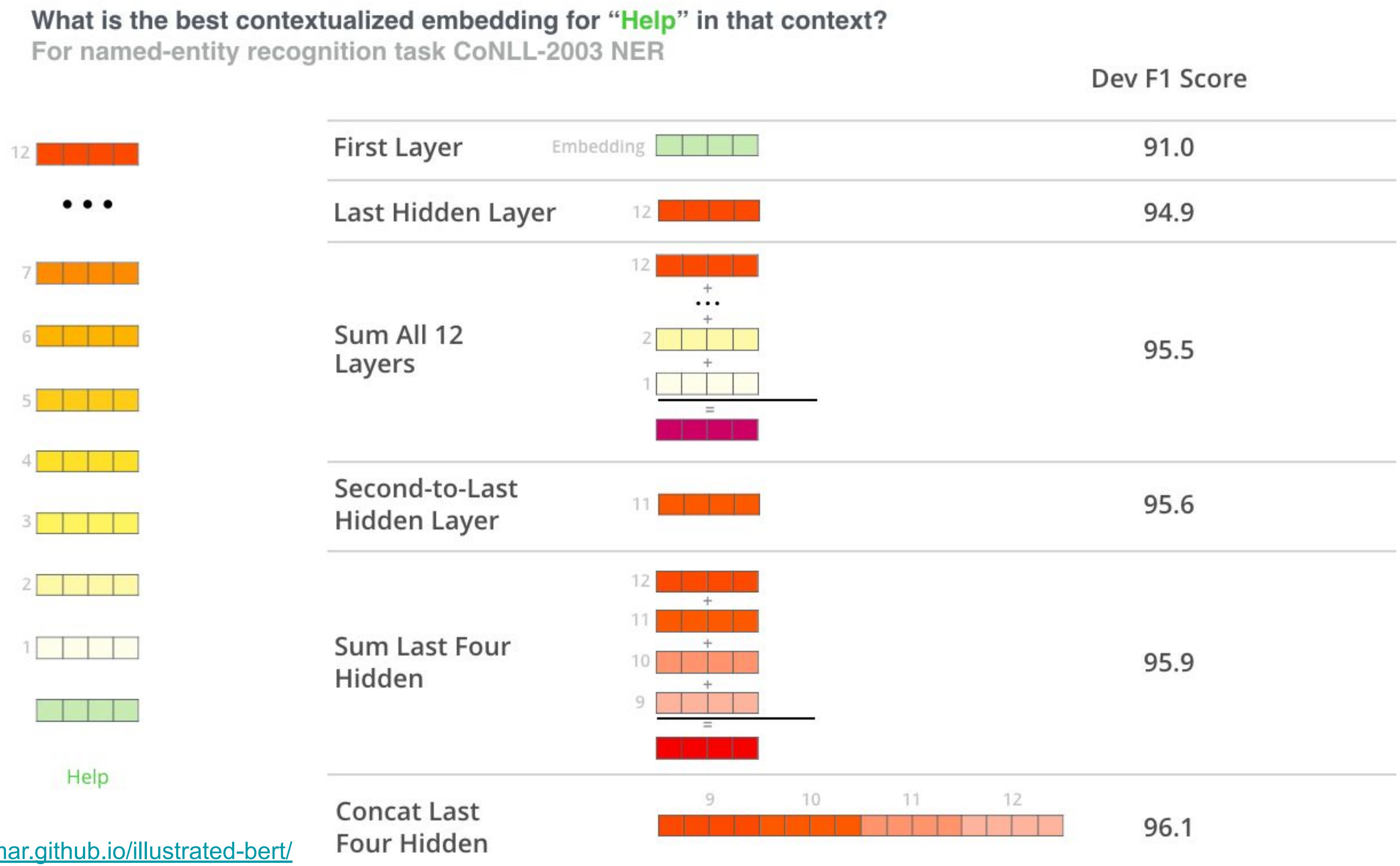


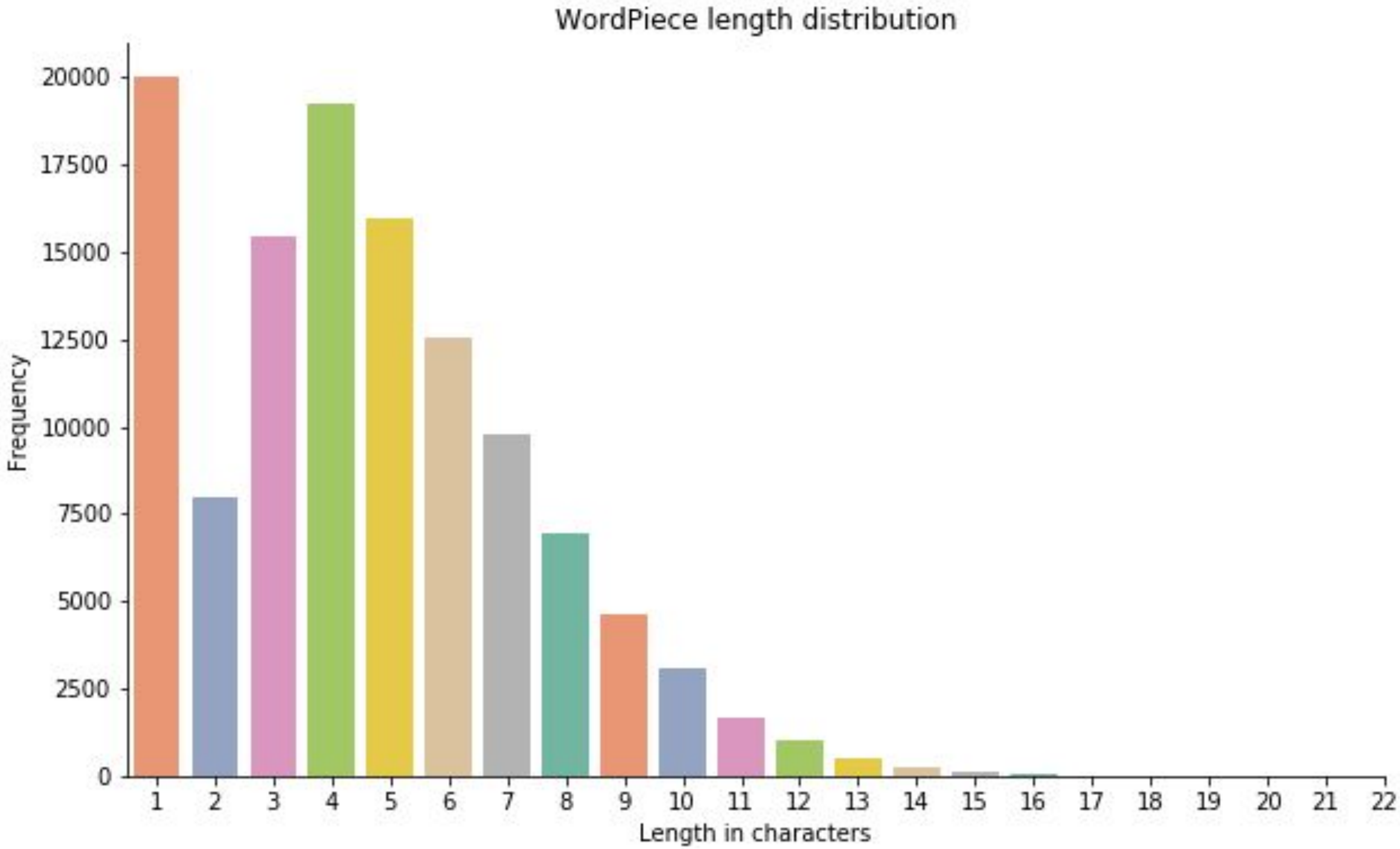
Image source: http://jalammar.github.io/illustrated-bert/

BERT: tokenization

<u>Example:</u> Unaffable -> un, ##aff, ##able

- Single model for 104 languages with a large shared vocabulary (119,547 WordPiece model)
- Non-word-initial units are prefixed with ##
- The first 106 symbols: constants like PAD and UNK
- 36.5% of the vocabulary are non-initial word pieces
- The alphabet consists of 9,997 unique characters that are defined as word-initial (C) and continuation symbols (##C), which together make up 19,994 word pieces
- The rest are multi character word pieces of various length.

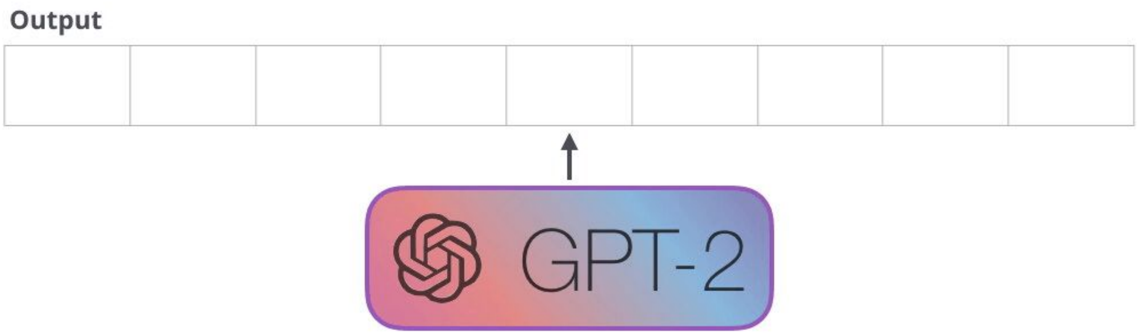
BERT: tokenization



GPT-2 & GPT-3

GPT-2

- Transformer-based architecture
- trained to predict the **next** word
- 1.5 billion parameters
- Trained on 8 million web-pages



On language tasks (question answering, reading comprehension, summarization, translation) works well **WITHOUT** fine-tuning

Image source: https://jalammar.github.io/illustrated-gpt2

GPT-2: question answering

EXAMPLES

Who wrote the book the origin of species?

Correct answer: *Charles Darwin*

Model answer: `Charles Darwin`

What is the largest state in the U.S. by land mass?

Correct answer: *Alaska*

Model answer: `California`

GPT-2: language modeling

EXAMPLE

Both its sun-speckled shade and the cool grass beneath were a welcome respite after the stifling kitchen, and I was glad to relax against the tree's rough, brittle bark and begin my breakfast of buttery, toasted bread and fresh fruit. Even the water was tasty, it was so clean and cold. It almost made up for the lack of...

Correct answer: *coffee*

Model answer: `food`

GPT-2: machine translation

EXAMPLE

French sentence:
Un homme a expliqué que l'opération gratuite qu'il avait subie pour soigner une hernie lui permettrait de travailler à nouveau.

Reference translation:
One man explained that the free hernia surgery he'd received will allow him to work again.

Model translation:
`A man told me that the operation gratuity he had been promised would not allow him to travel.`

GPT-2: fake news and hype

New AI fake text generator may be too dangerous to ... - The Guardian
https://www.theguardian.com/.../elon-musk-backed-ai-writes-convincing-news-fiction
4 days ago - The Elon Musk-backed nonprofit company OpenAI declines to release research publicly for fear of misuse. The creators of a revolutionary AI system that can write news stories and works of fiction – dubbed “deepfakes for text” – have taken the unusual step of not releasing ...

OpenAI built a text generator so good, it's considered too dangerous to ...
https://techcrunch.com/2019/02/17/openai-text-generator-dangerous/ ▾
12 hours ago - A storm is brewing over a new language model, built by non-profit artificial intelligence research company OpenAI, which it says is so good at ...

The AI Text Generator That's Too Dangerous to Make Public | WIRED
https://www.wired.com/story/ai-text-generator-too-dangerous-to-make-public/ ▾
4 days ago - In 2015, car-and-rocket man Elon Musk joined with influential startup backer Sam Altman to put artificial intelligence on a new, more open ...

Elon Musk-backed AI Company Claims It Made a Text Generator ...
https://gizmodo.com/elon-musk-backed-ai-company-claims-it-made-a-text-gener-183... ▾
Elon Musk-backed AI Company Claims It Made a Text Generator That's **Too Dangerous to** Release · Rhett Jones · Friday 12:15pm · Filed to: OpenAI Filed to: ...

Scientists have made an AI that they think is too dangerous to ...
https://www.weforum.org/.../amazing-new-ai-churns-out-coherent-paragraphs-of-text/ ▾
3 days ago - Sample outputs suggest that the AI system is an extraordinary step forward, producing text rich with context, nuance and even something ...

New AI Fake Text Generator May Be Too Dangerous To ... - Slashdot
https://news.slashdot.org/.../new-ai-fake-text-generator-may-be-too-dangerous-to-rele... ▾
3 days ago - An anonymous reader shares a report: The creators of a revolutionary AI system that can write news stories and works of fiction -- dubbed ...

Top stories

OpenAI built a text generator so good, it’s considered too dangerous to release
TechCrunch
11 hours ago

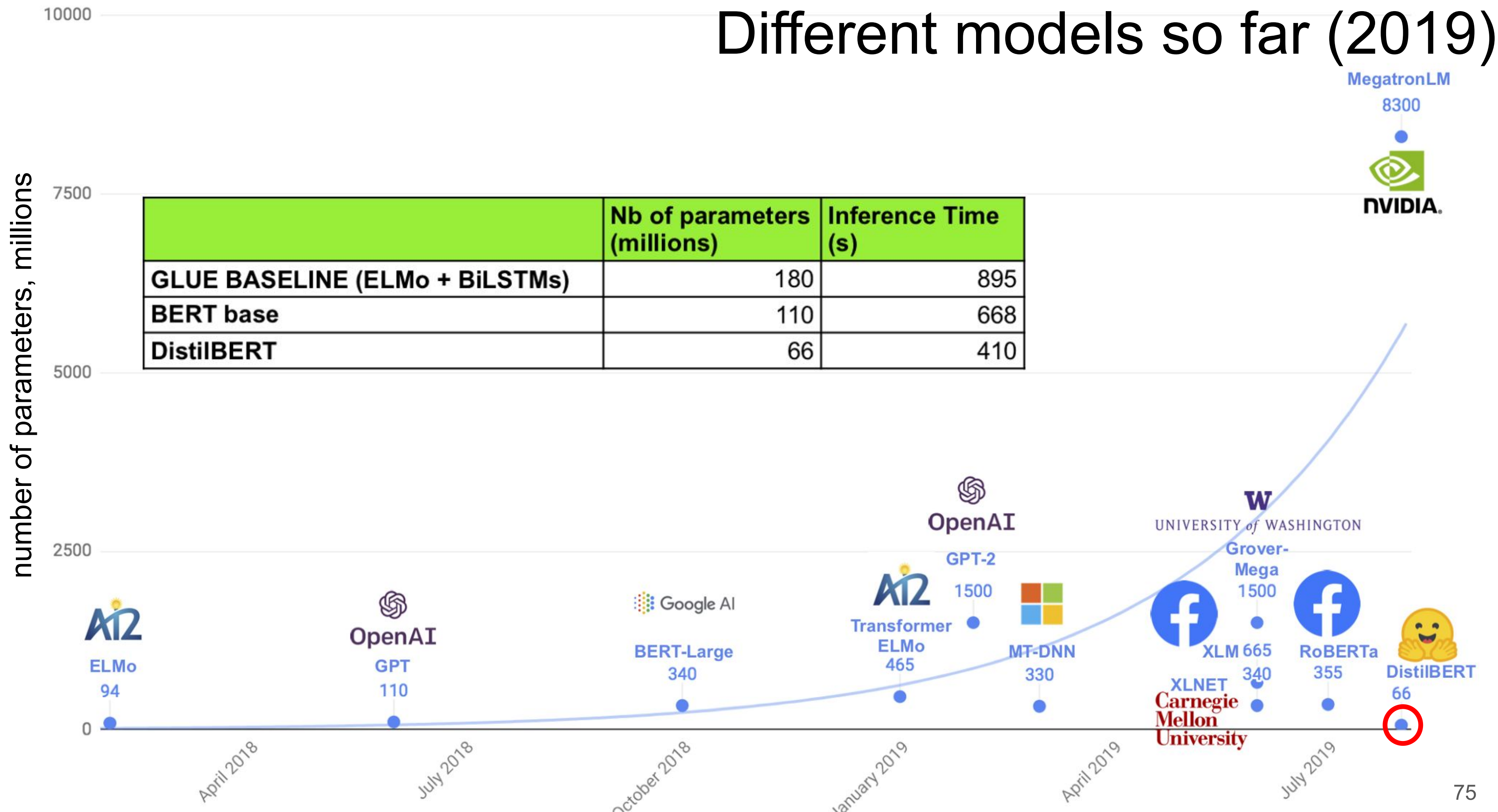
Elon Musk’s AI company created a fake news generator it’s too scared to make public
BGR.com
9 hours ago

The AI That Can Write A Fake News Story From A Handful Of Words
NDTV.com
2 hours ago

When Is Technology Too Dangerous to Release to the Public?
Slate · 2 days ago

Scientists Developed an AI So Advanced They Say It's Too Dangerous to Release
ScienceAlert · 6 days ago

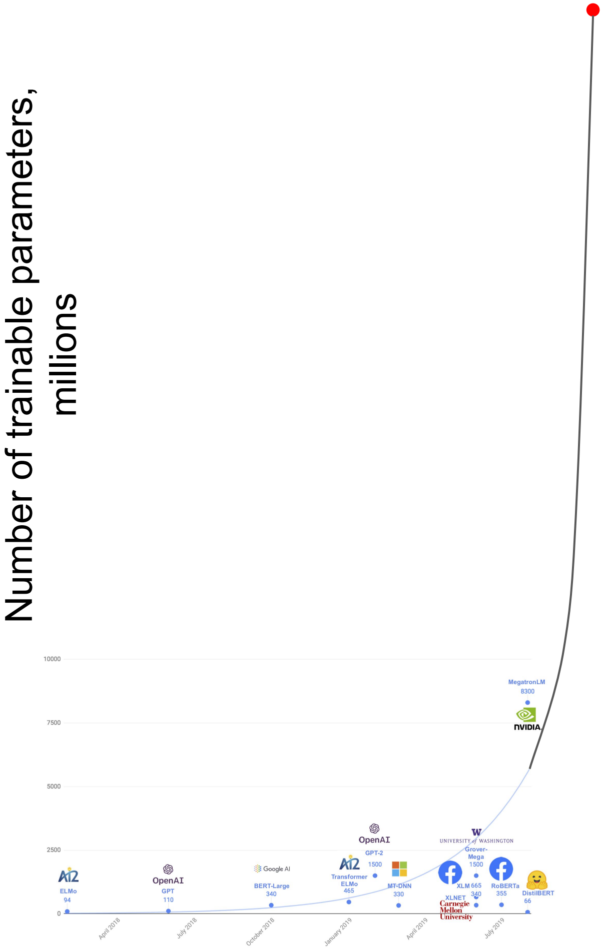
Different models so far (2019)



	Nb of parameters (millions)	Inference Time (s)
GLUE BASELINE (ELMo + BiLSTMs)	180	895
BERT base	110	668
DistilBERT	66	410

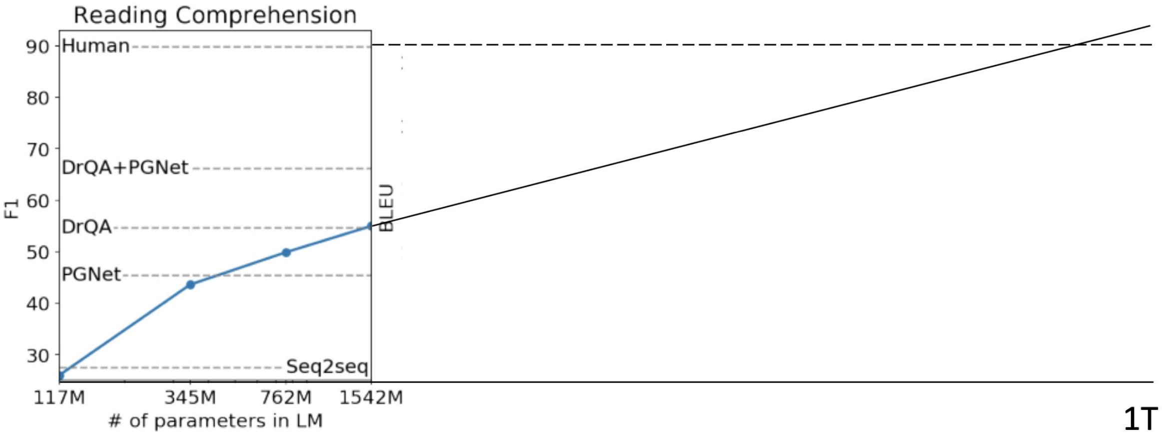
Image source: [Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT]

Latest achievements: GPT-3



GPT-3, May 2020

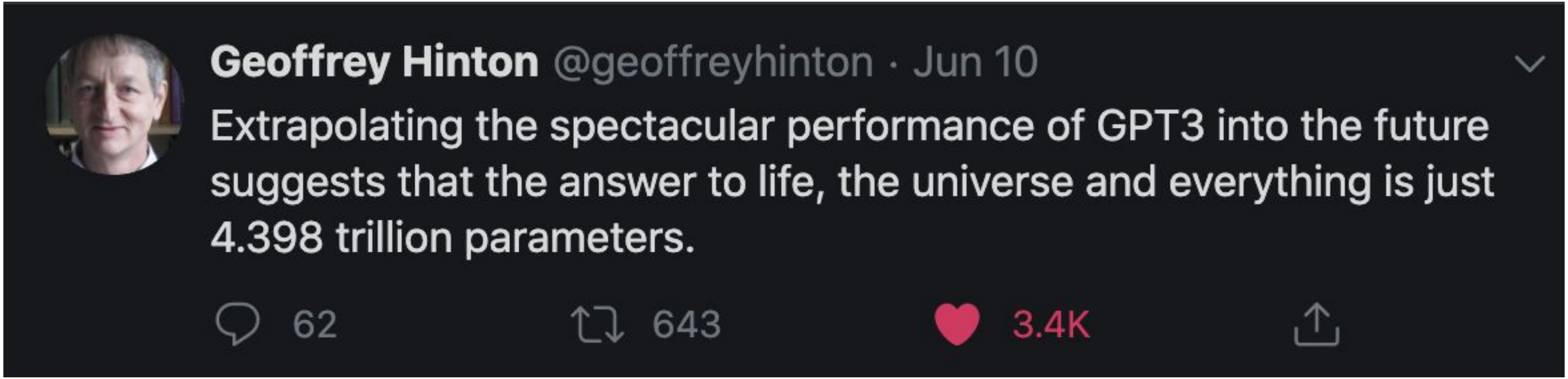
Proportions are not preserved for visual sake



Hypothesis from Stanford CS224N Lecture 20 (2019)

May 2020: GPT-3

- GPT-2: 1.5 billion parameters
- GPT-3: **175 billion** parameters



- [Transformer]
- [OpenAI Transformer]
- [ELMO]
- [BERT]
- [BERTology]
- [GPT]
- [GPT-2]
- [GPT-3]

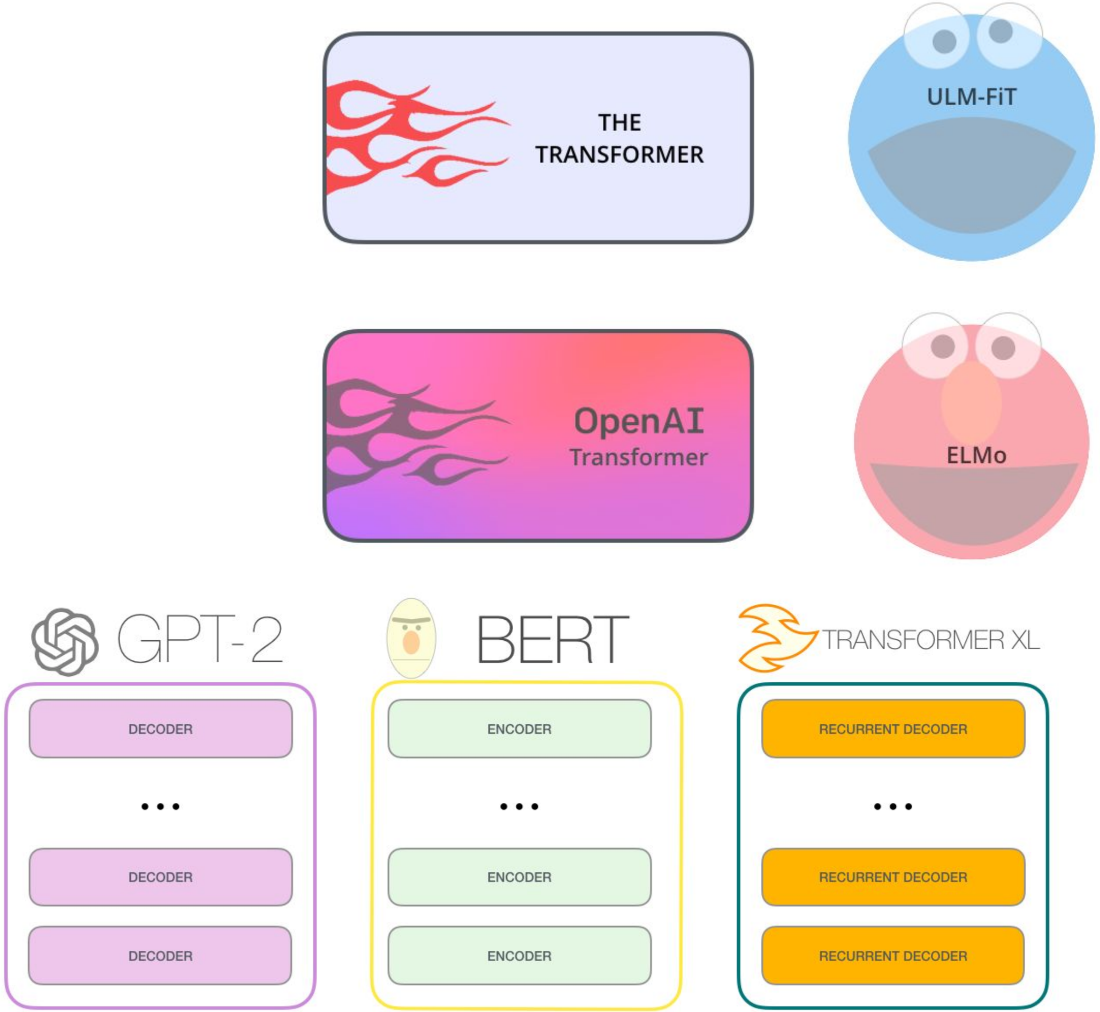


Image sources: http://jalammar.github.io/illustrated-bert/
http://jalammar.github.io/illustrated-gpt2/

Outro

- Transformer is novel and very powerful architecture
- It is worth it to understand how Self-Attention works
- BERT is variant of Decoders from Transformer for variety of tasks
- GPT are even bigger and better in metrics but they are made by corporations