Artificial Intelligence for Text Analytics

Natural Language Processing with Transformers

1121AITA04
MBA, IM, NTPU (M5265) (Fall 2023)
Tue 2, 3, 4 (9:10-12:00) (B3F17)

Min-Yuh Day, Ph.D, Associate Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday

https://meet.google.com/miy-fbif-max

2023-10-04

Syllabus

Week	Date	Subject/Topics
1	2023/09/13	Introduction to Artificial Intelligence for Text Analytics
2	2023/09/20	Foundations of Text Analytics: Natural Language Processing (NLP)
3	2023/09/27	Python for Natural Language Processing
4	2023/10/04	Natural Language Processing with Transformers
5	2023/10/11	Case Study on Artificial Intelligence for Text Analytics I
6	2023/10/18	Text Classification and Sentiment Analysis

Syllabus

Week	Date	Subject/Topics
7	2023/10/25	Multilingual Named Entity Recognition (NER)
8	2023/11/01	Midterm Project Report
9	2023/11/08	Text Similarity and Clustering
10	2023/11/15	Text Summarization and Topic Models
11	2023/11/22	Text Generation with Large Language Models (LLMs)
12	2023/11/29	Case Study on Artificial Intelligence for Text Analytics II

Syllabus

Week	Date	Subject/Topics
13	2023/12/06	Question Answering and Dialogue Systems
14	2023/12/13	Deep Learning, Generative AI, Transfer Learning, Zero-Shot, and Few-Shot Learning for Text Analytics
15	2023/12/20	Final Project Report I
16	2023/12/27	Final Project Report II

Natural Language Processing with Transformers

Outline

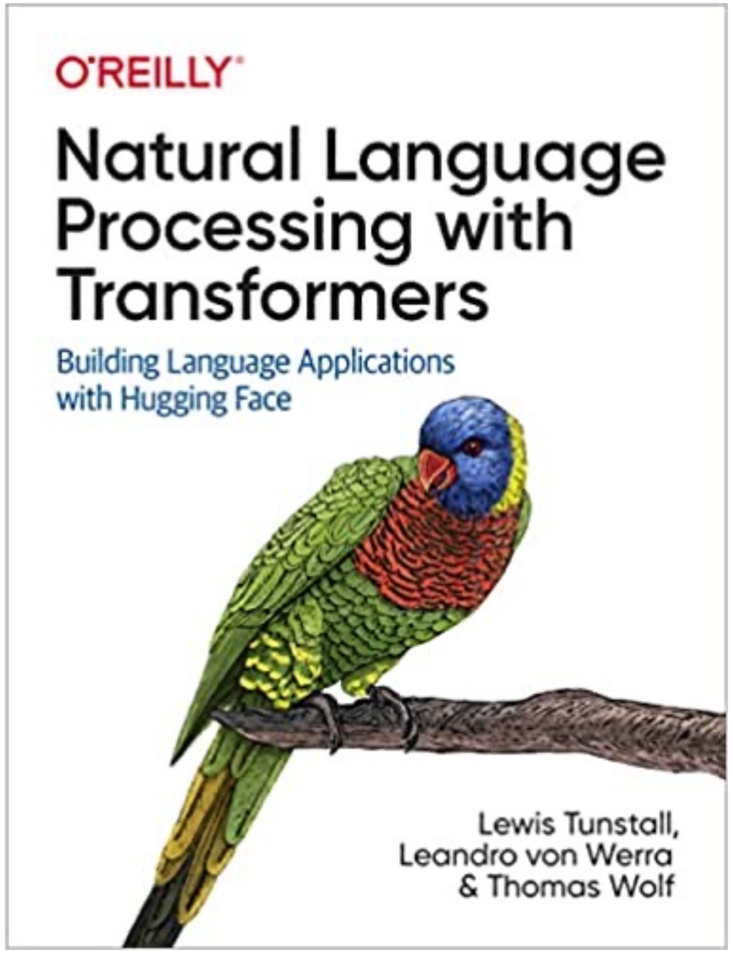
- **Natural Language Processing with Transformers**
  - **Transformer (Attention is All You Need)**
    - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
    - ChatGPT: Large Language Models (LLMs), Foundation Models
  - **Encoder-Decoder**
  - **Attention Mechanisms**
  - **Transfer Learning in NLP: Pre-train, Fine-tune**

Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022),

Natural Language Processing with Transformers:

Building Language Applications with Hugging Face,

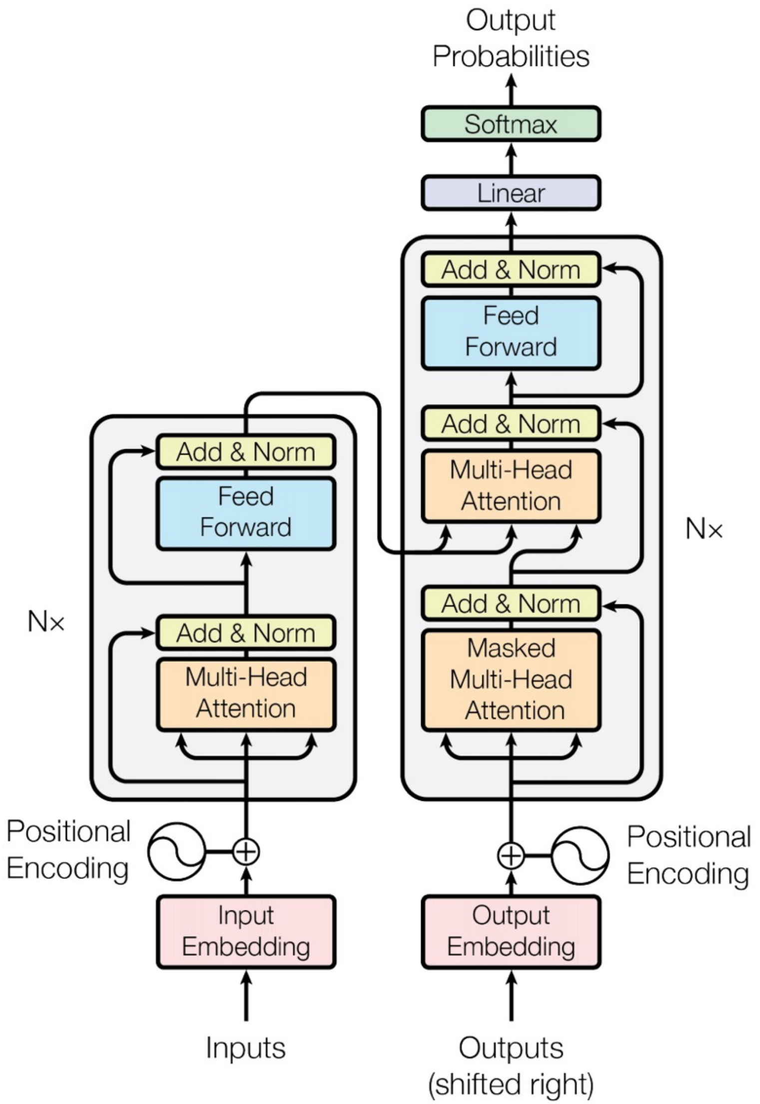
O'Reilly Media.



Source: https://www.amazon.com/Natural-Language-Processing-Transformers-Applications/dp/1098103246

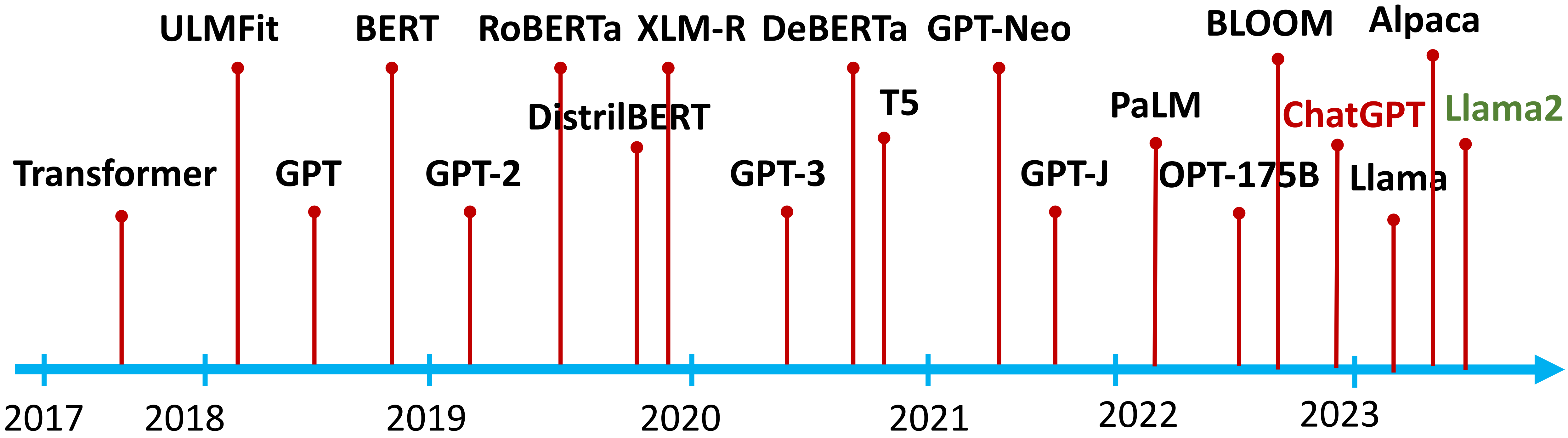
Transformer (Attention is All You Need)

(Vaswani et al., 2017)



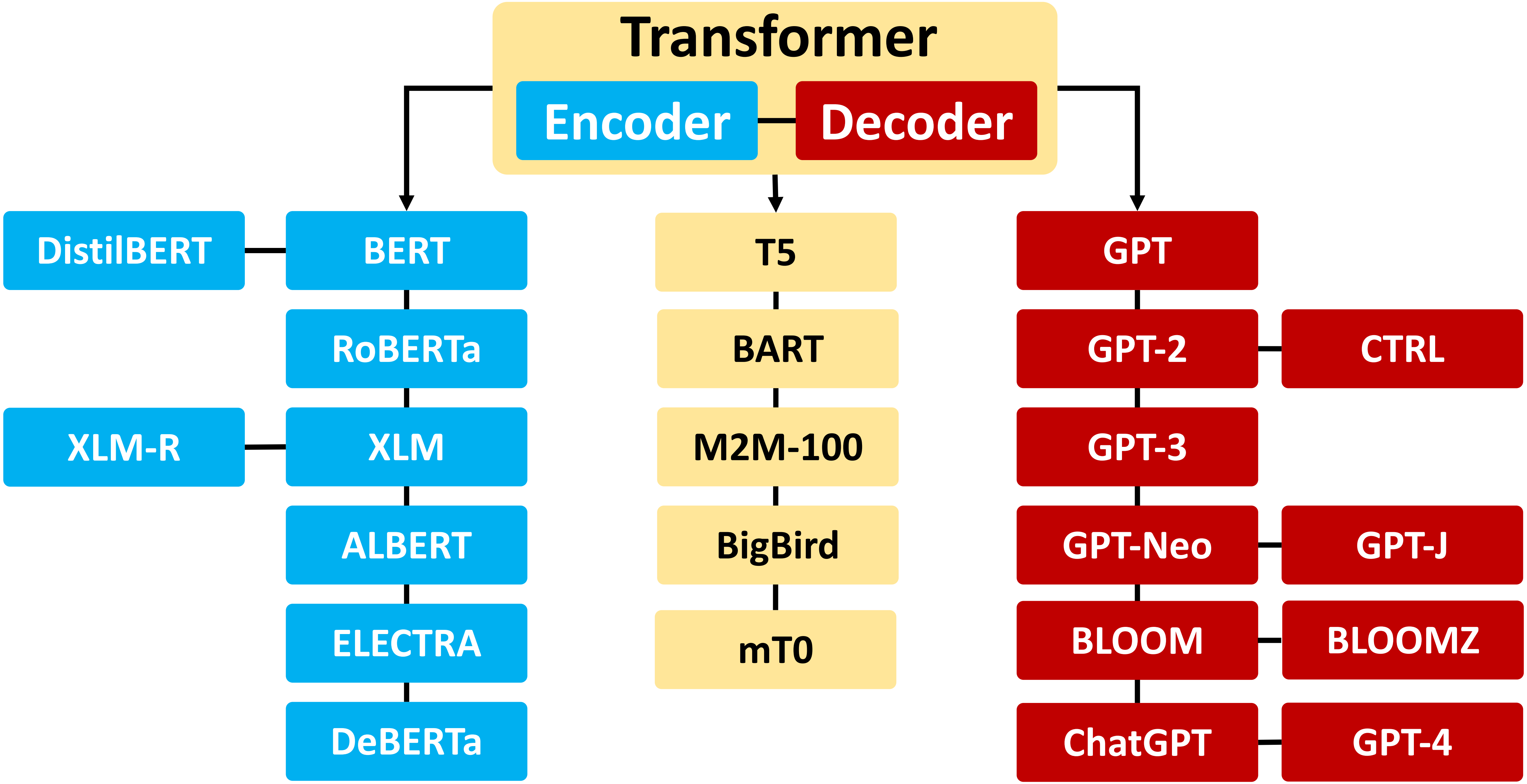
Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

The Transformers Timeline



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Transformer Models

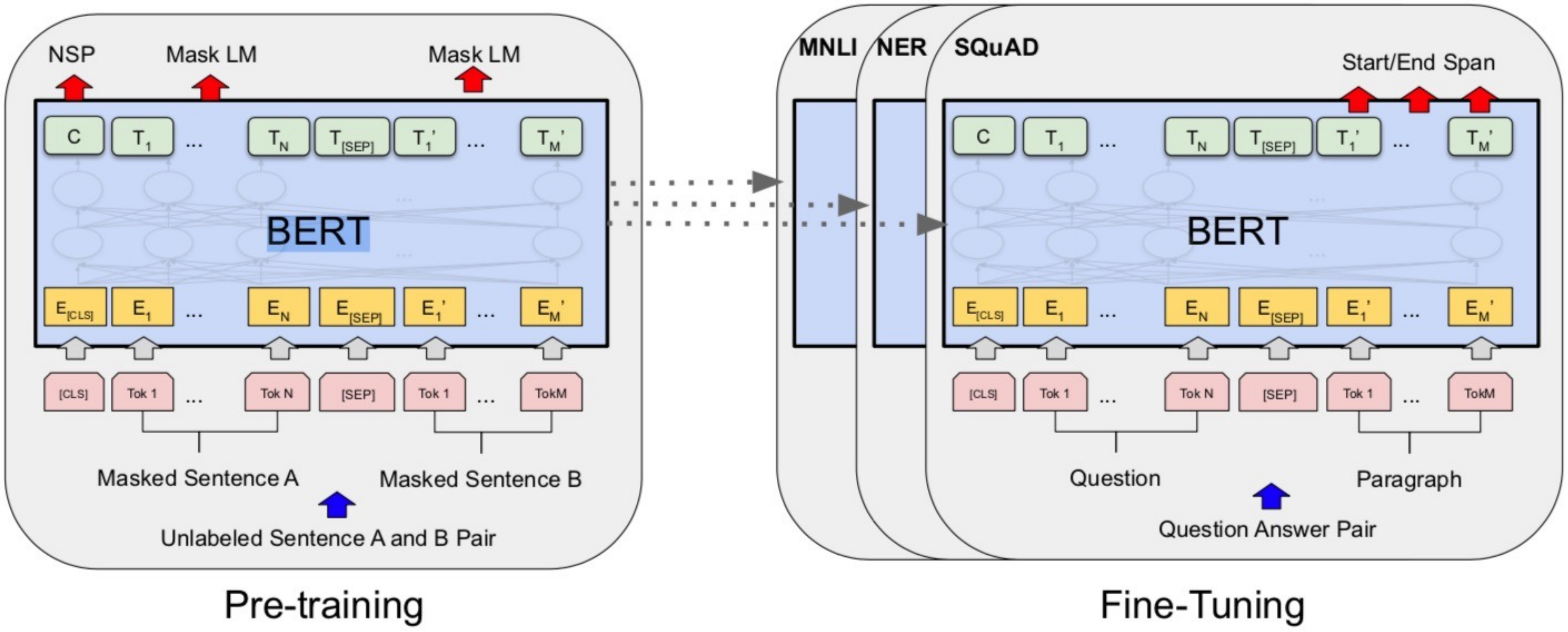


Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT



Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

BERT:
Pre-training of Deep Bidirectional Transformers for Language Understanding

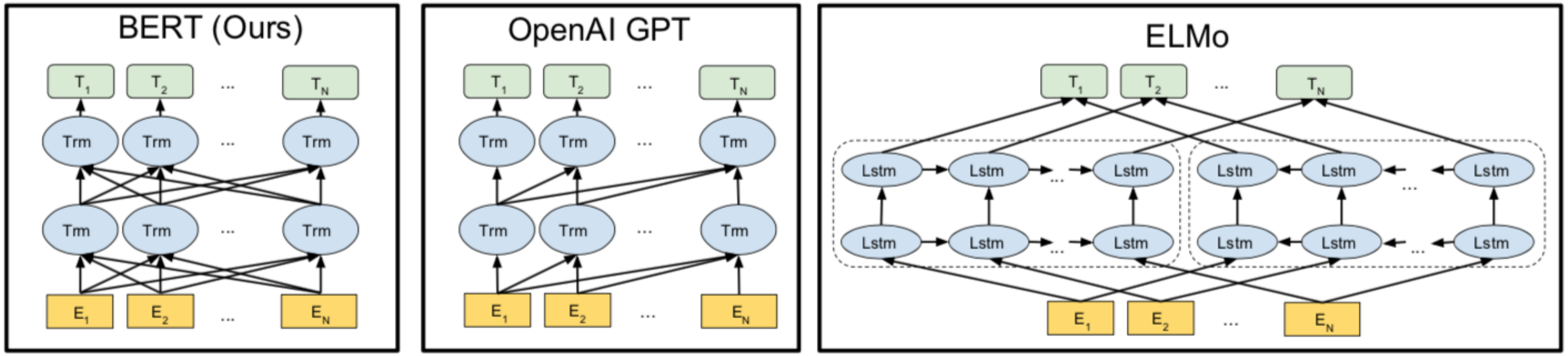
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT

Bidirectional Encoder Representations from Transformers



Pre-training model architectures

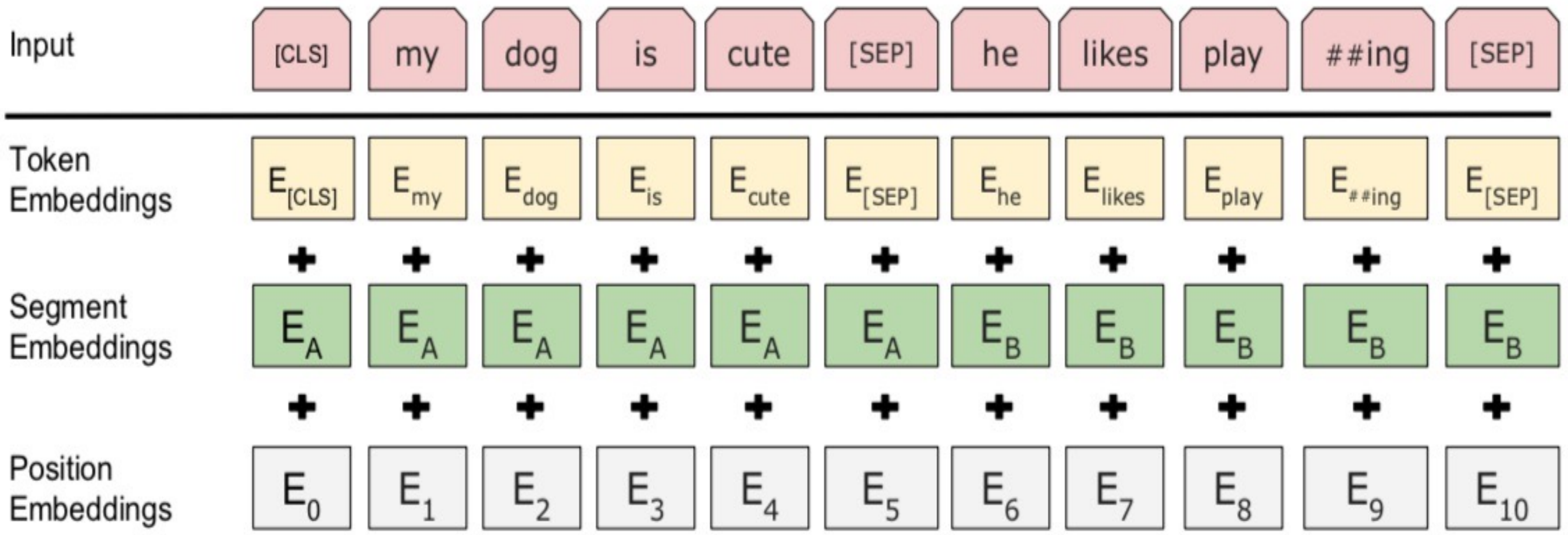
BERT uses a bidirectional Transformer.
OpenAI GPT uses a left-to-right Transformer.
ELMo uses the concatenation of independently trained left-to-right and right- to-left LSTM to generate features for downstream tasks.
Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

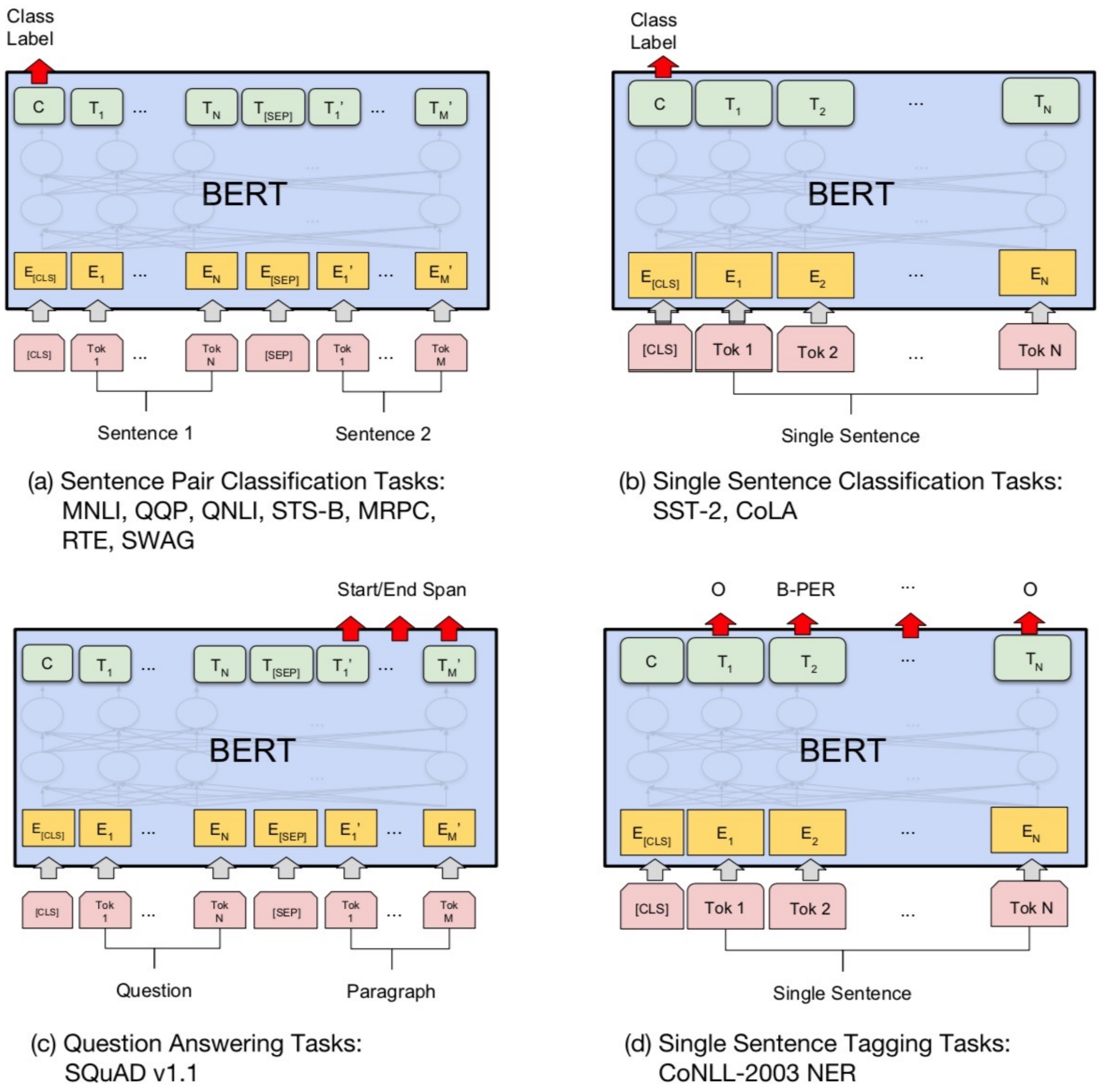
BERT input representation



The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

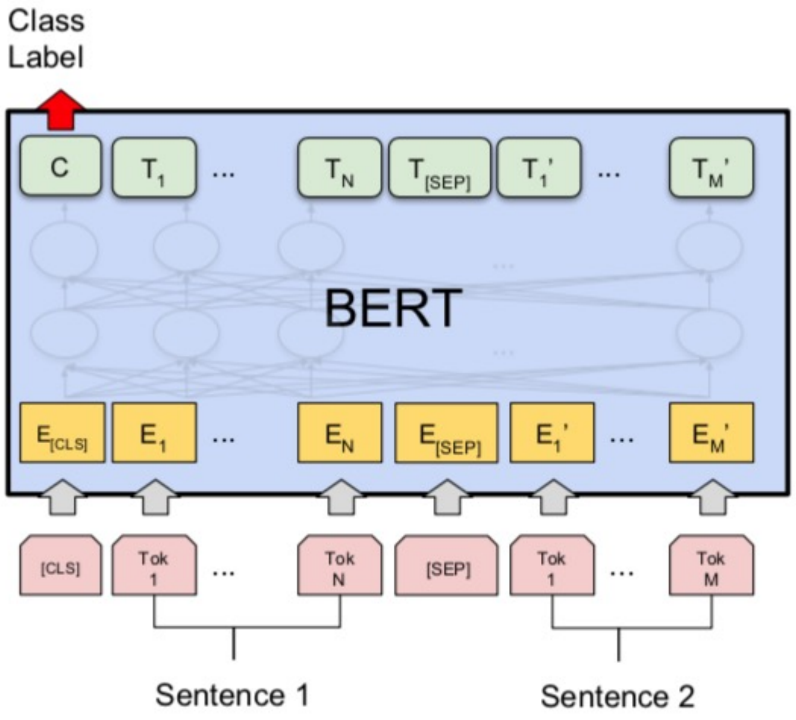
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Fine-tuning BERT on NLP Tasks

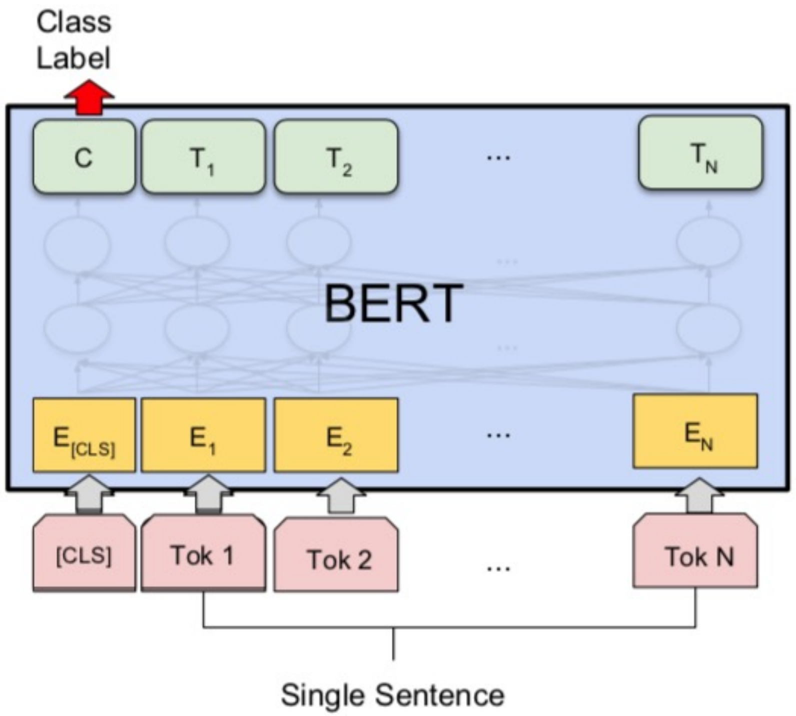


Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT Sequence-level tasks



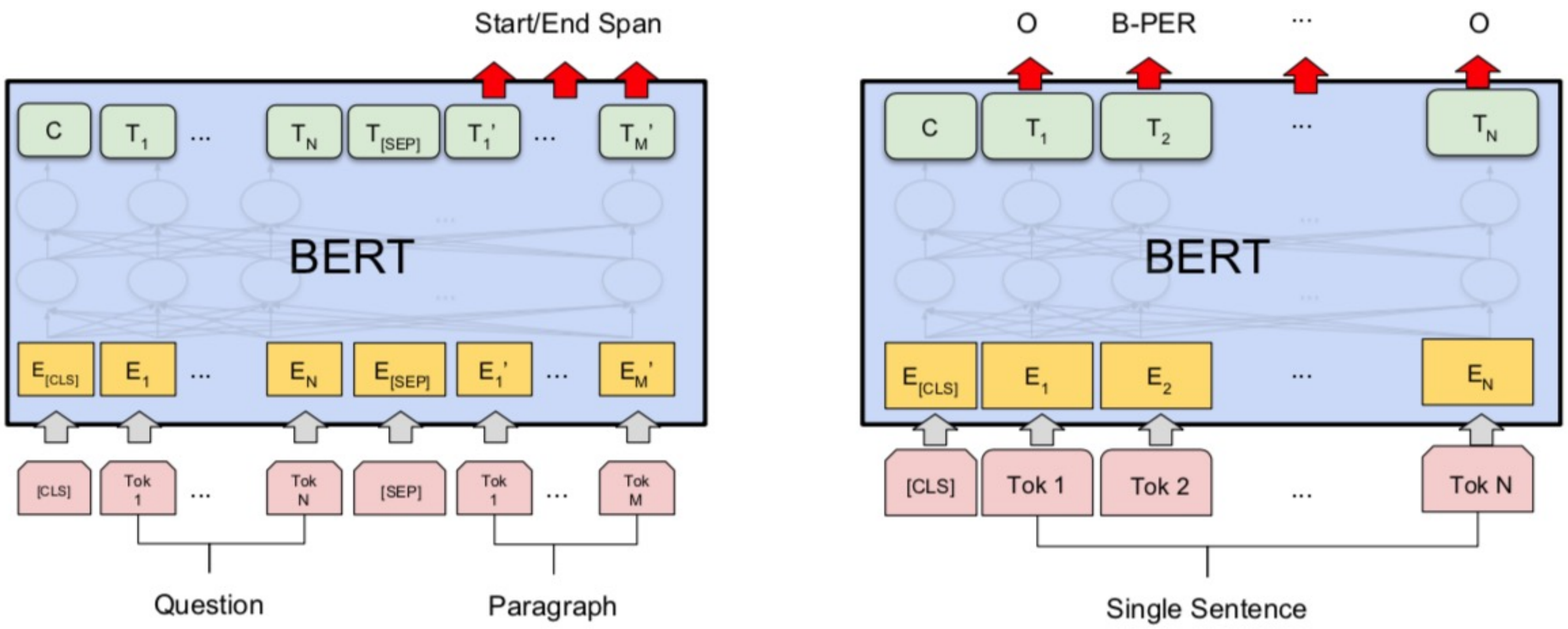
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT Token-level tasks

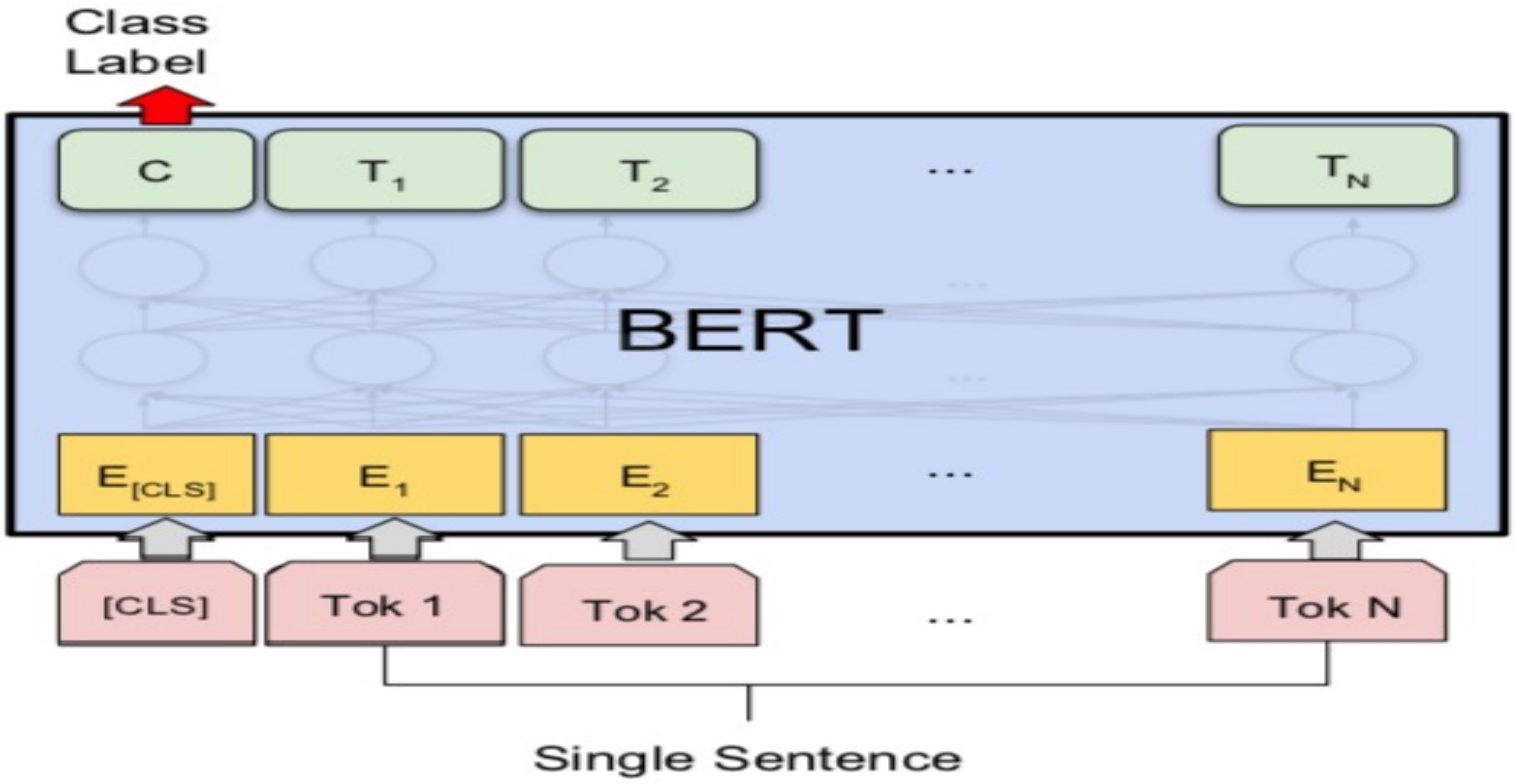


(c) Question Answering Tasks:
SQuAD v1.1

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

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

Sentiment Analysis: Single Sentence Classification



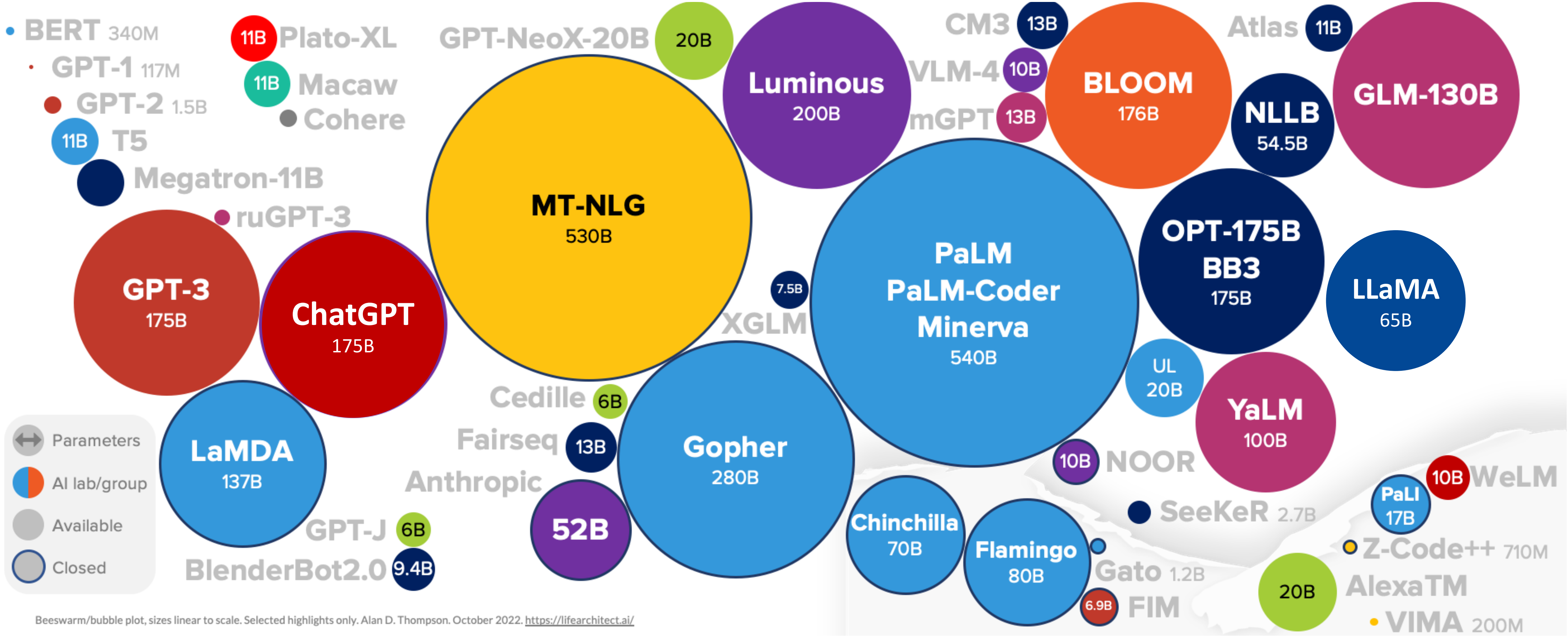
(b) Single Sentence Classification Tasks: SST-2, CoLA

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

ChatGPT

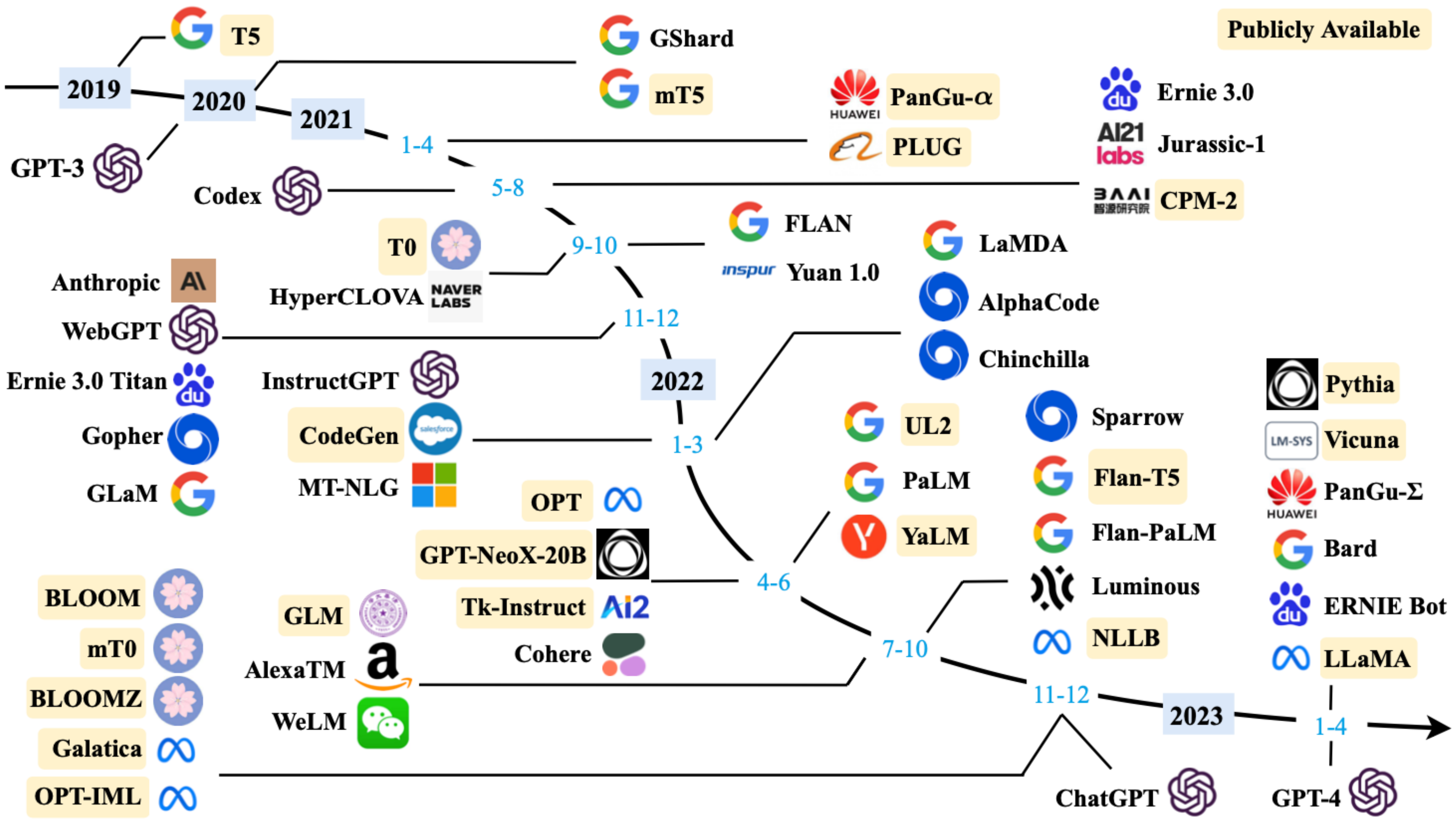
Large Language Models
(LLMs)
Foundation Models

Large Language Models (LLM)
(GPT-3, ChatGPT, PaLM, BLOOM, OPT-175B, LLaMA)



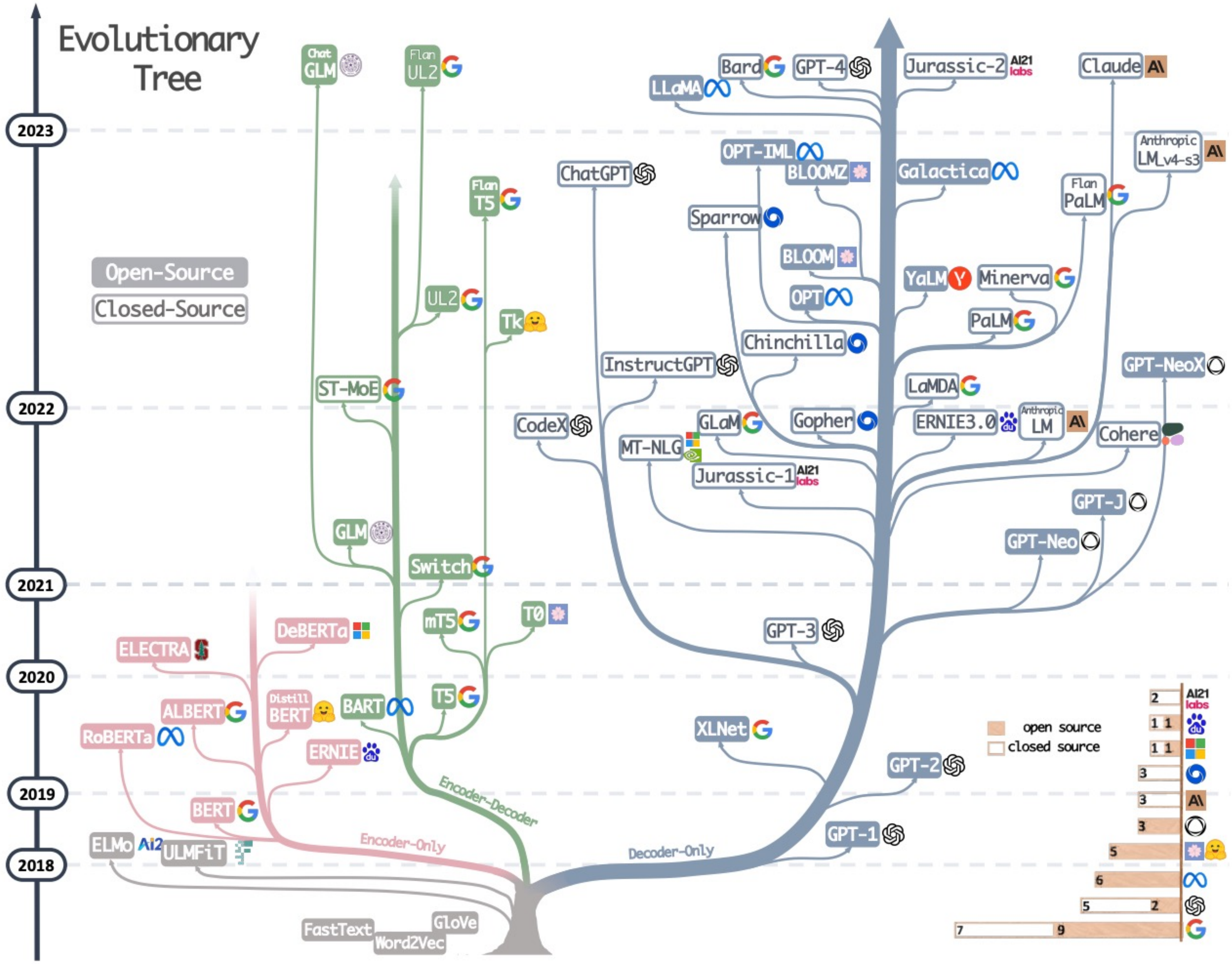
Source: https://lifearchitect.ai/models/

Large Language Models (LLMs) (larger than 10B)



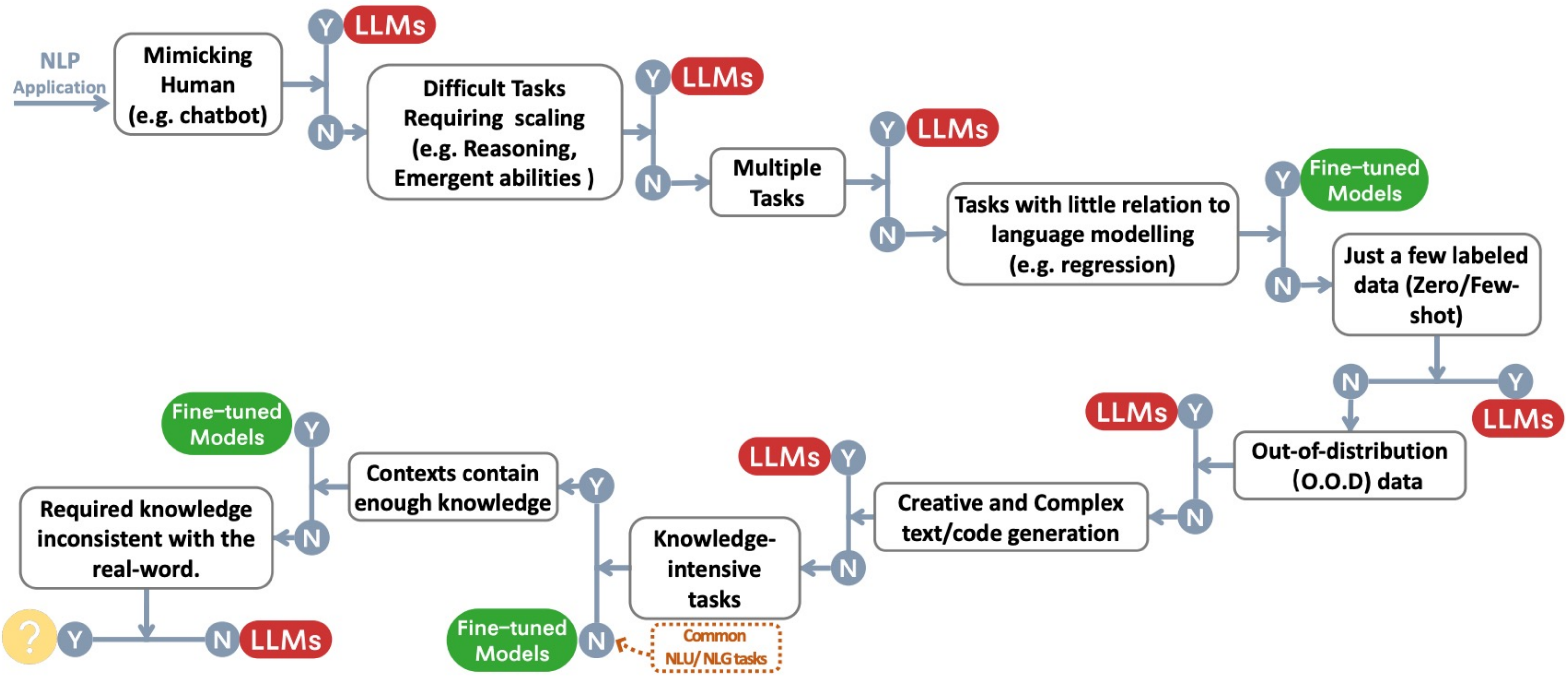
Source: Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min et al. (2023) "A Survey of Large Language Models." arXiv preprint arXiv:2303.18223.

Large Language Models (LLMs) Evolutionary Tree



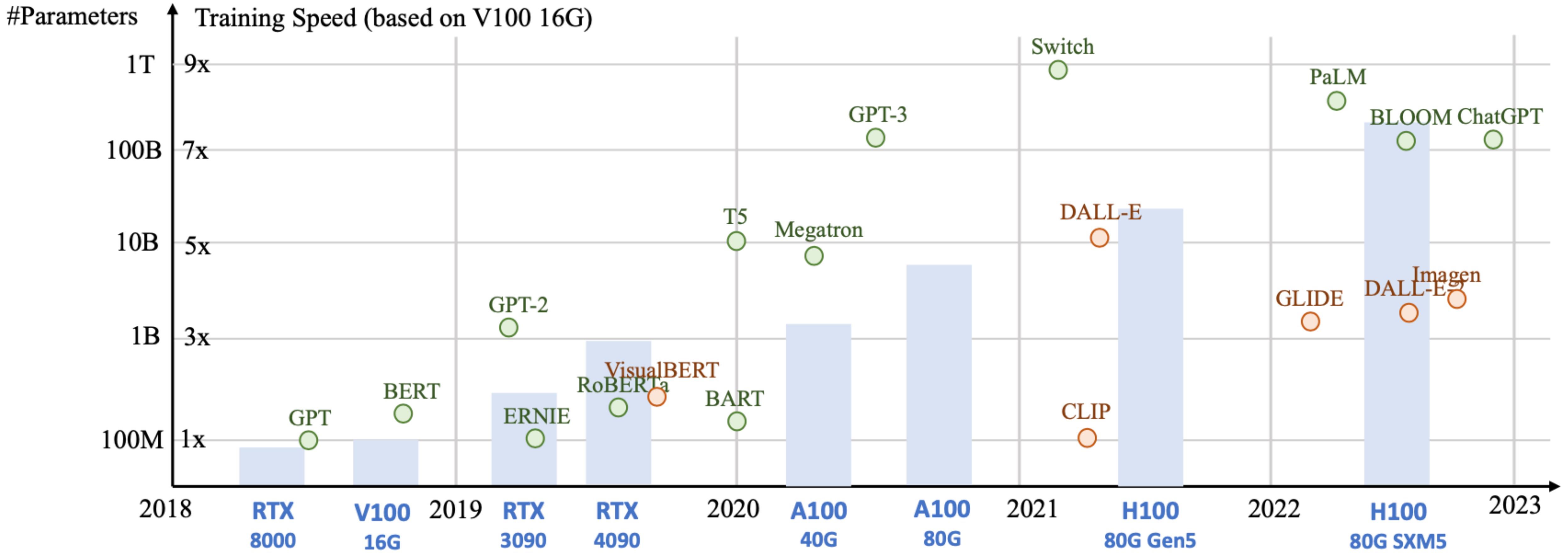
Source: Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu (2023). "Harnessing the power of llms in practice: A survey on chatgpt and beyond." arXiv preprint arXiv:2304.13712.

The Decision Flow for Choosing LLMs or Fine-tuned Models for NLP Applications



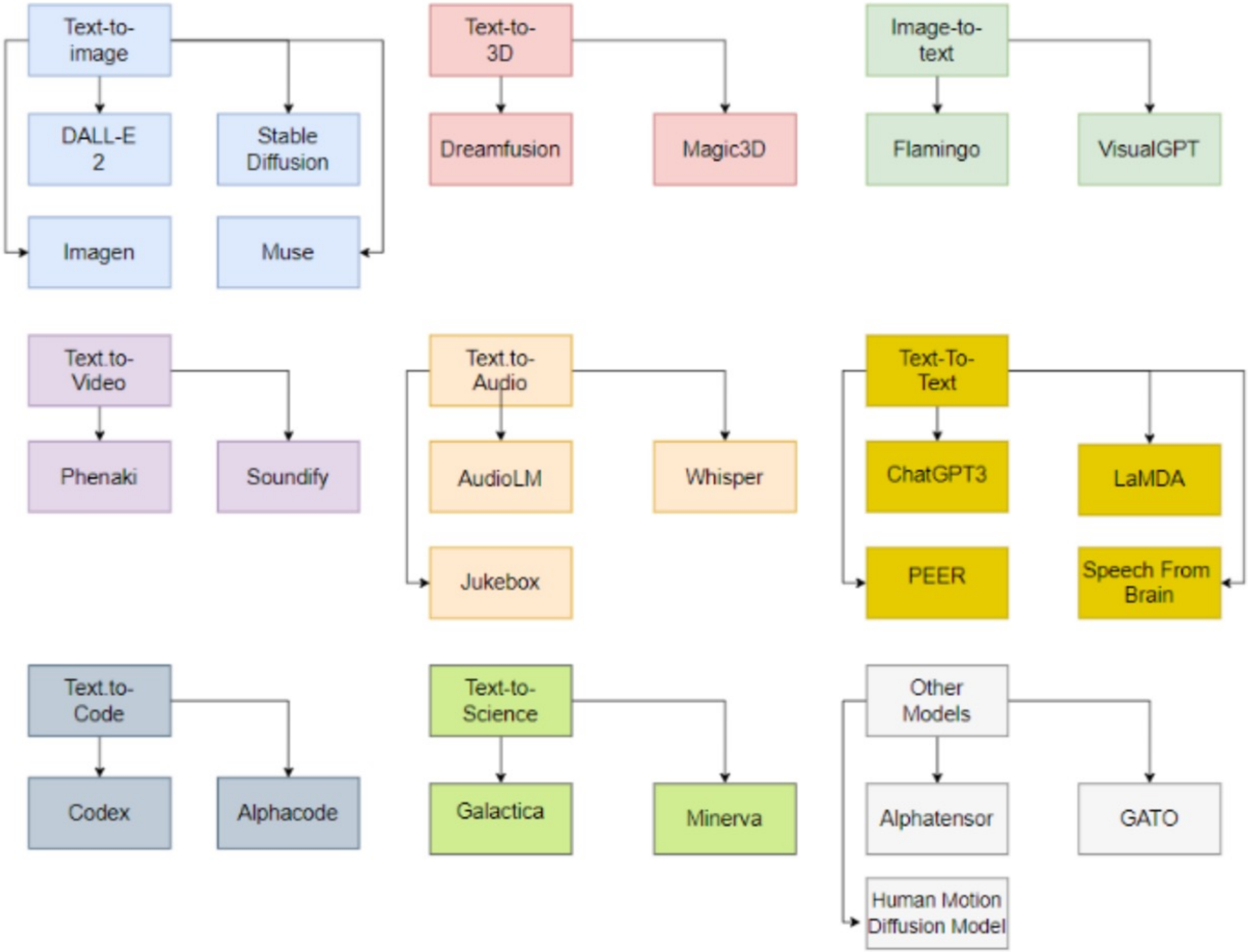
Source: Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu (2023). "Harnessing the power of llms in practice: A survey on chatgpt and beyond." arXiv preprint arXiv:2304.13712.

Generative AI Foundation Models



Source: Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun (2023). "A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT." arXiv preprint arXiv:2303.04226.

Generative AI Models



ChatGPT is not all you need

Attention is all you need

Source: Gozalo-Brizuela, Roberto, and Eduardo C. Garrido-Merchan (2023). "ChatGPT is not all you need. A State of the Art Review of large Generative AI models." arXiv preprint arXiv:2301.04655 (2023).

Meta Llama-2 70B: Best Open Source and Commercial LLM (Llama-2, Falcon, MPT)

MODEL SIZE (PARAMETERS)	PRETRAINED	FINE-TUNED FOR CHAT USE CASES
7B 13B 70B	**Model architecture:** **Pretraining Tokens:** 2 Trillion **Context Length:** 4096	**Data collection for helpfulness and safety:** **Supervised fine-tuning:** Over 100,000 **Human Preferences:** Over 1,000,000

Llama 2 pretrained models are trained on 2 trillion tokens, and have double the context length than Llama 1. Its fine-tuned models have been trained on over 1 million human annotations.

Source: https://ai.meta.com/llama/

Meta **Llama-2 70B**: Best Open Source and Commercial LLM (Llama-2, Falcon, MPT)

Benchmark (Higher is better)	MPT (7B)	Falcon (7B)	Llama-2 (7B)	Llama-2 (13B)	MPT (30B)	Falcon (40B)	Llama-1 (65B)	Llama-2 (70B)
MMLU	26.8	26.2	45.3	54.8	46.9	55.4	63.4	68.9
TriviaQA	59.6	56.8	68.9	77.2	71.3	78.6	84.5	85.0
Natural Questions	17.8	18.1	22.7	28.0	23.0	29.5	31.0	33.0
GSM8K	6.8	6.8	14.6	28.7	15.2	19.6	50.9	56.8
HumanEval	18.3	N/A	12.8	18.3	25.0	N/A	23.7	29.9
AGIEval (English tasks only)	23.5	21.2	29.3	39.1	33.8	37.0	47.6	54.2
BoolQ	75.0	67.5	77.4	81.7	79.0	83.1	85.3	85.0

Llama 2 outperforms other open source language models on many external benchmarks, including reasoning, coding, proficiency, and knowledge tests.

Source: https://ai.meta.com/llama/

Llama-2: Comparison to closed-source models (GPT-3.5, GPT-4, PaLM) on academic benchmarks

Benchmark (shots)	GPT-3.5	GPT-4	PaLM	PaLM-2-L	Llama 2
MMLU (5-shot)	70.0	**86.4**	69.3	78.3	68.9
TriviaQA (1-shot)	–	–	81.4	**86.1**	85.0
Natural Questions (1-shot)	–	–	29.3	**37.5**	33.0
GSM8K (8-shot)	57.1	**92.0**	56.5	80.7	56.8
HumanEval (0-shot)	48.1	**67.0**	26.2	–	29.9
BIG-Bench Hard (3-shot)	–	–	52.3	**65.7**	51.2

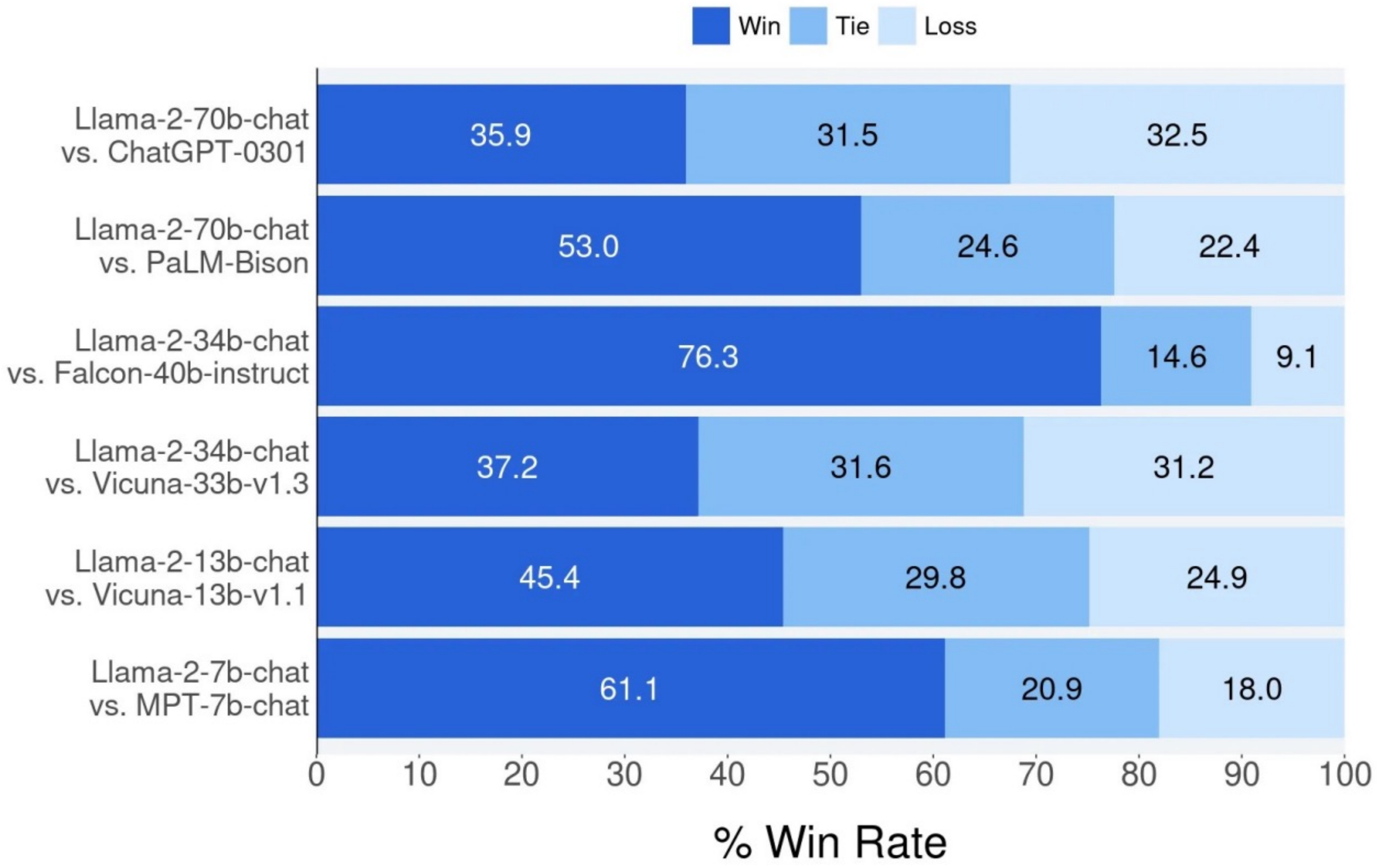
Results for GPT-3.5 and GPT-4 are from OpenAI (2023).

Results for the PaLM model are from Chowdhery et al. (2022).

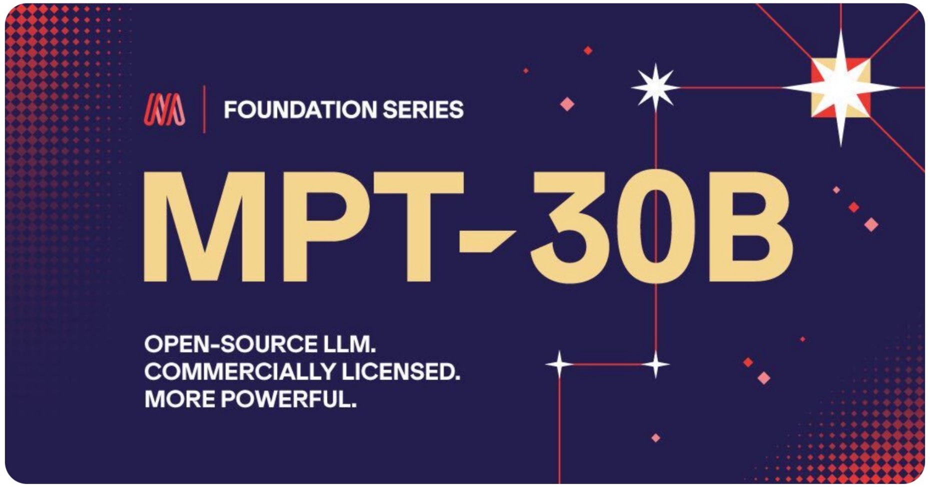
Results for the PaLM-2-L are from Anil et al. (2023).

Source: Touvron, Hugo, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov et al. (2023) "Llama 2: Open Foundation and Fine-Tuned Chat Models." arXiv preprint arXiv:2307.09288 (2023).

Llama-2 Chat: Helpfulness Human Evaluation



Source: Touvron, Hugo, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov et al. (2023) "Llama 2: Open Foundation and Fine-Tuned Chat Models." arXiv preprint arXiv:2307.09288 (2023).

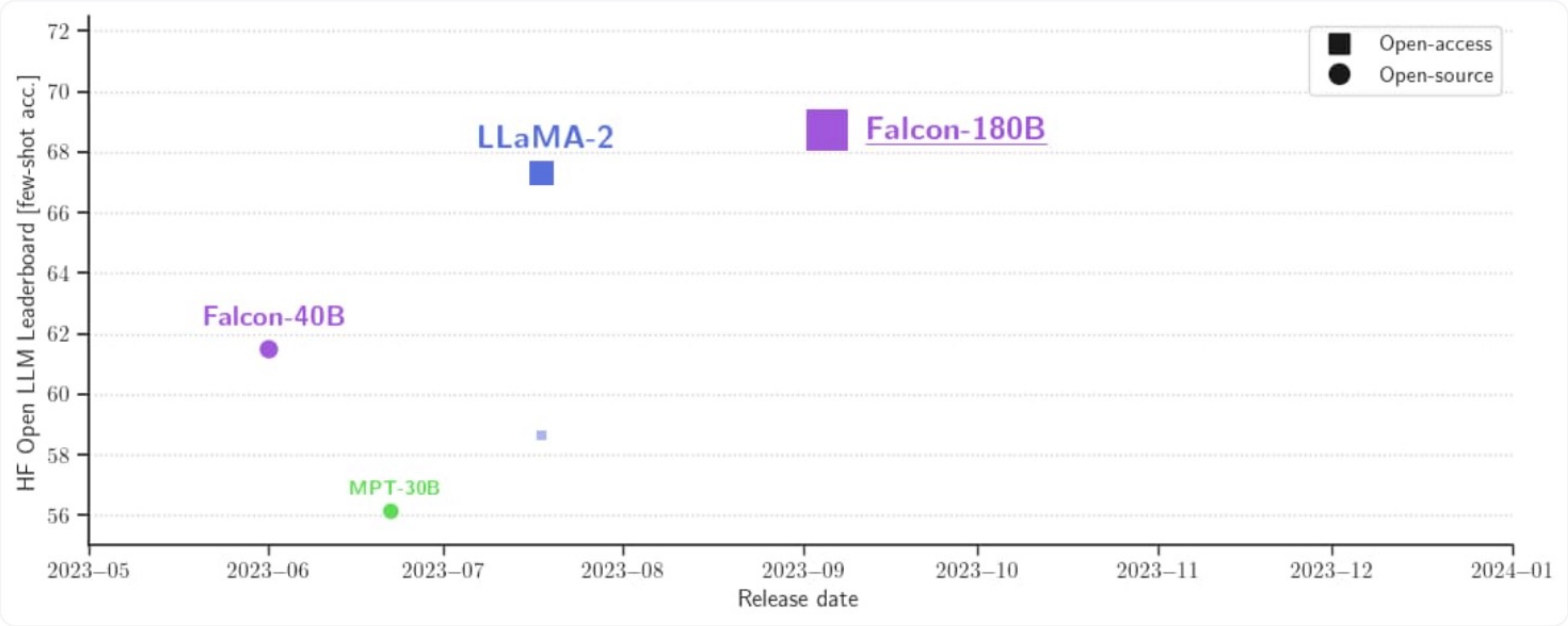


MPT-30B, MPT-7B
LLaMa-30B, LLaMa-7B

Model Purpose	Model Series	Model	Sequence Length	Accuracy (Pass@1)	Externally Reported Pass@1 & [Source]
General Purpose	MPT	MPT-30B	1024	25.00%	N/A
		MPT-30B Chat	1024	37.20%	N/A
		MPT-30B Instruct	1024	26.20%	N/A
		MPT-7B	1024	15.90%	N/A
		MPT-7B Instruct	1024	16.50%	N/A
	LLaMa	LLaMa-7B	1024	10.10%	10.5% [1]
		LLaMa-13B	1024	16.50%	15.8% [1]
		LLaMa-30B	1024	20.10%	21.7% [1]
	Falcon	Falcon-40B	1024	1.2%* (did not generate code)	N/A
		Falcon-40B Instruct	1024	0.6%* (did not generate code)	18.9% [2]

Source: https://www.mosaicml.com/blog/mpt-30b

Falcon 180B



Source: https://huggingface.co/blog/falcon-180b

Falcon 180B, Llama 2 70B, MPT 30B

Model	Size	Leaderboard score	Commercial use or license	Pretraining length
Falcon	180B	68.74	🟠	3,500B
Llama 2	70B	67.35	🟠	2,000B
LLaMA	65B	64.23	🔴	1,400B
Falcon	40B	61.48	🟢	1,000B
MPT	30B	56.15	🟢	1,000B

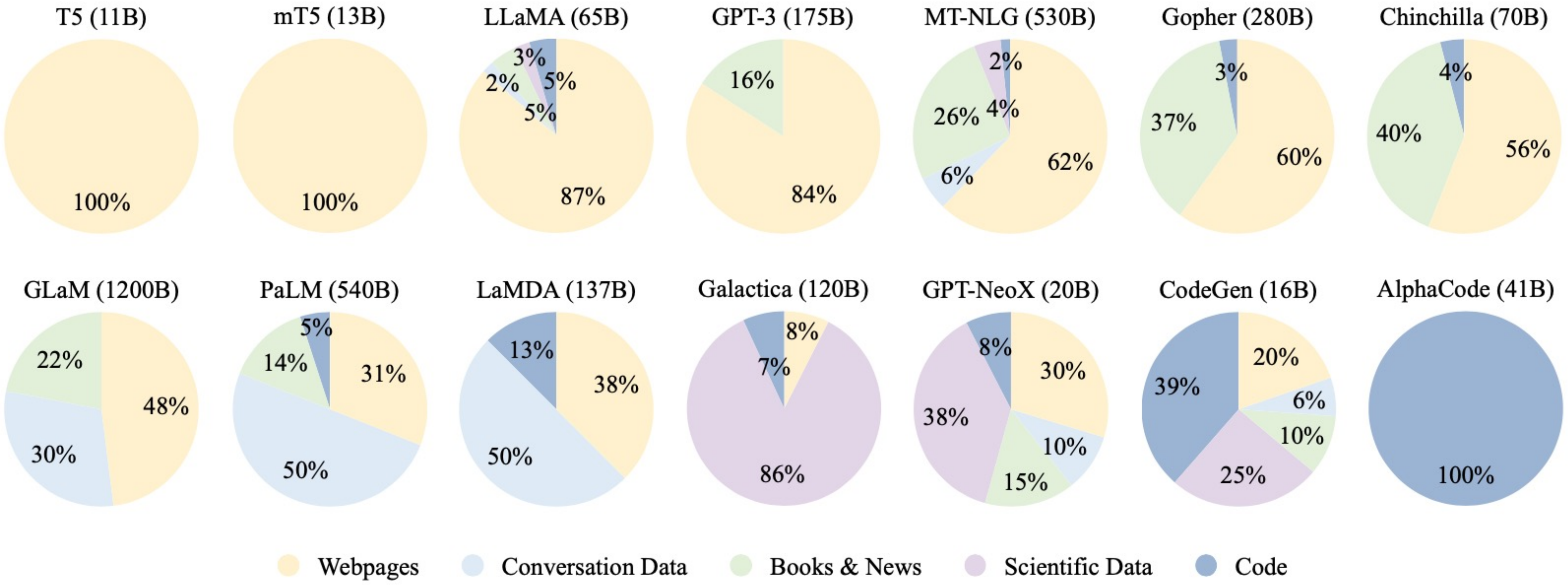
Falcon 180B
Hardware requirements

NVIDIA A100 80 GB: $16,135

	Type	Kind	Memory	Example
Falcon 180B	Training	Full fine-tuning	5120GB	8x 8x A100 80GB
Falcon 180B	Training	LoRA with ZeRO-3	1280GB	2x 8x A100 80GB
Falcon 180B	Training	QLoRA	160GB	2x A100 80GB
Falcon 180B	Inference	BF16/FP16	640GB	8x A100 80GB
Falcon 180B	Inference	GPTQ/int4	320GB	8x A100 40GB

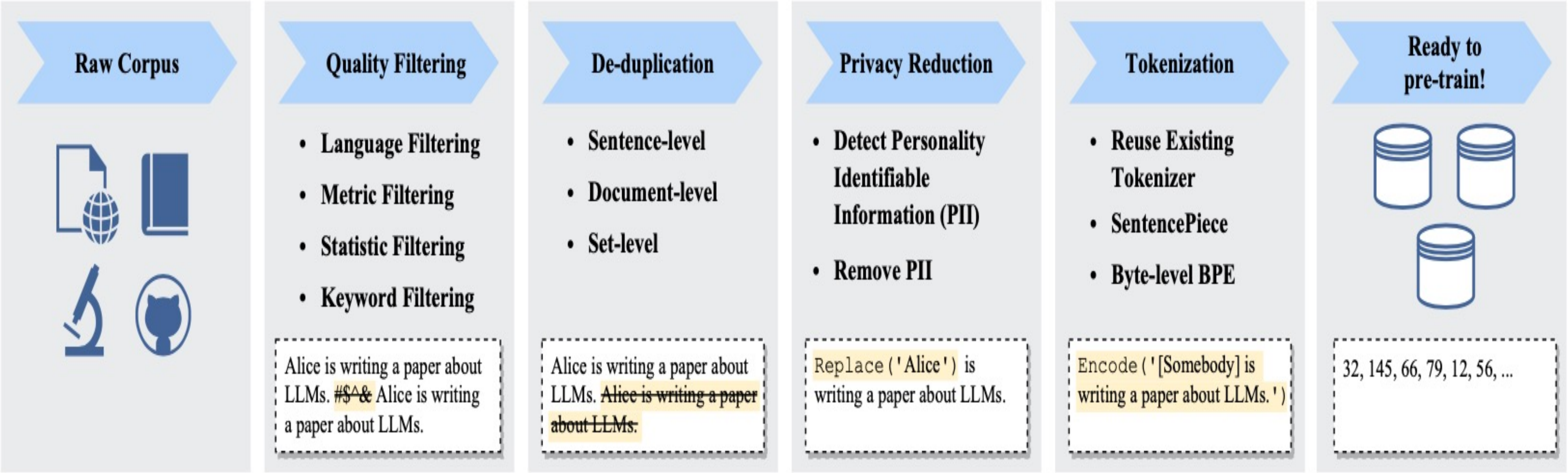
Source: https://huggingface.co/blog/falcon-180b

Ratios of various data sources in the pre-training data for existing LLMs



Source: Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min et al. (2023) "A Survey of Large Language Models." arXiv preprint arXiv:2303.18223.

Typical Data Preprocessing Pipeline for Pre-training Large Language Models (LLMs)



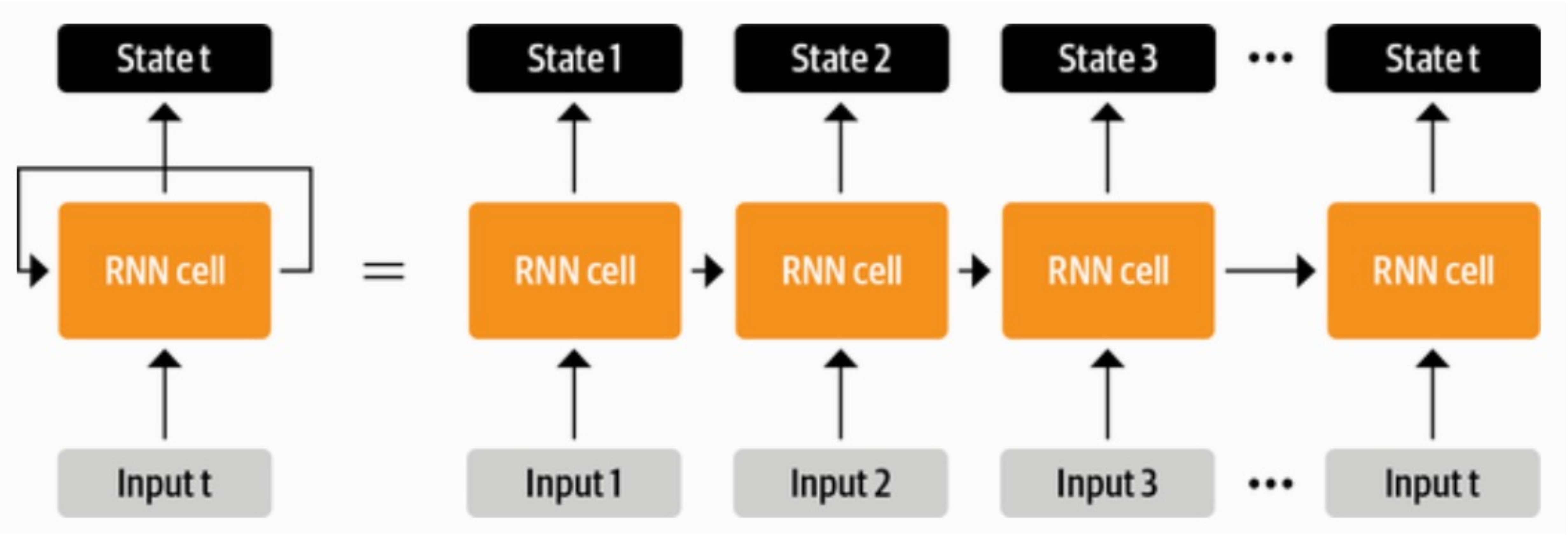
Source: Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min et al. (2023) "A Survey of Large Language Models." arXiv preprint arXiv:2303.18223.

The Encoder-Decoder Framework

- The encoder-decoder framework
- Attention Mechanisms
- Transfer Learning in NLP

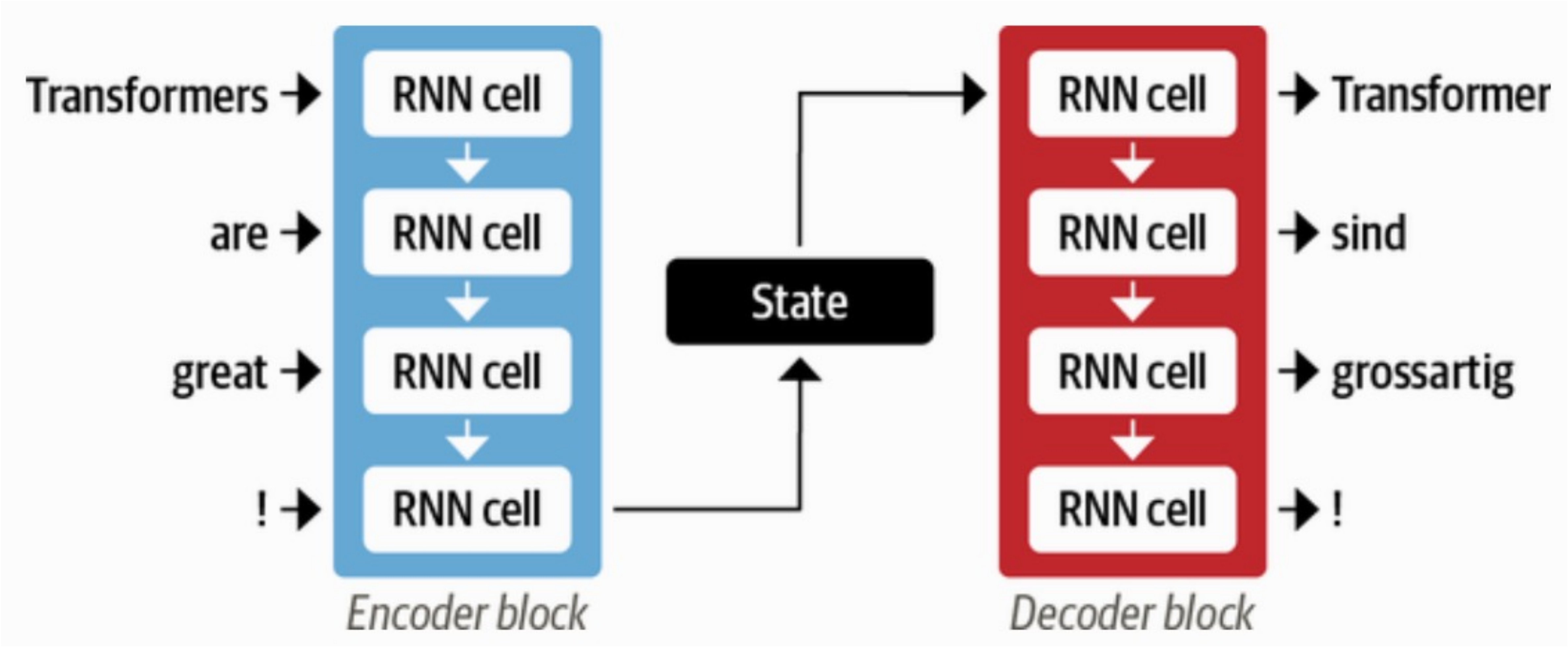
Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

RNN



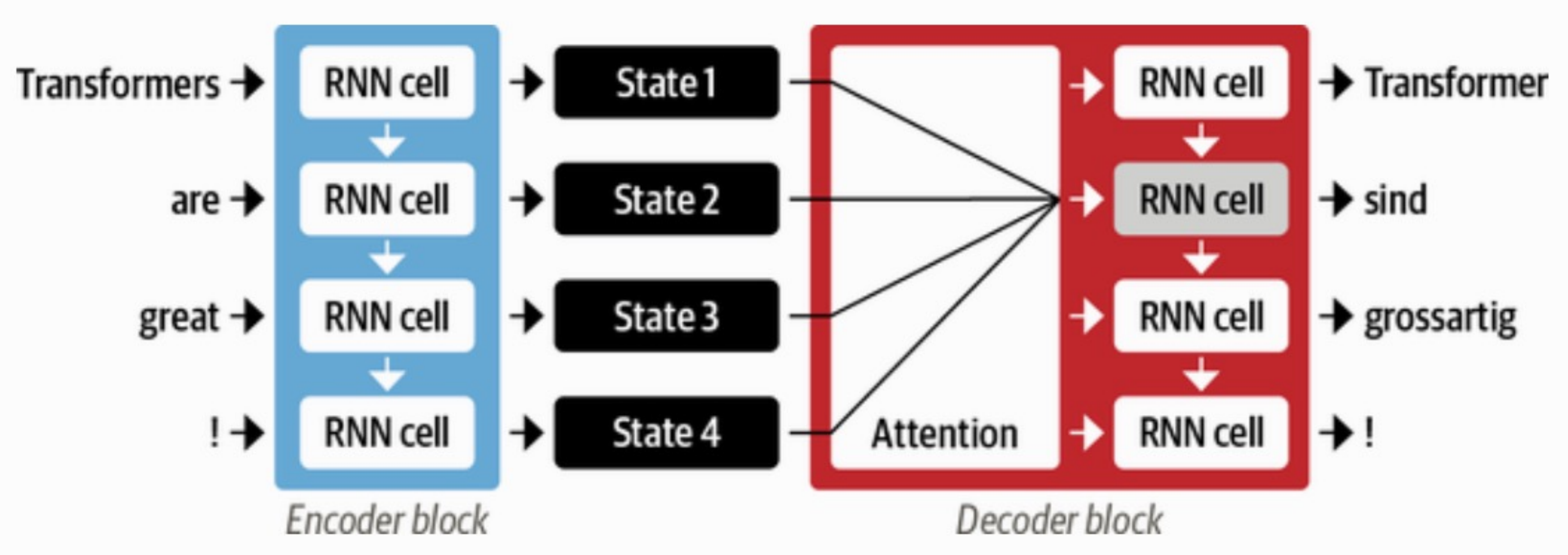
Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

An encoder-decoder architecture with a pair of RNN



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Attention Mechanisms

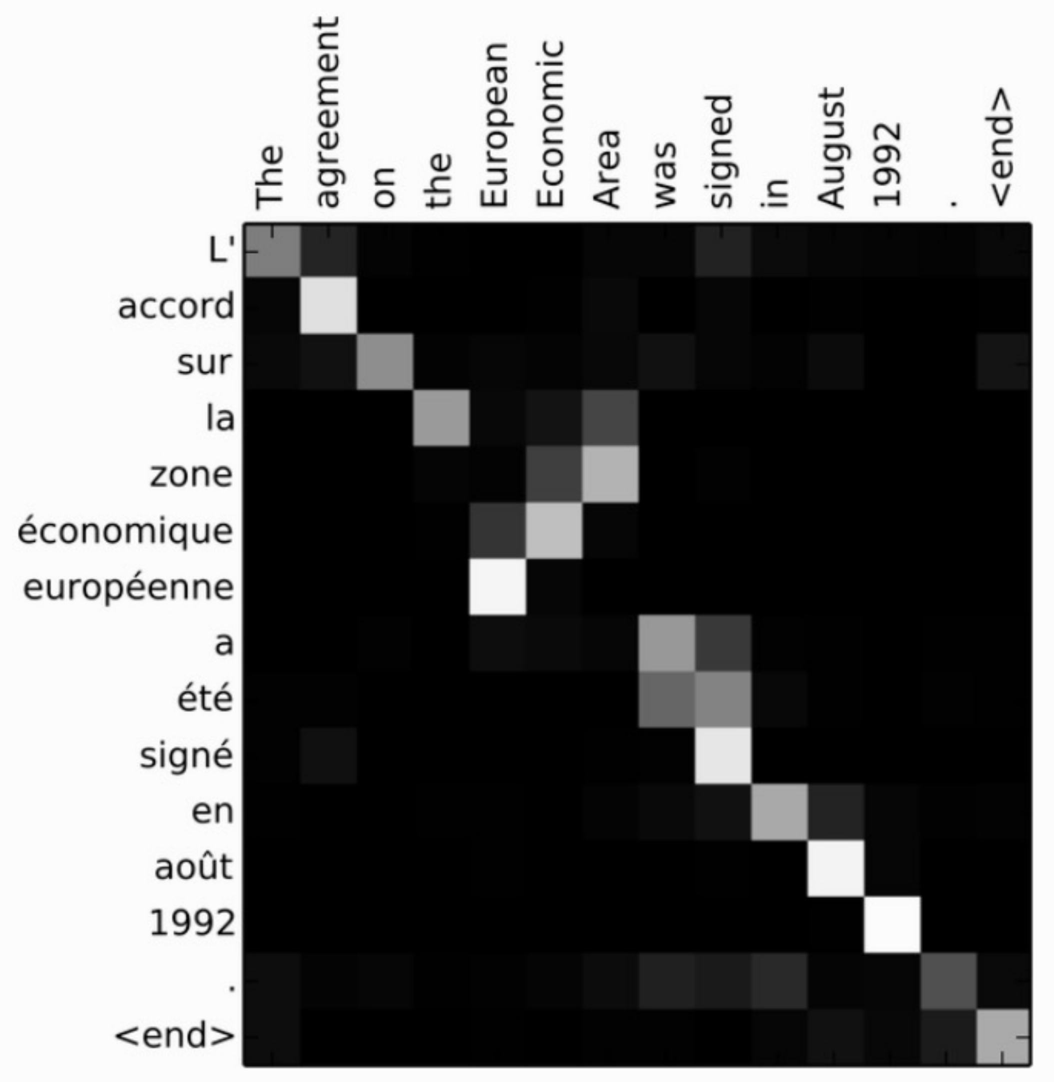


An encoder-decoder architecture with an attention mechanism

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

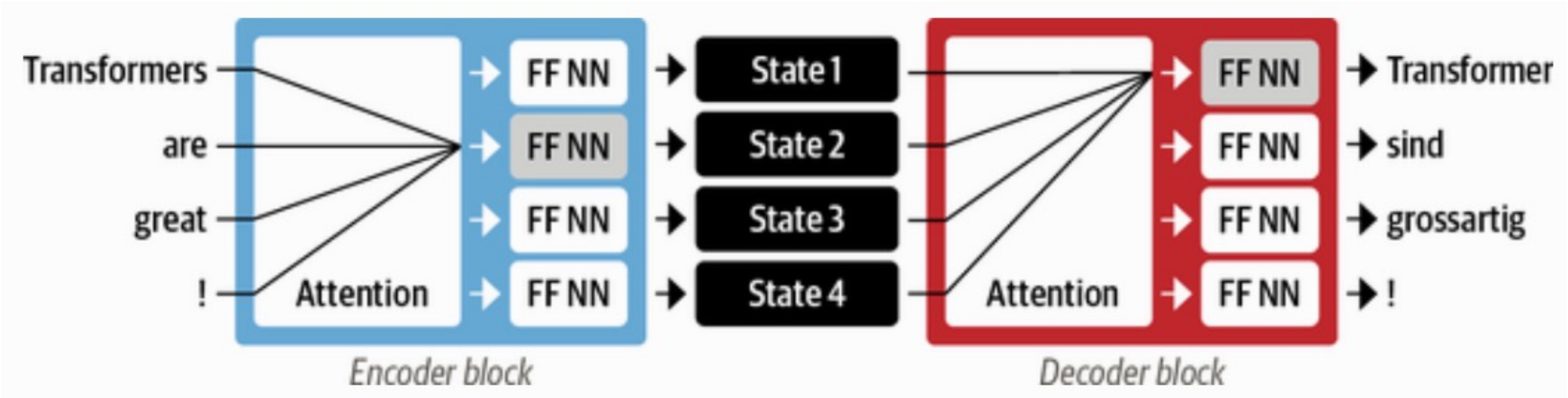
RNN Encoder-Decoder

alignment of words in English and the generated translation in French



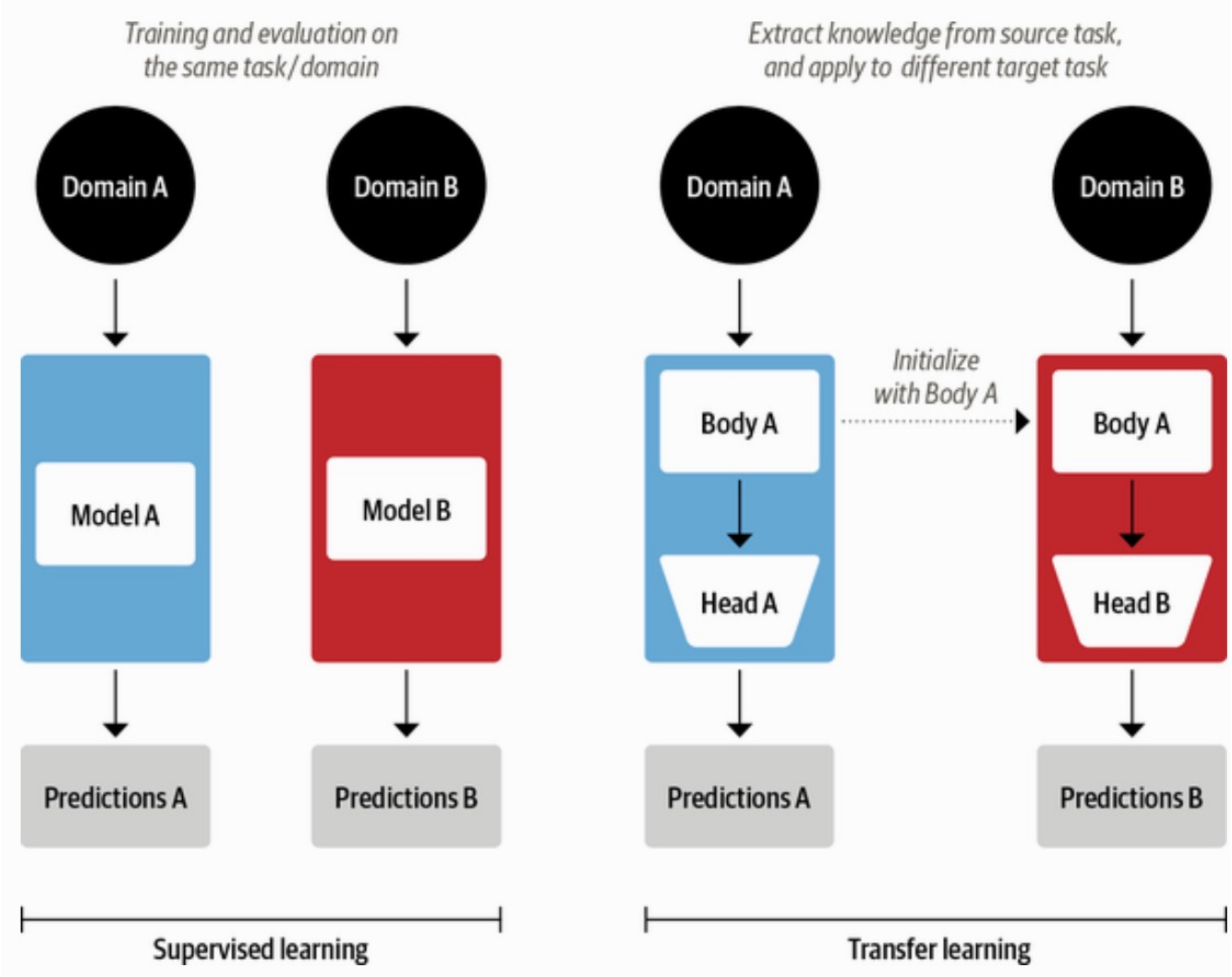
Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Encoder-Decoder Architecture of the Original Transformer



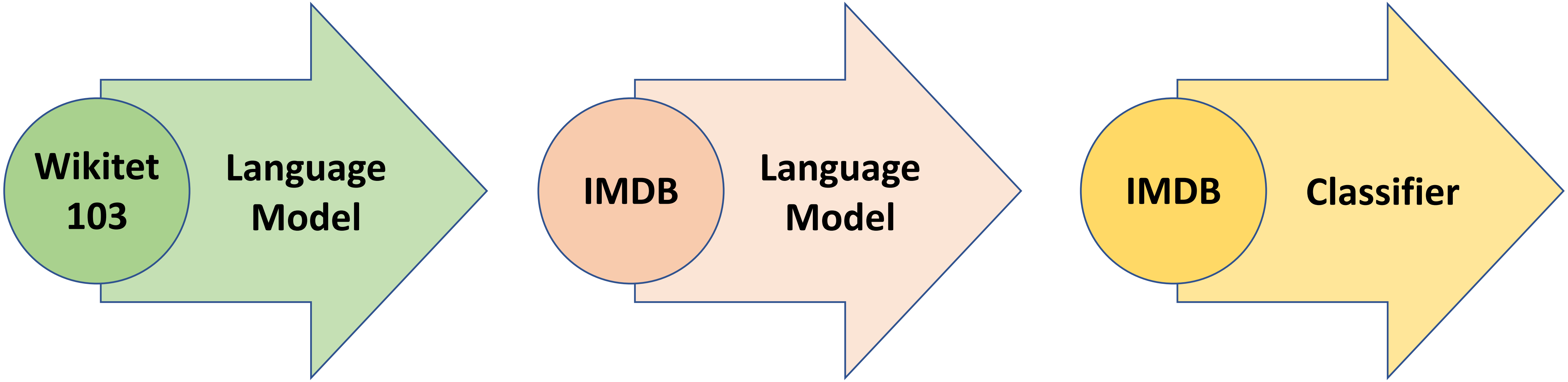
Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Comparison of Traditional Supervised Learning and Transfer Learning



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

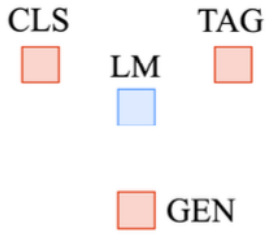
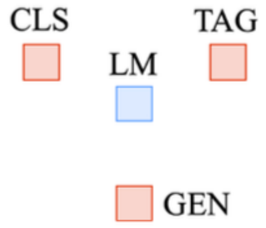
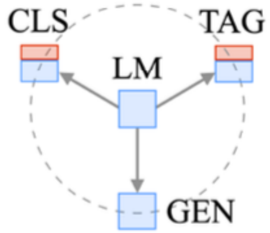
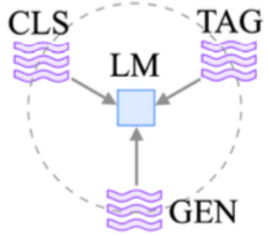
ULMFiT: 3 Steps
Transfer Learning in NLP



1. Pretraining 2. Domain adaptation 3. Fine-tuning

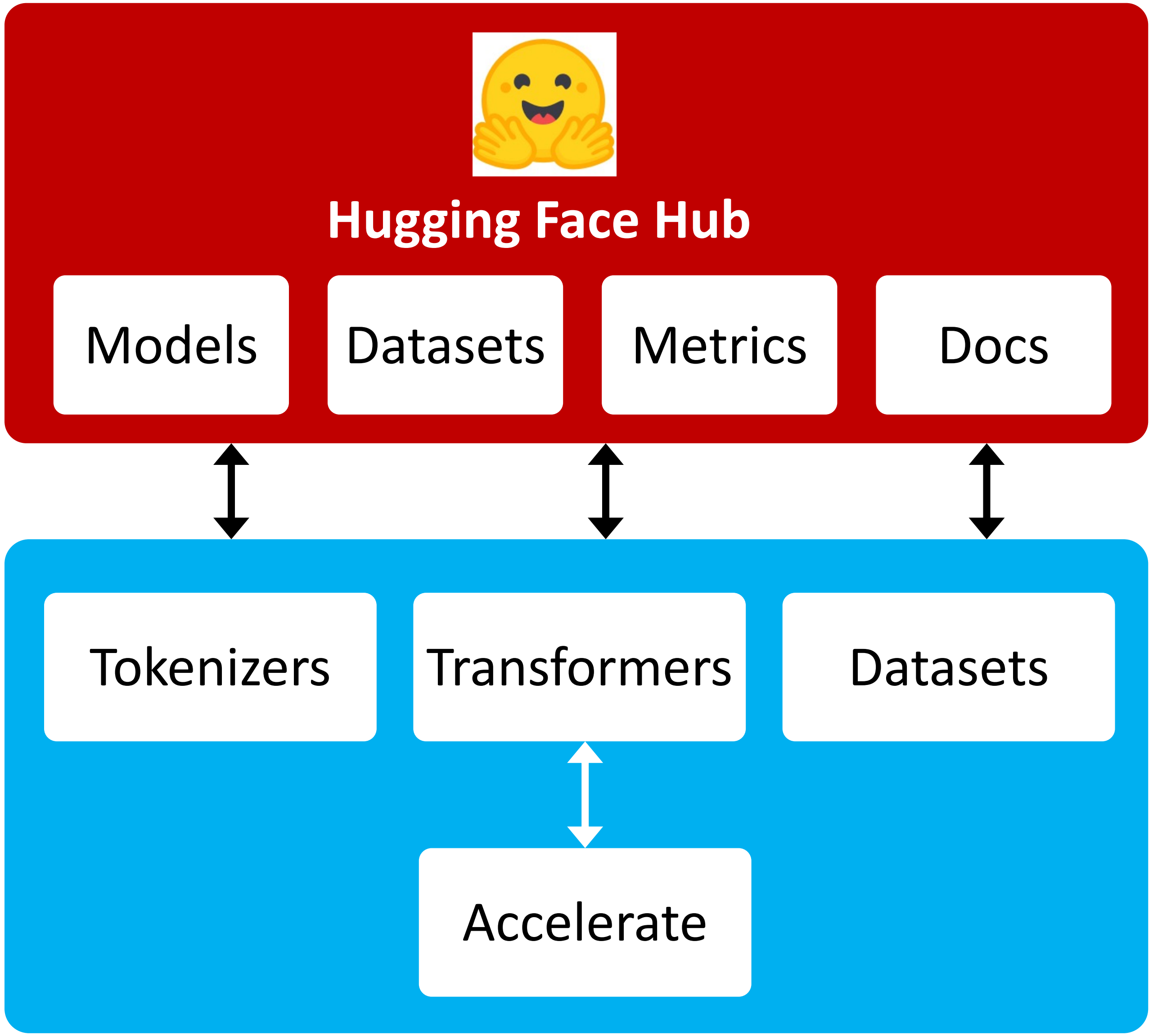
Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

Four Paradigms in NLP

Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Feature (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

Source: Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. (2023) "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing." ACM Computing Surveys 55, no. 9 (2023): 1-35.

An overview of the Hugging Face Ecosystem



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

A typical pipeline for training transformer models

with the Datasets, Tokenizers, and Transformers libraries

Datasets	Tokenizers	Transformers	Datasets
Load and process datasets	Tokenize input texts	Load models, train and infer	Load metrics evaluate models

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

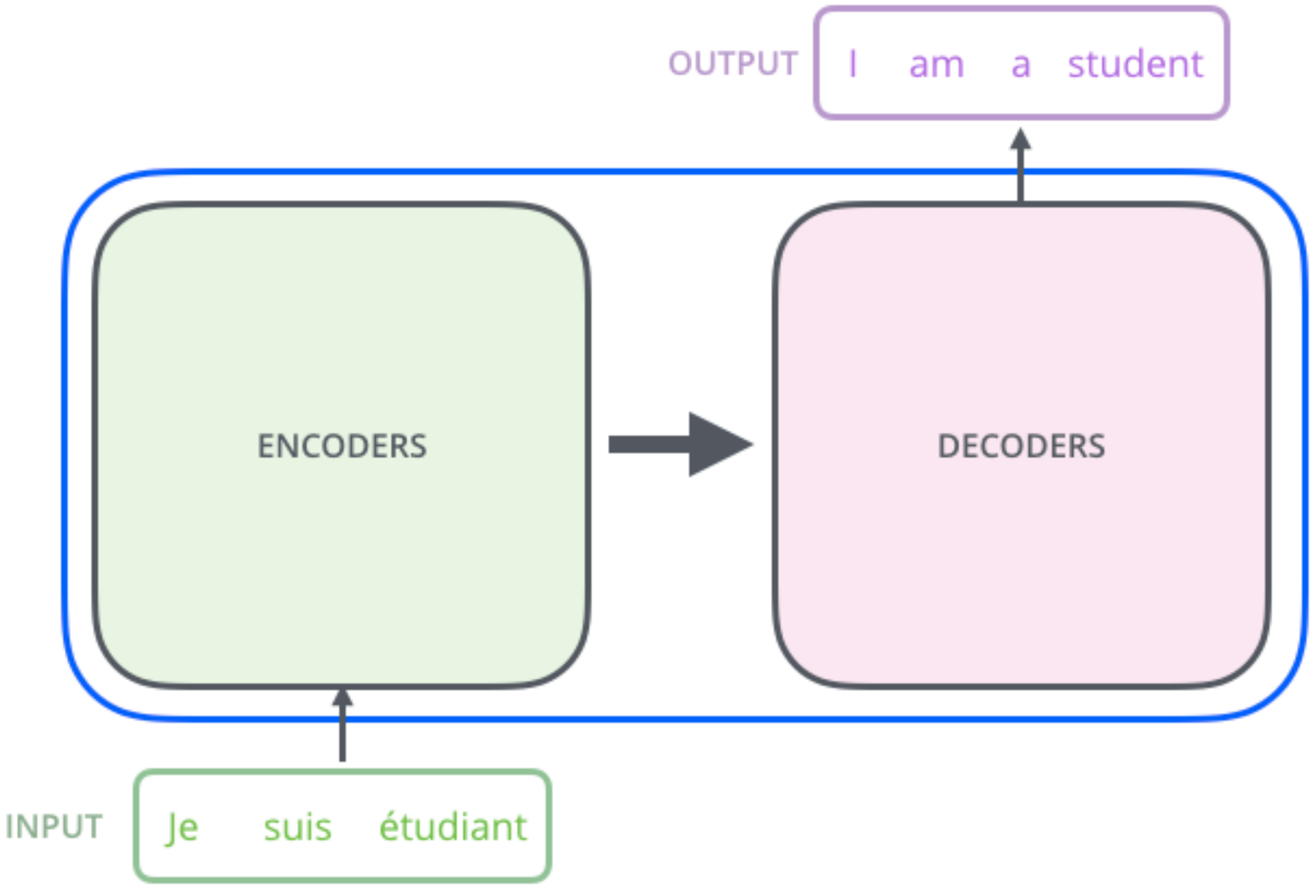
The Illustrated Transformer

Jay Alammar (2018)

Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

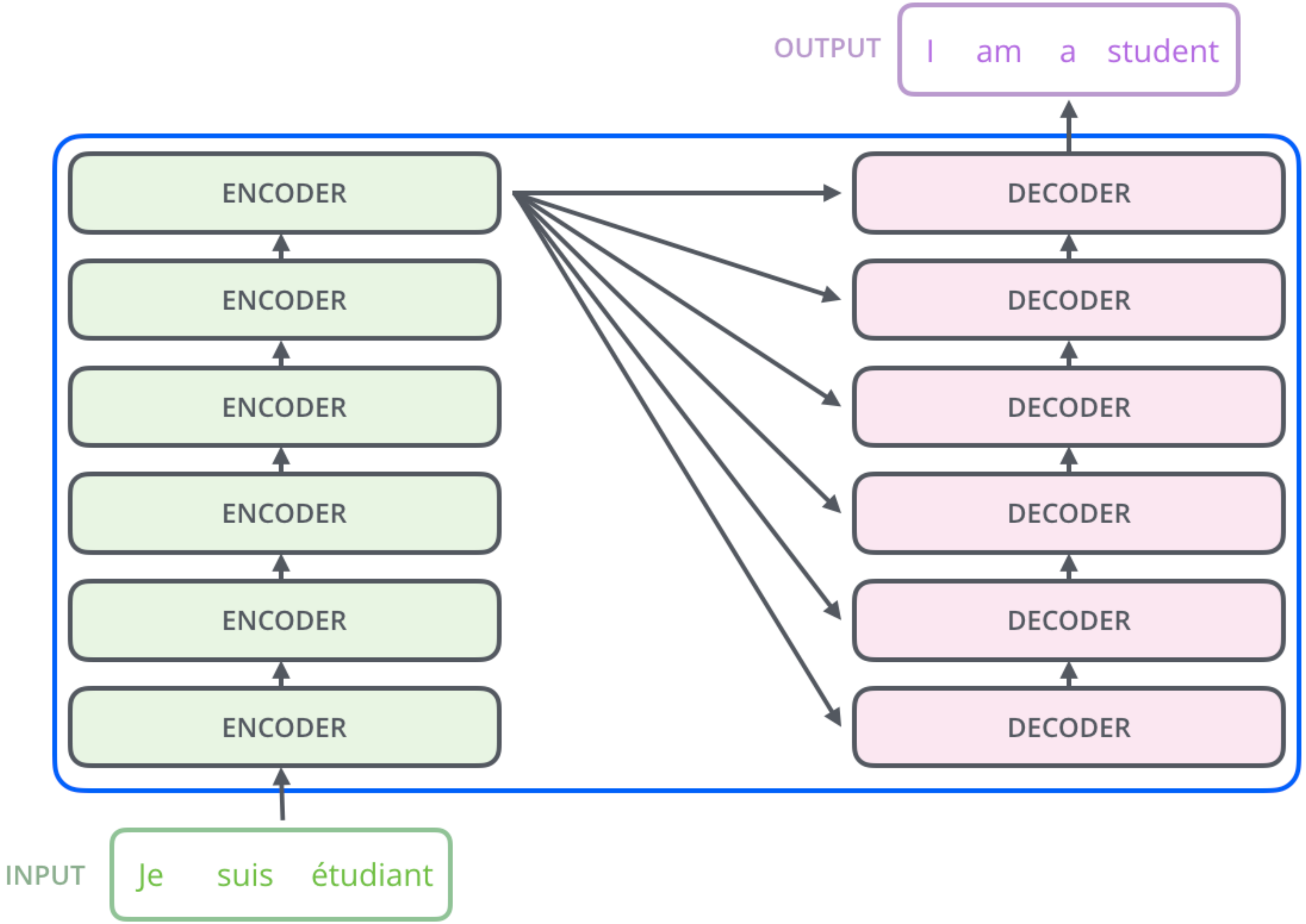
Jay Alammar (2018)



Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

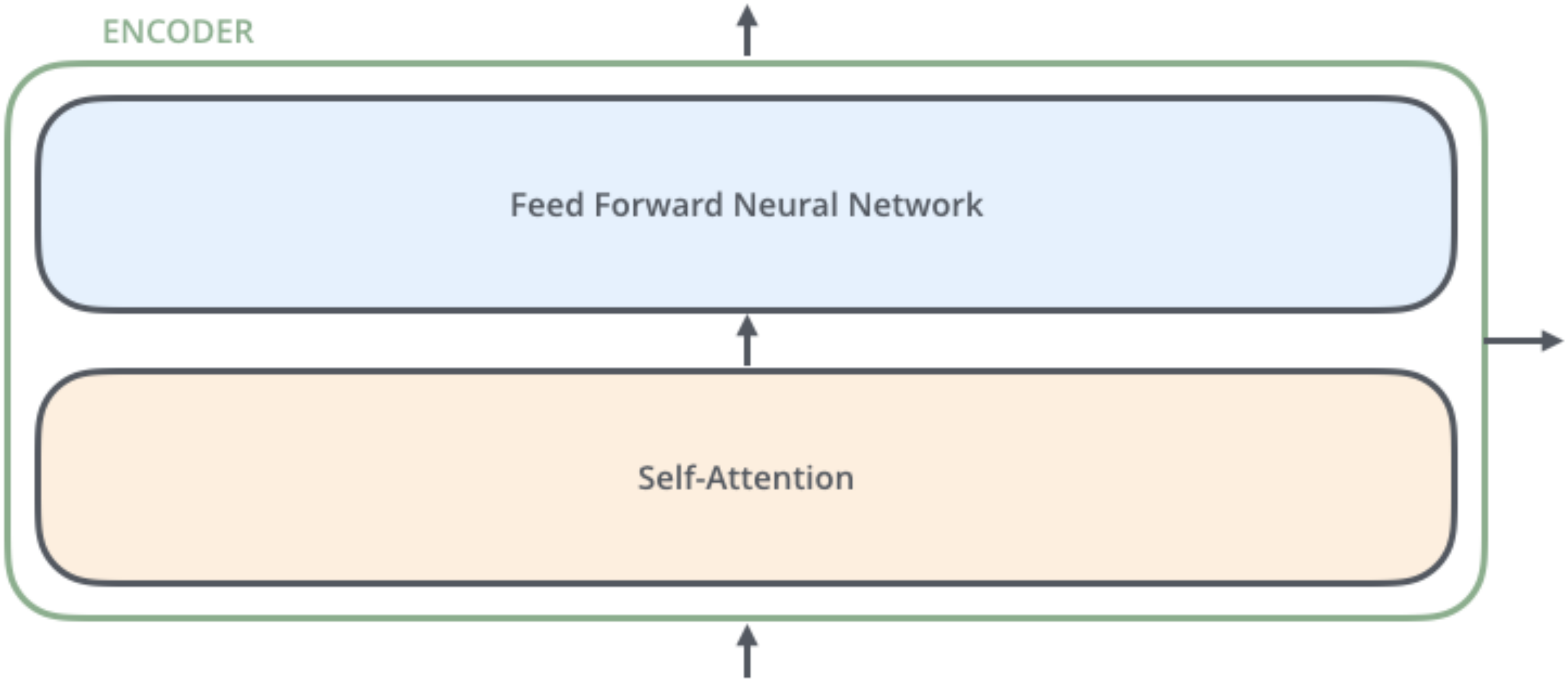
Jay Alammar (2018)



Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

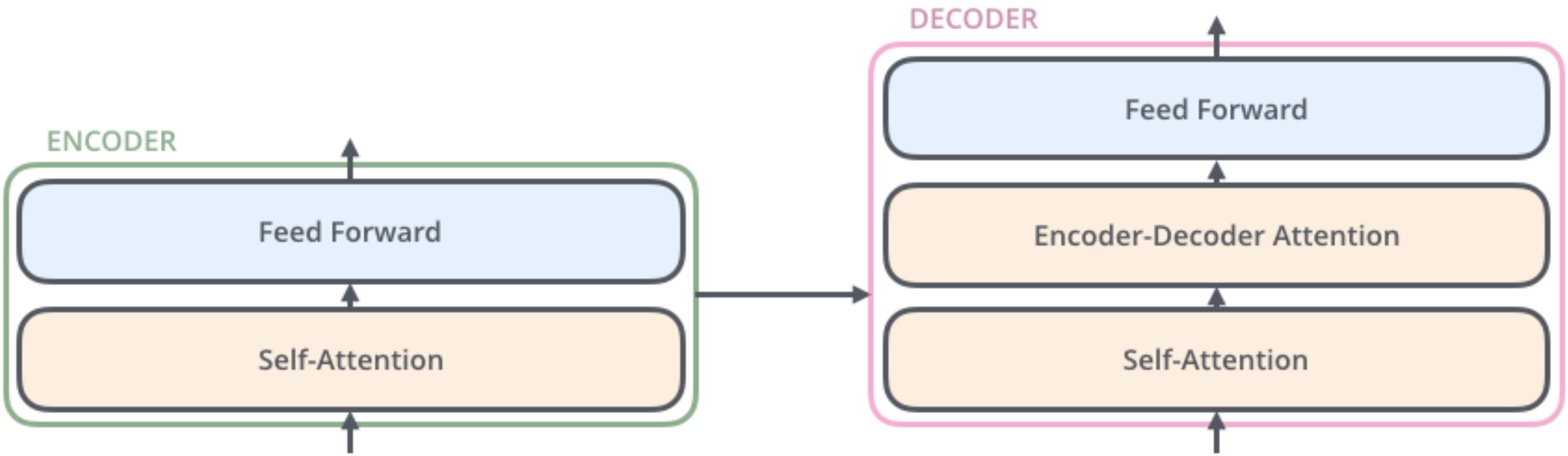
Jay Alammar (2018)



Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

Jay Alammar (2018)



Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

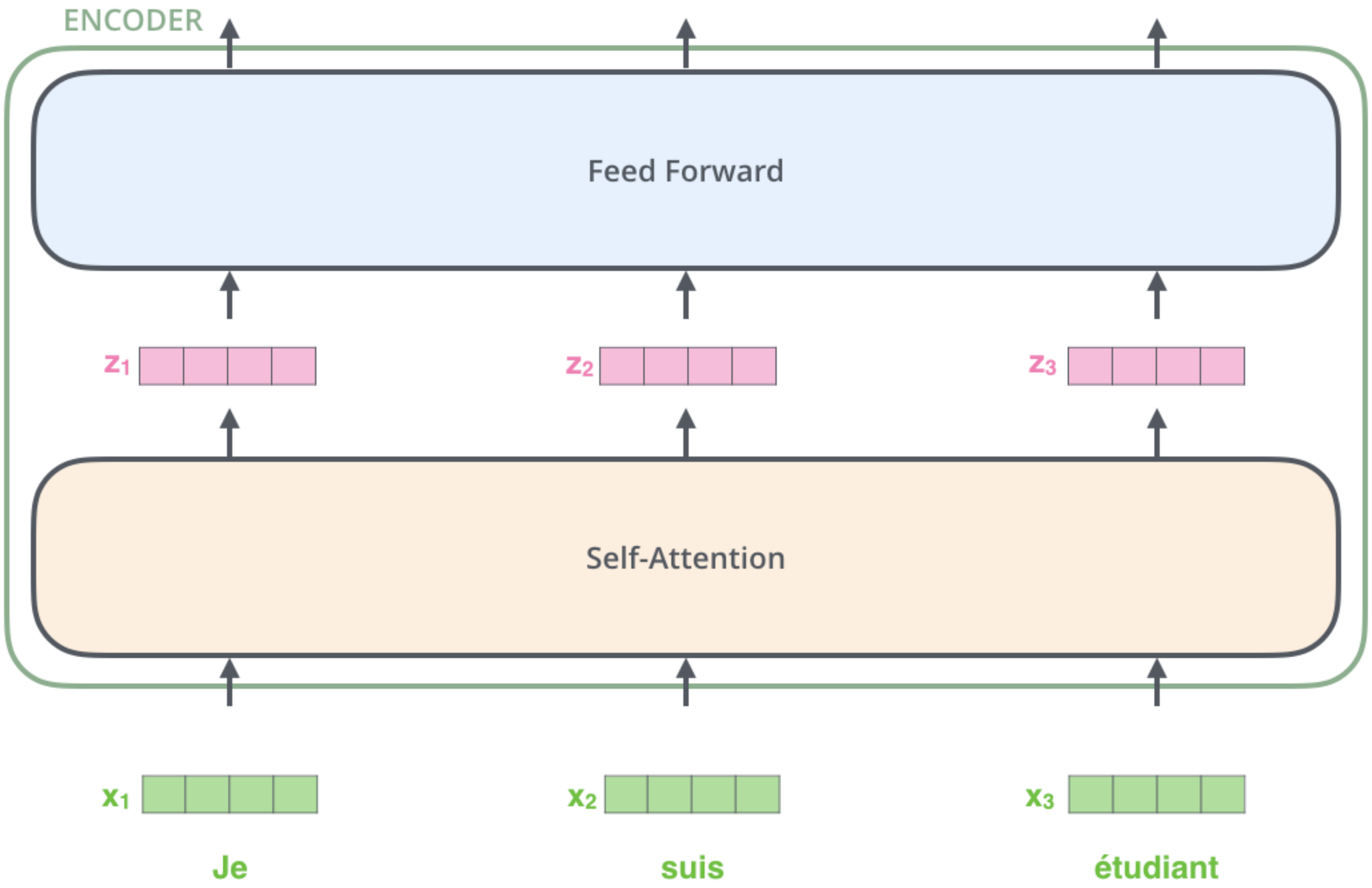
Jay Alammar (2018)

Each word is embedded into a vector of size 512.

Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

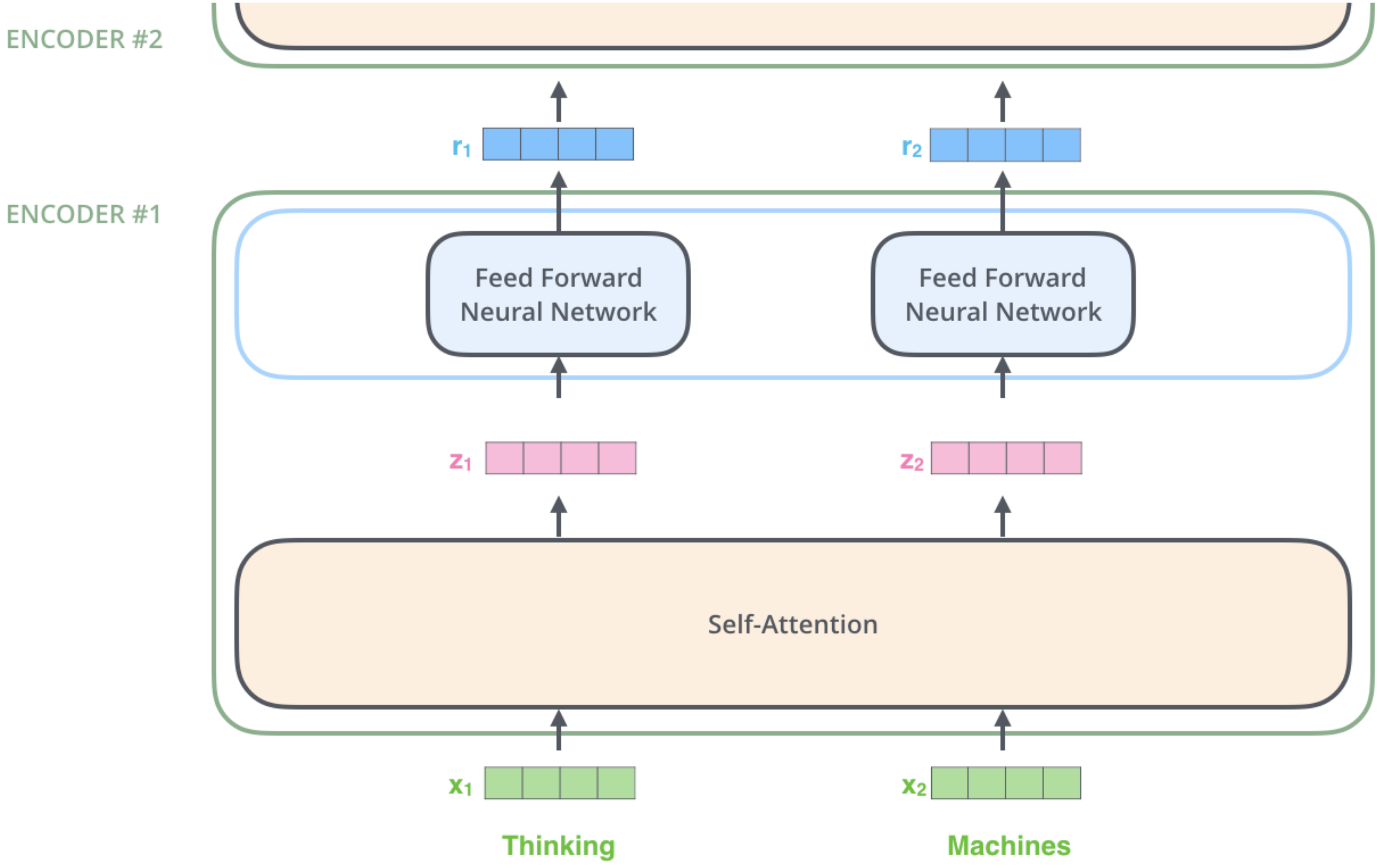
Jay Alammar (2018)



Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

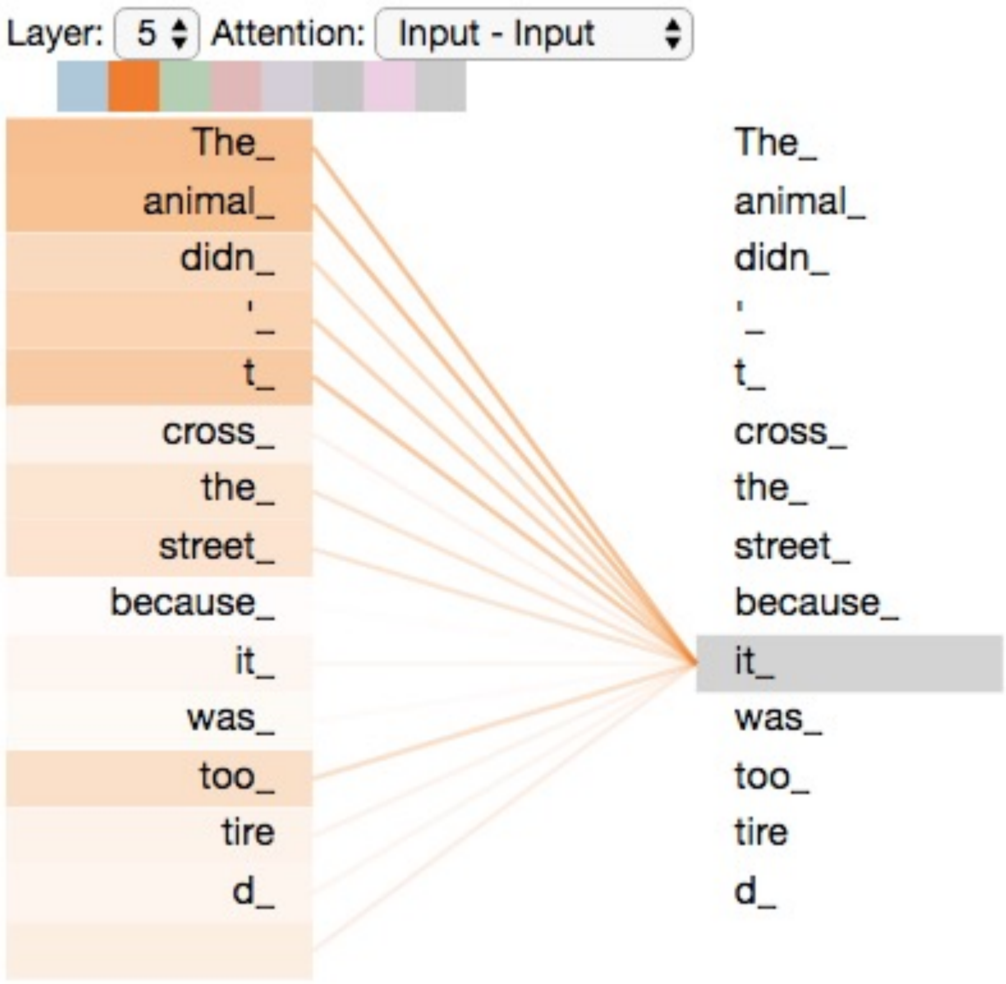
Jay Alammar (2018)



Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

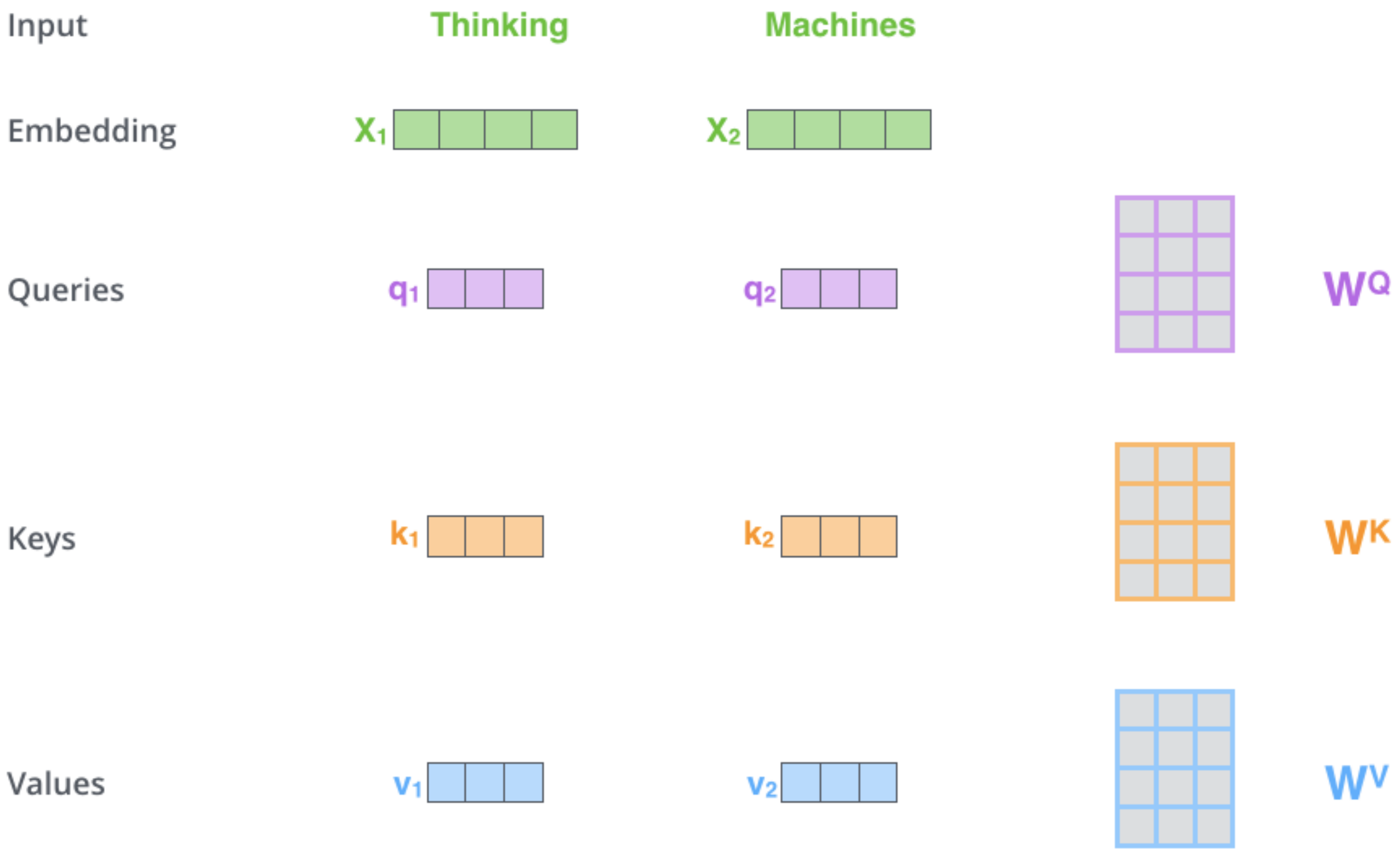
Jay Alammar (2018)



Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Multiplying x1 by the WQ weight matrix produces q1, the "query" vector associated with that word.

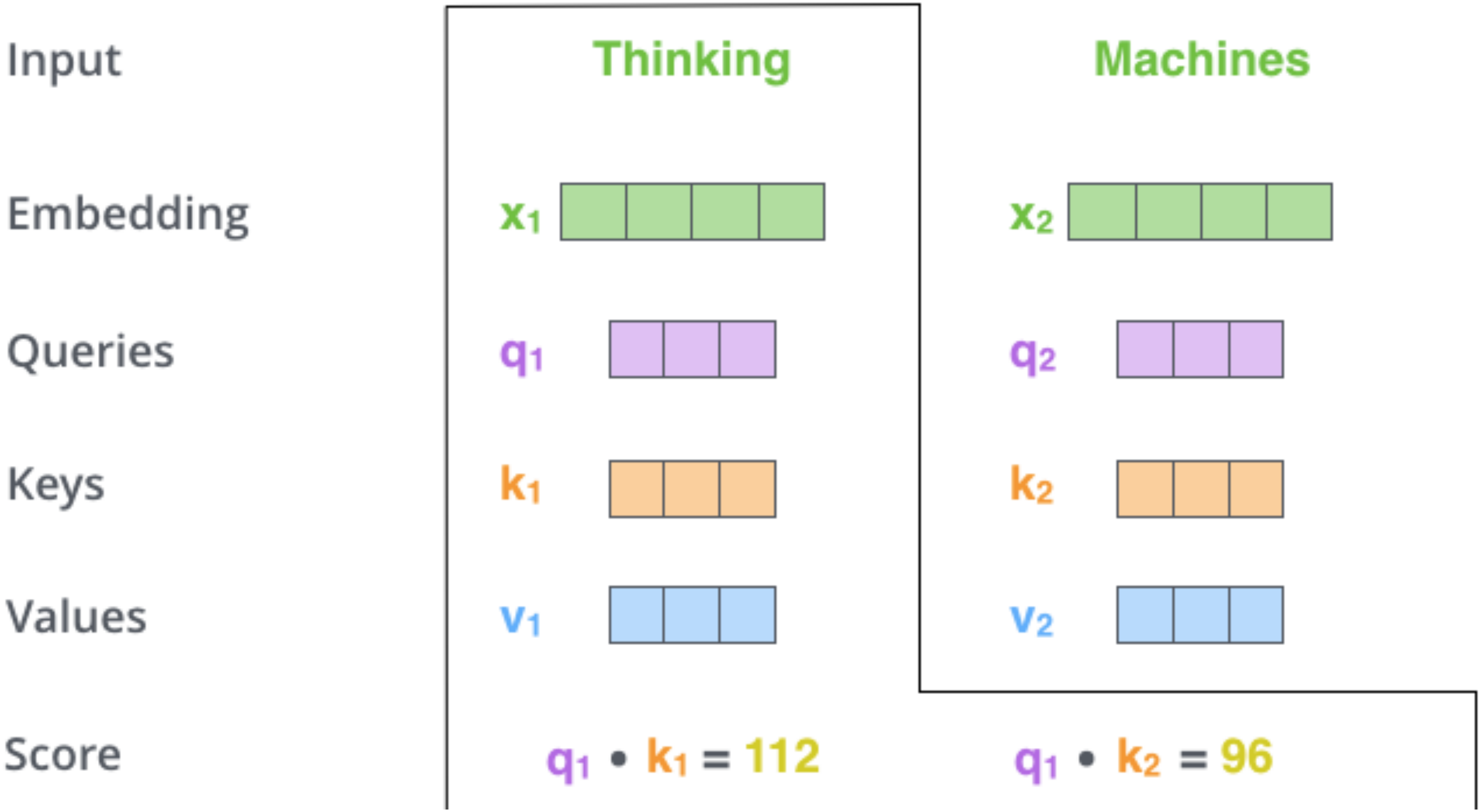
We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.



Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

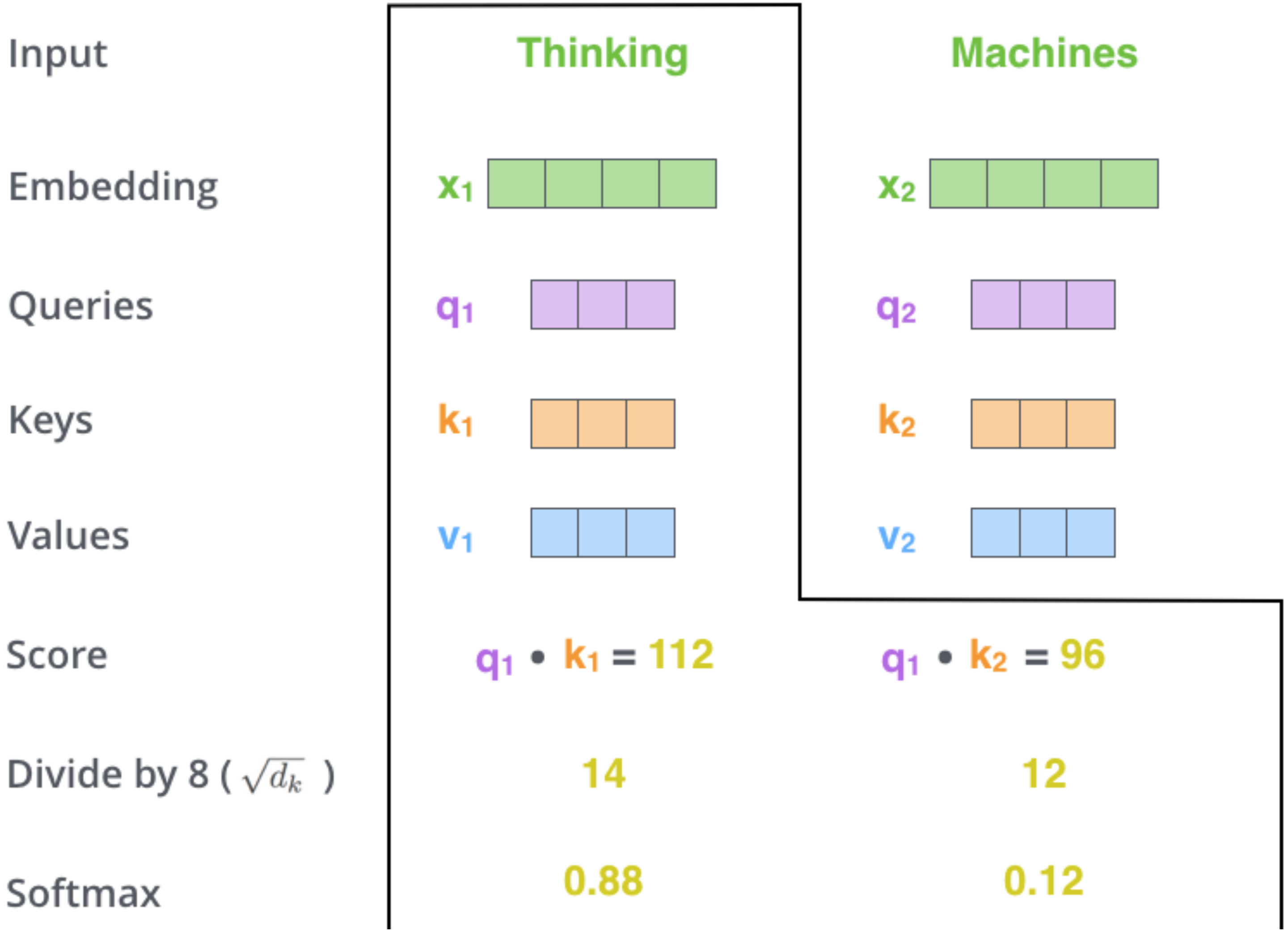
Jay Alammar (2018)



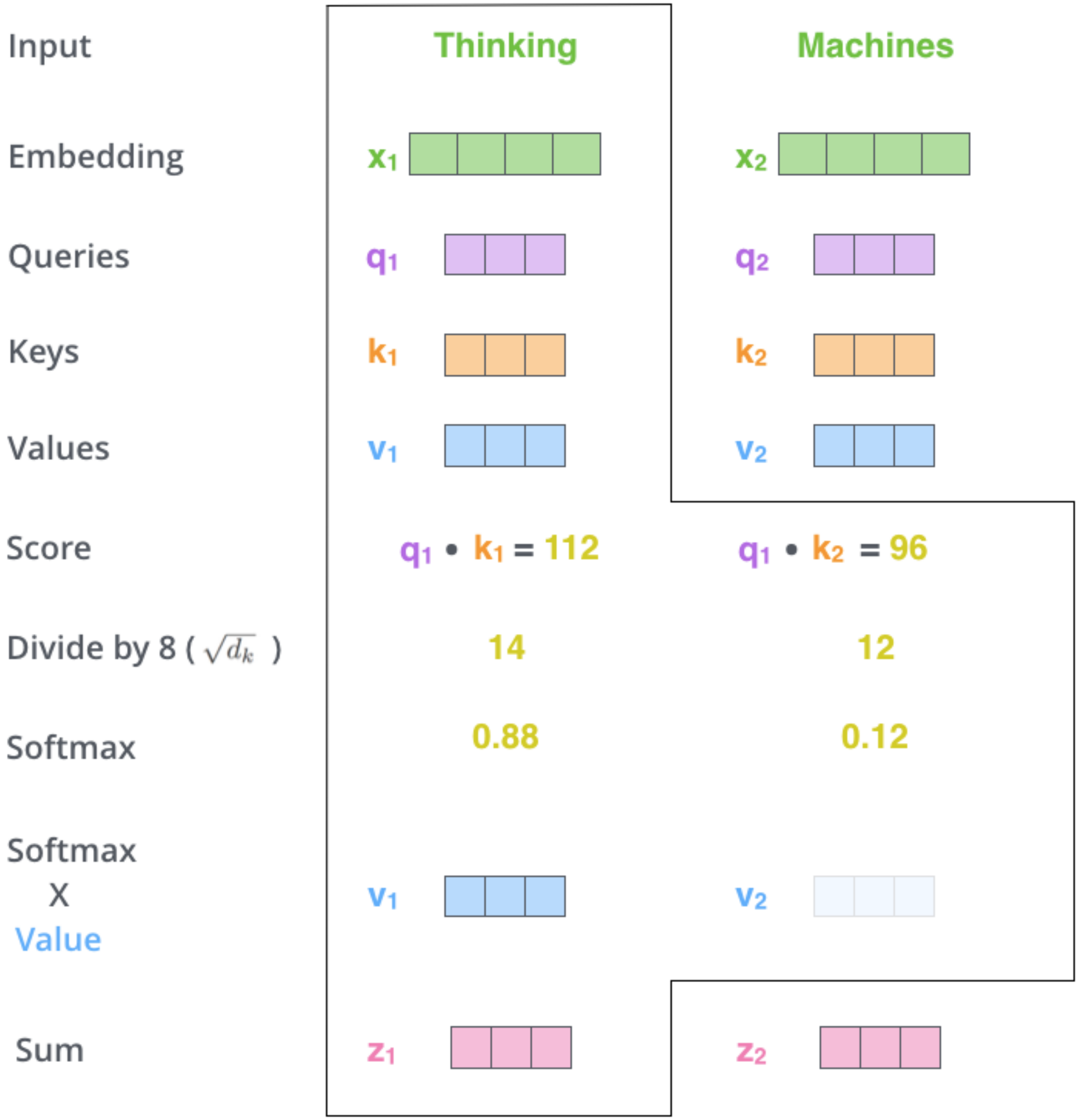
Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

Jay Alammar (2018)

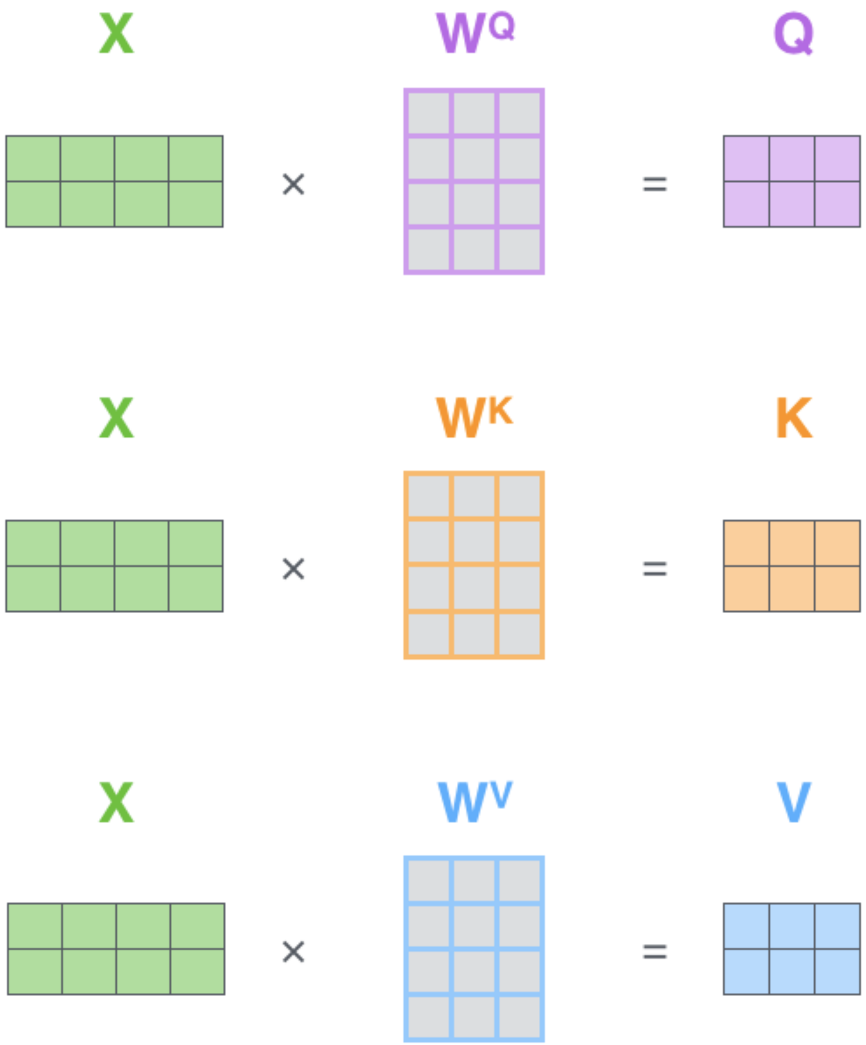


Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/



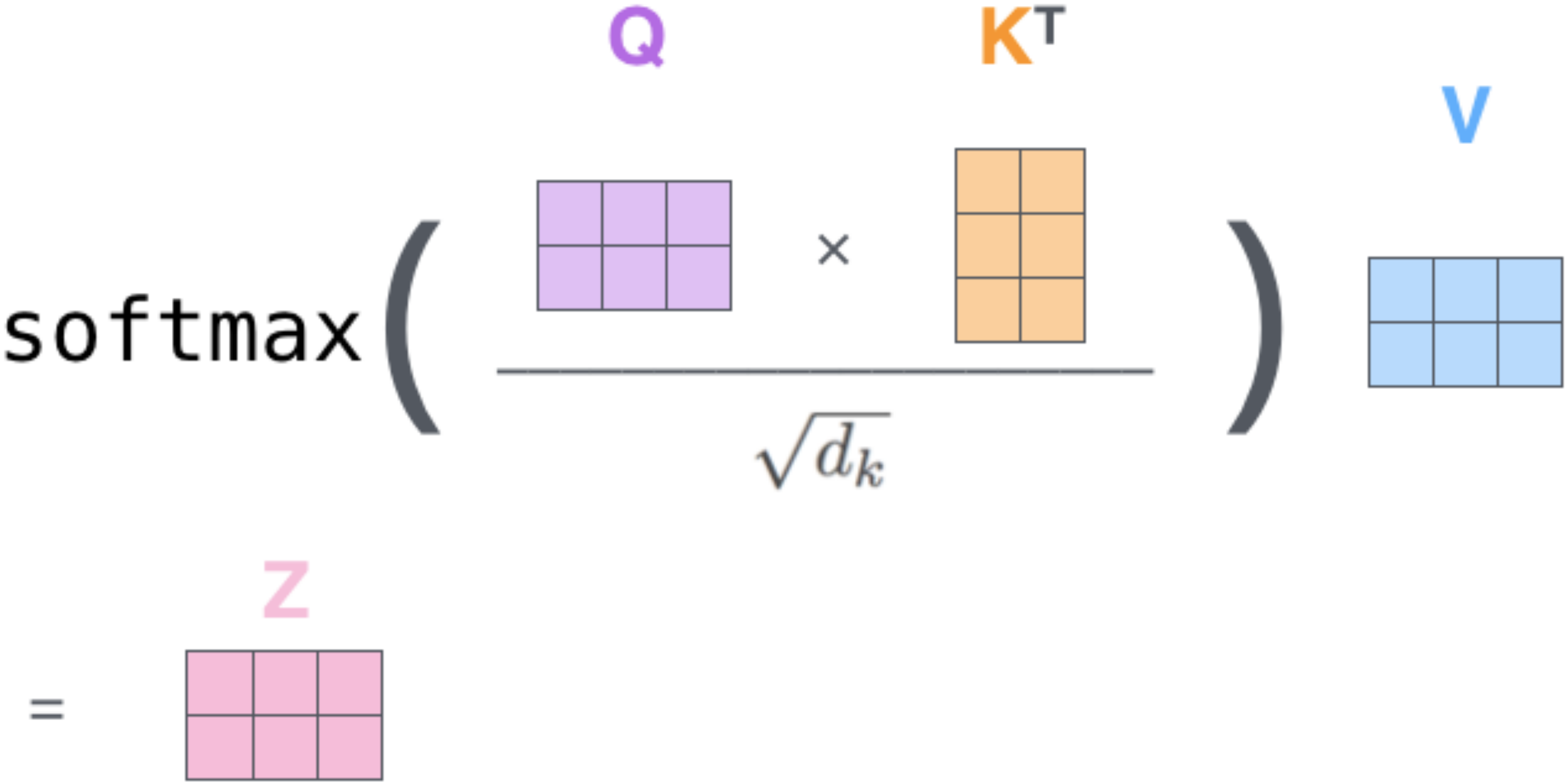
Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

Matrix Calculation of Self-Attention



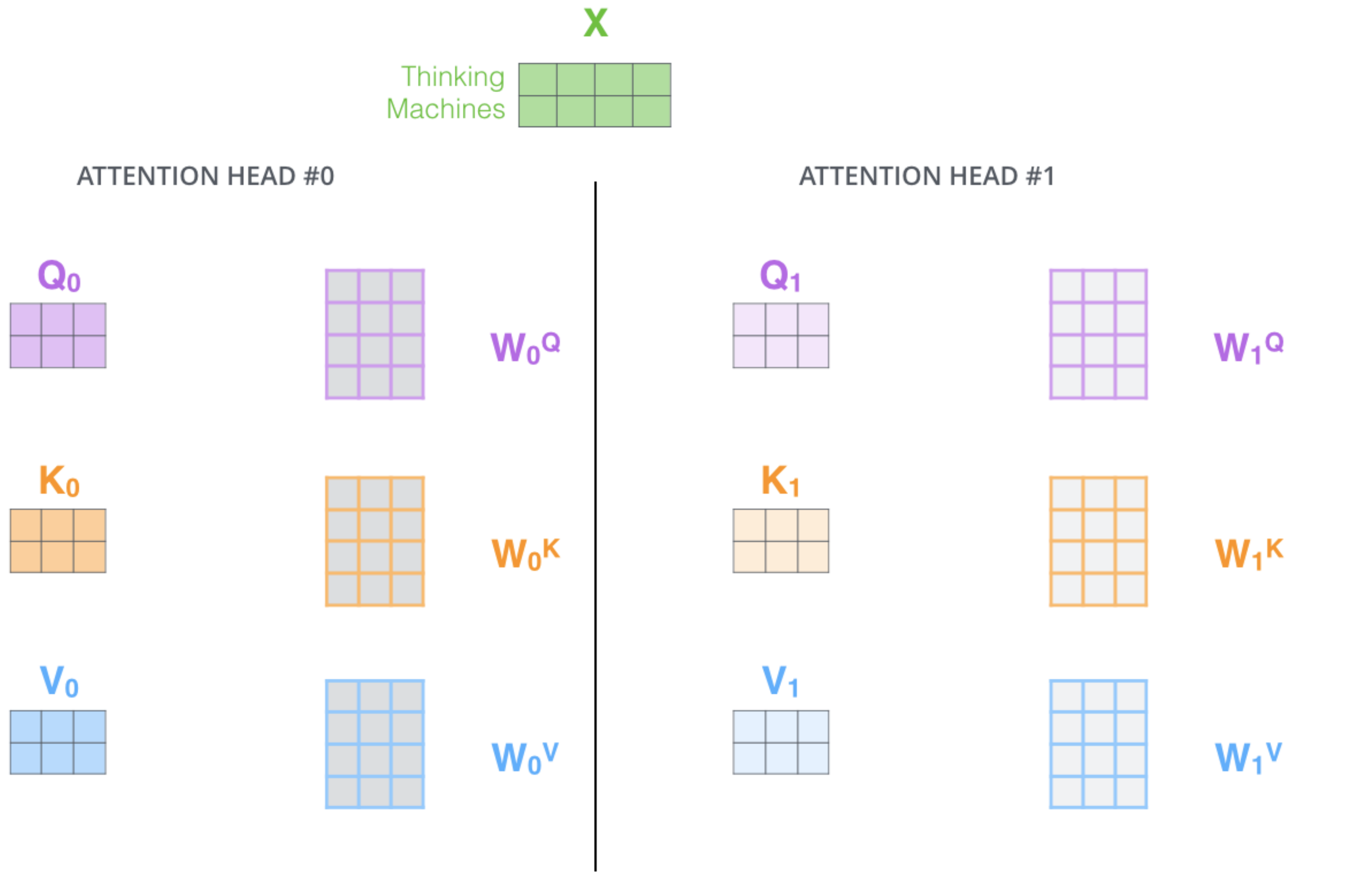
Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

The self-attention calculation in matrix form



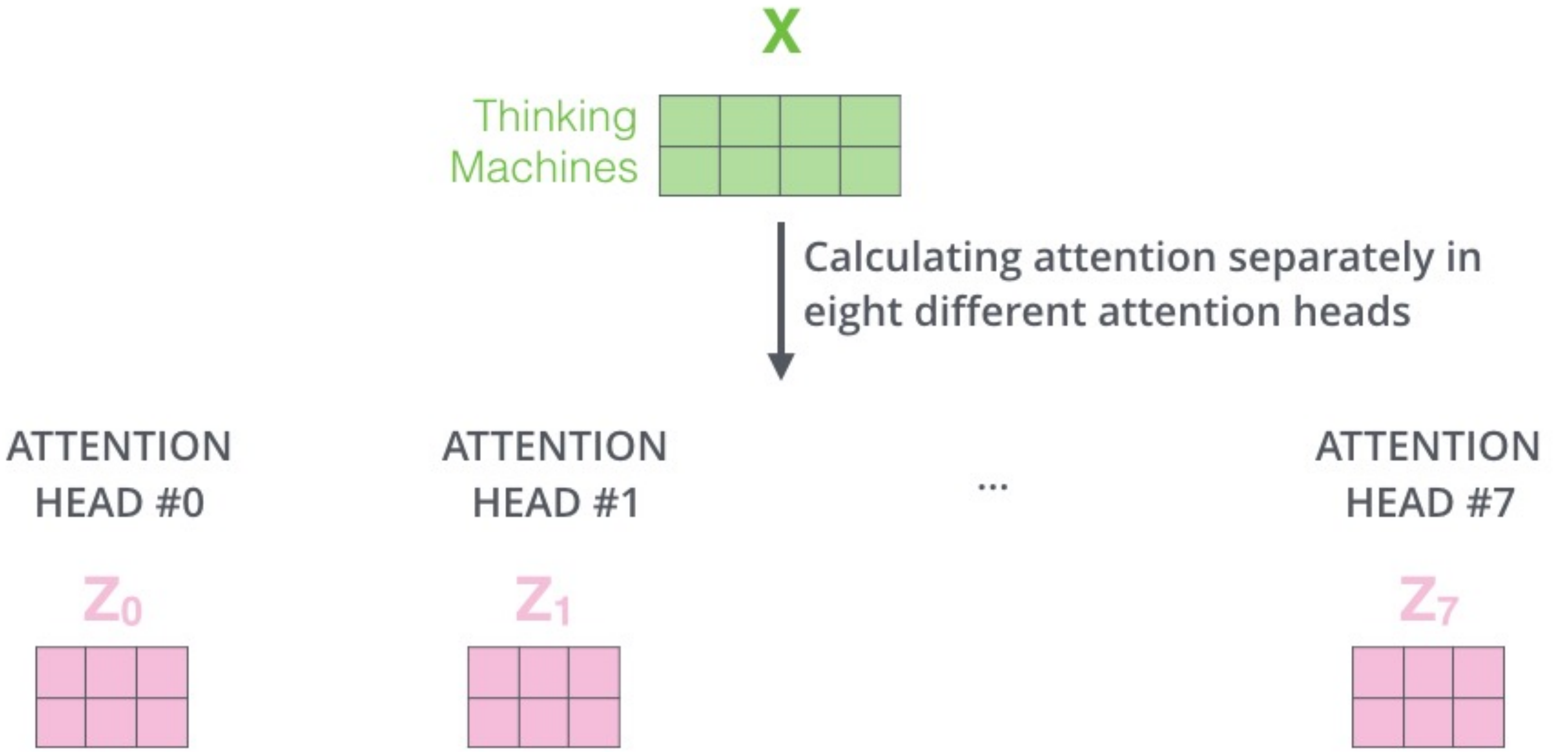
Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

Multi-headed Attention



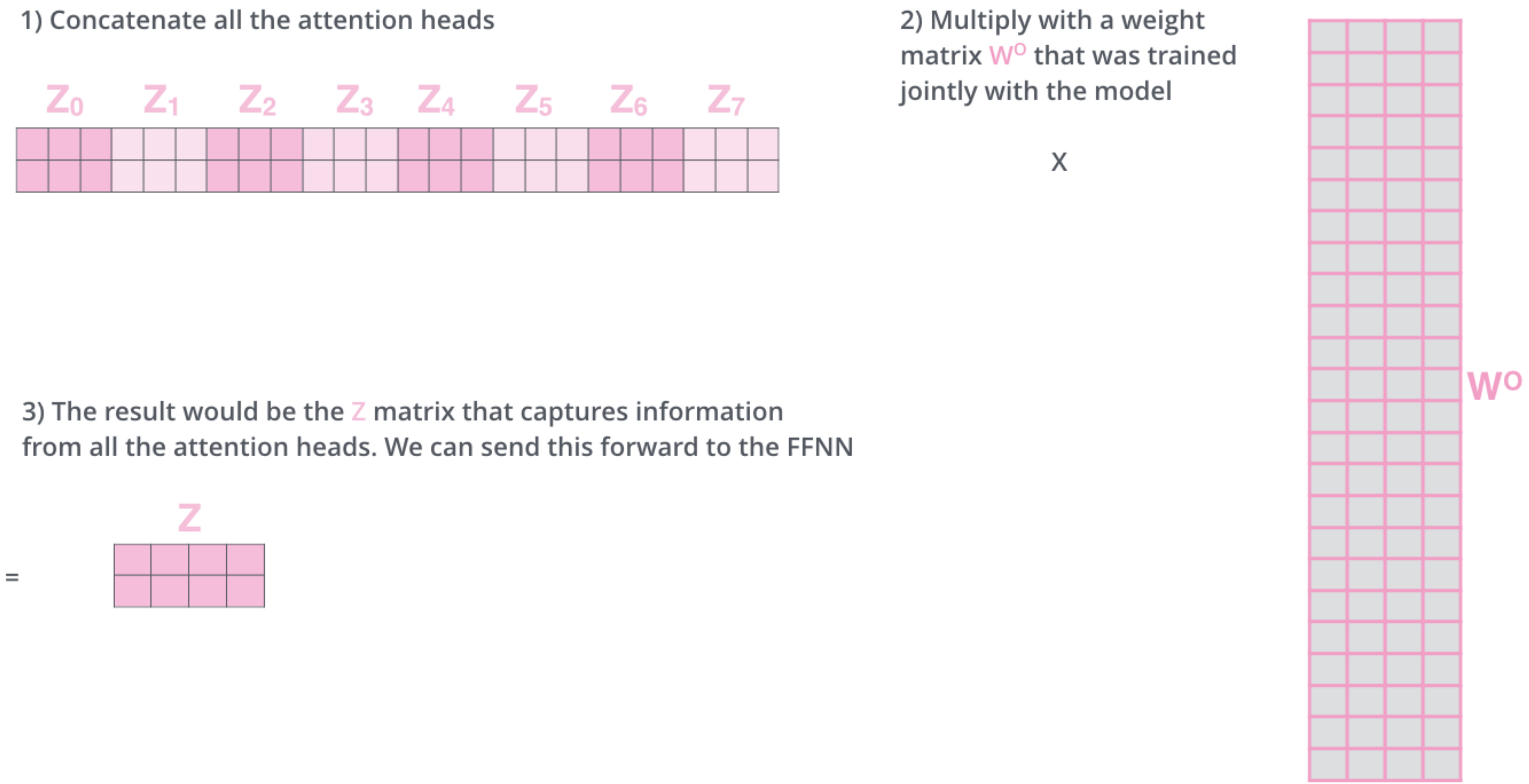
Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

Multi-headed Attention



Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

Multi-headed Attention



Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Multi-headed Attention

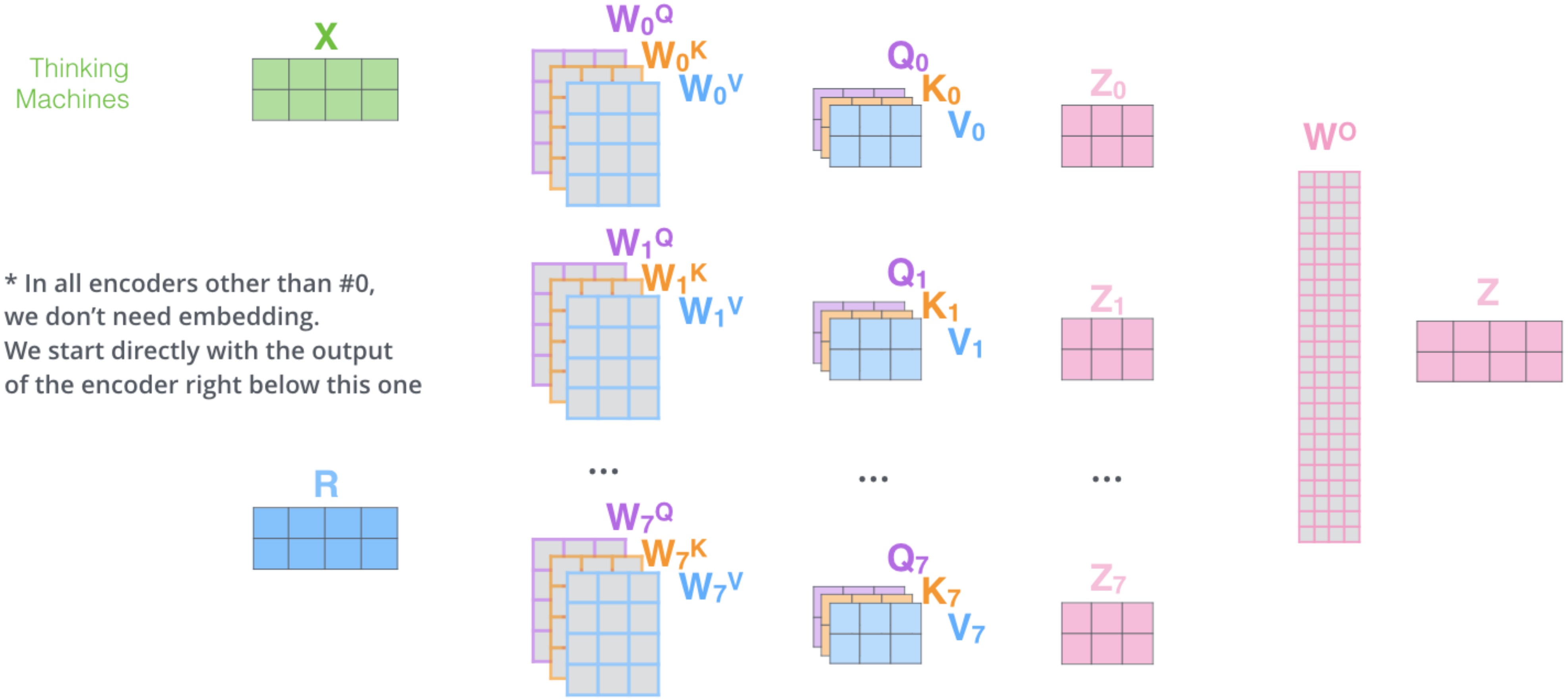
1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

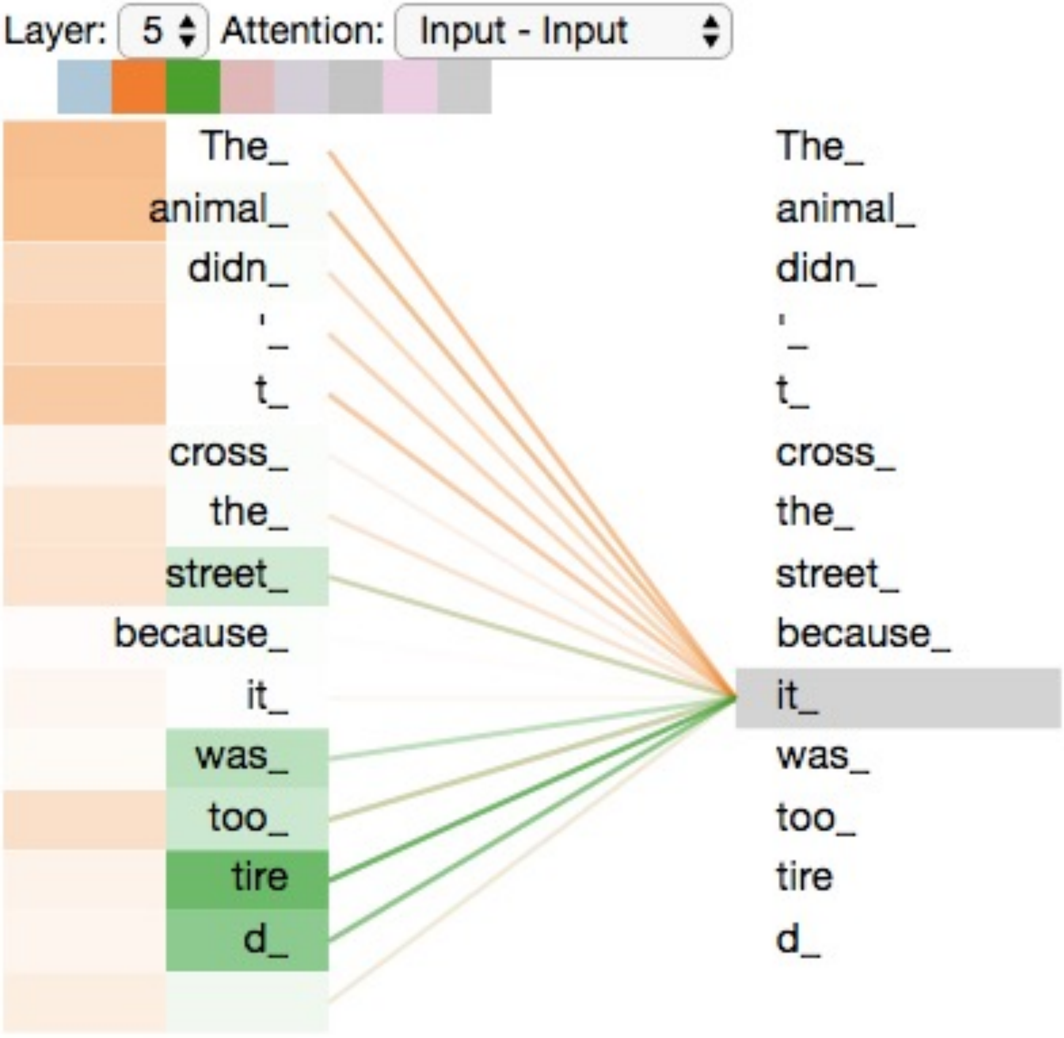
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

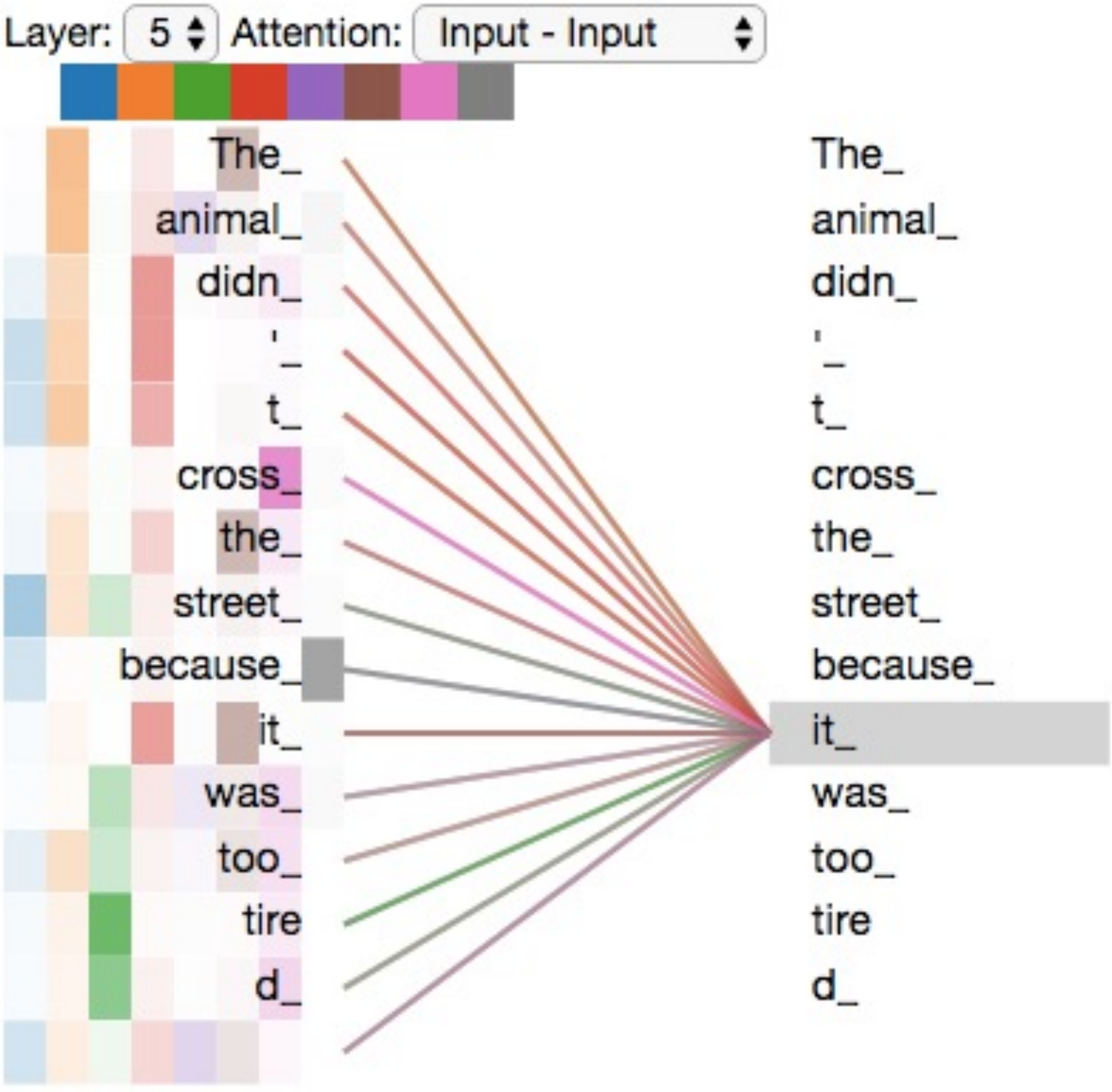
Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

As we encode the word "it", one attention head is focusing most on "the animal", while another is focusing on "tired" -- in a sense, the model's representation of the word "it" bakes in some of the representation of both "animal" and "tired".



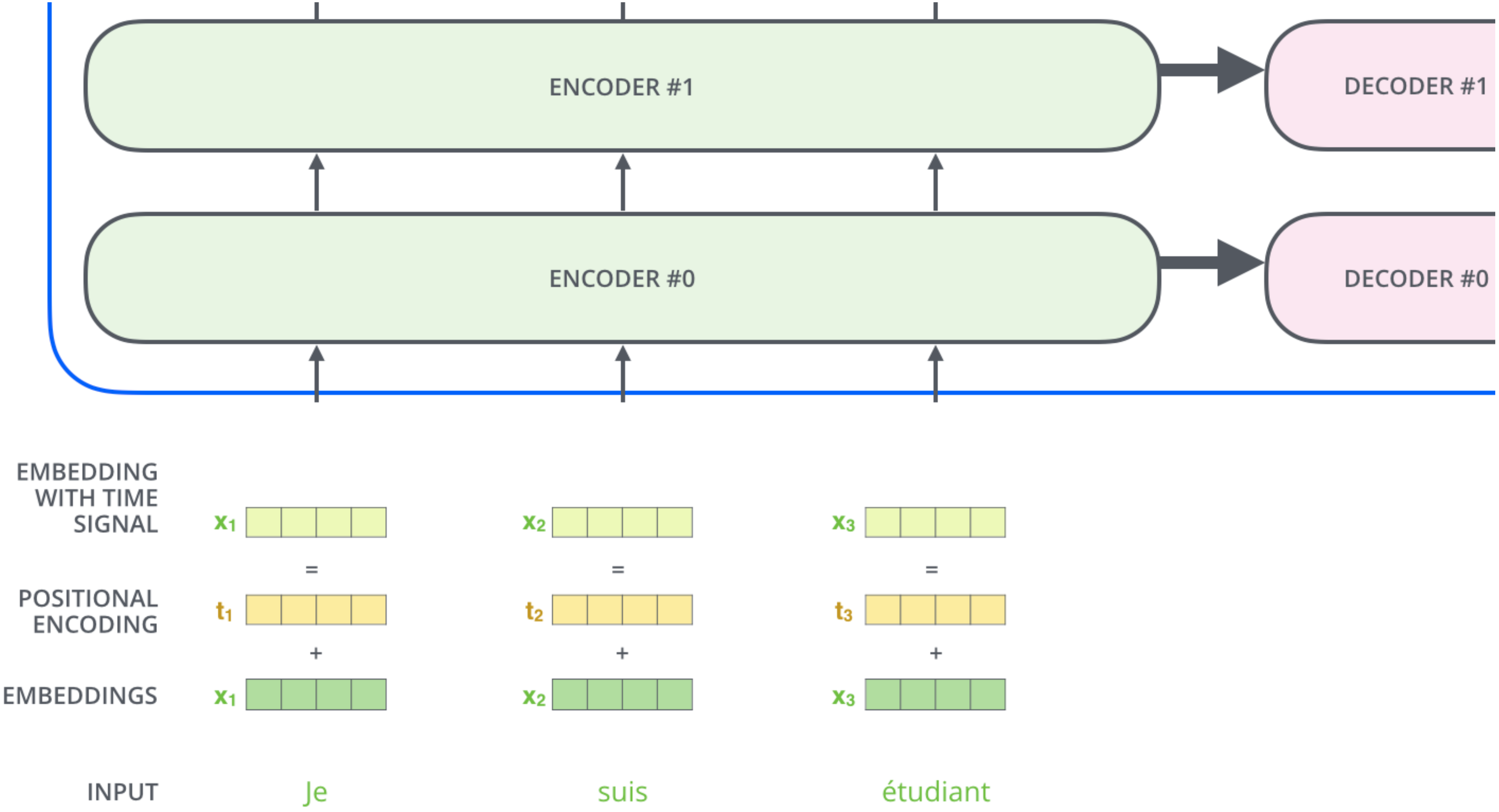
Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Add all the attention heads



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

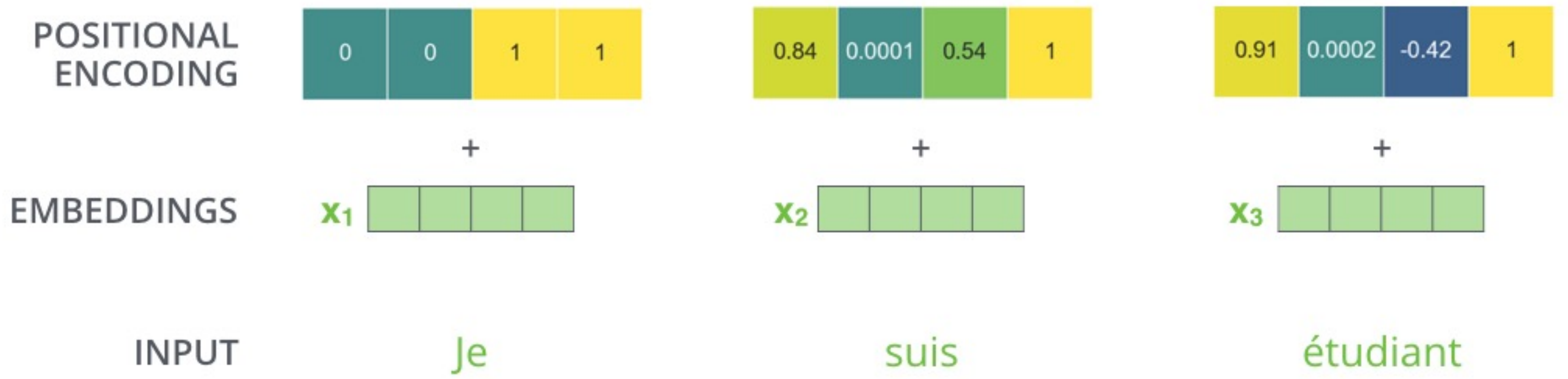
Positional Encoding



To give the model a sense of the order of the words, we add positional encoding vectors -- the values of which follow a specific pattern.

Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

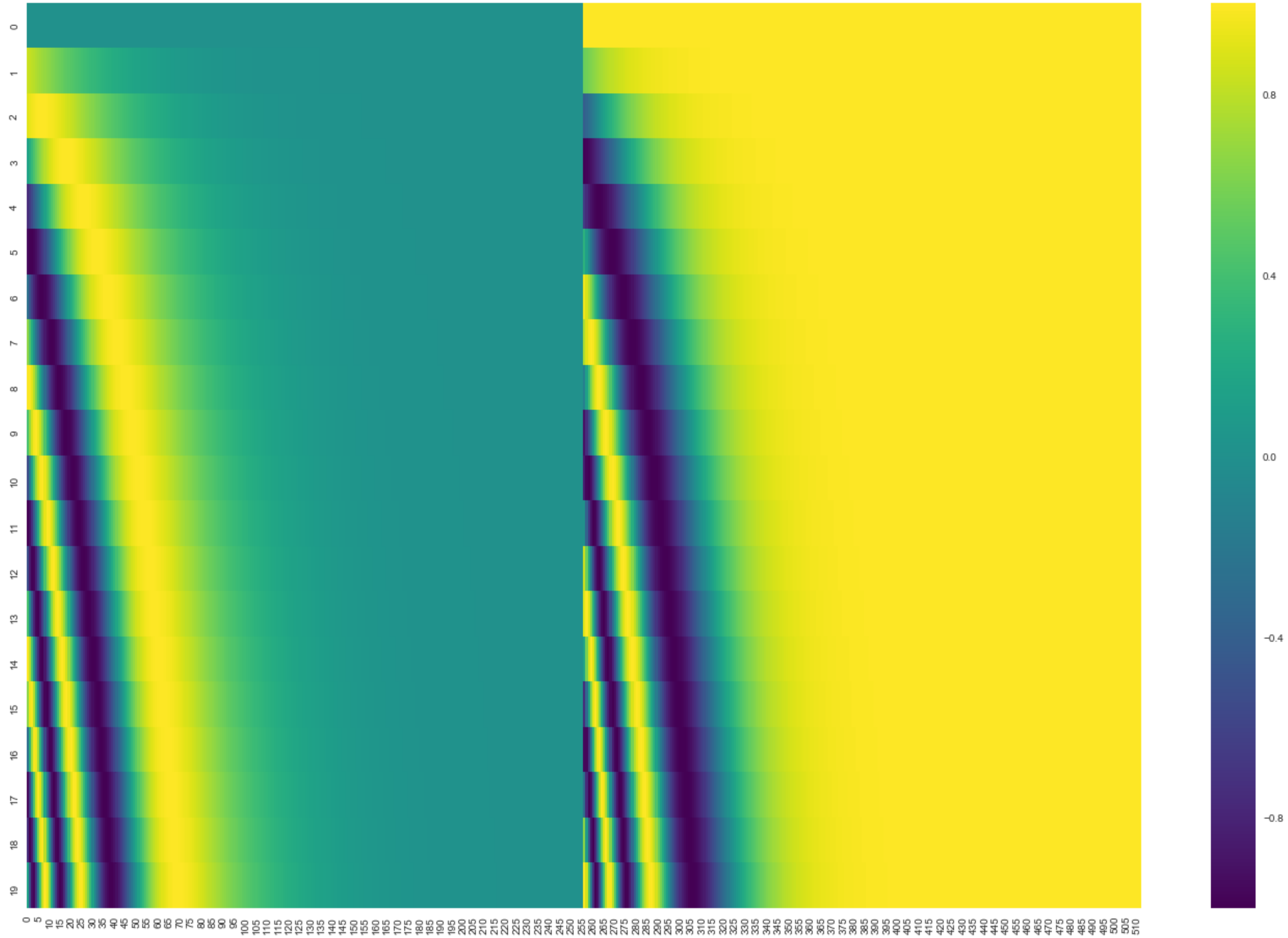
Positional Encoding



Positional encoding with a toy embedding size of 4

Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

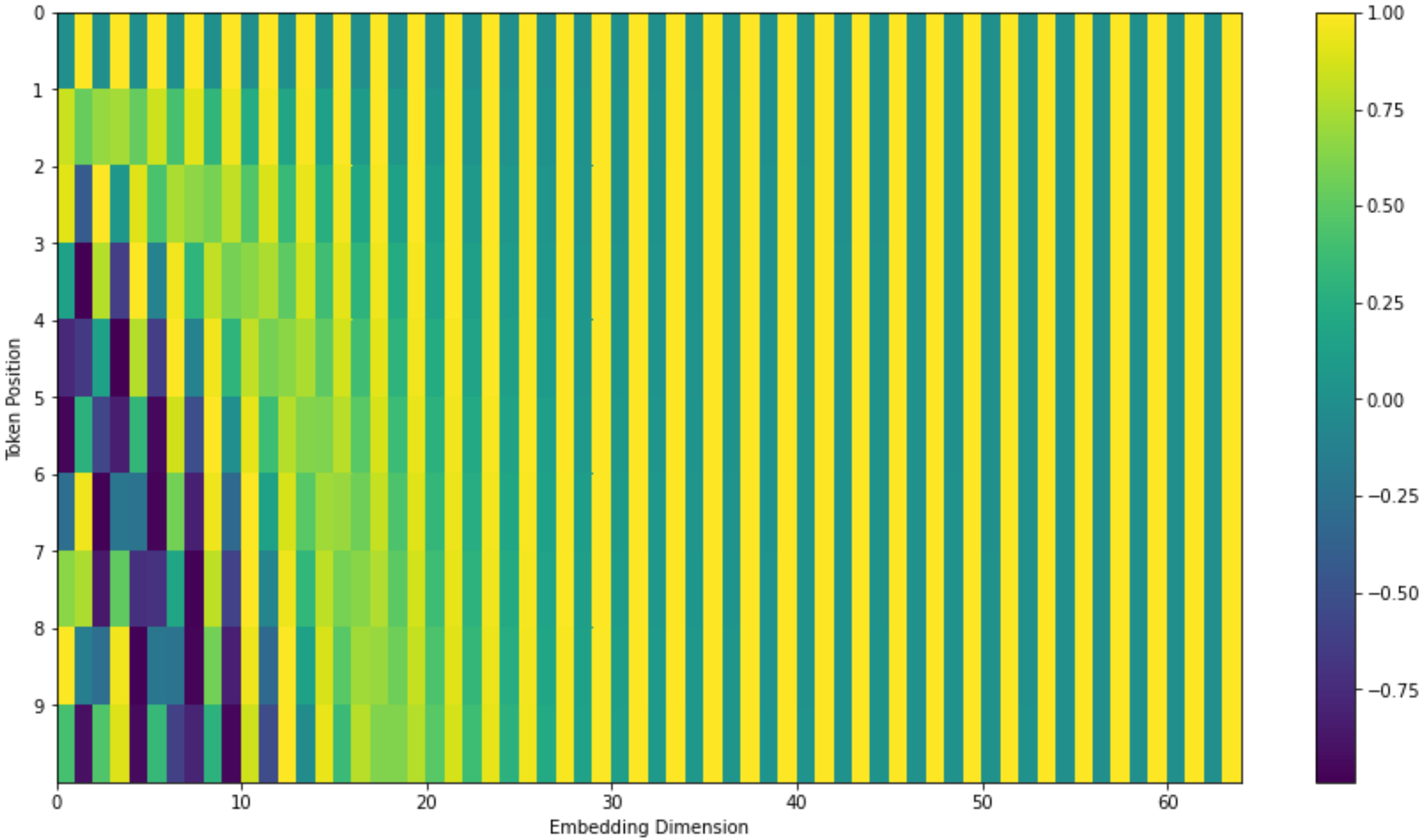
Positional encoding for 20 words (rows) with an embedding size of 512 (columns)



You can see that it appears split in half down the center. That's because the values of the left half are generated by one function (which uses sine), and the right half is generated by another function (which uses cosine). They're then concatenated to form each of the positional encoding vectors.

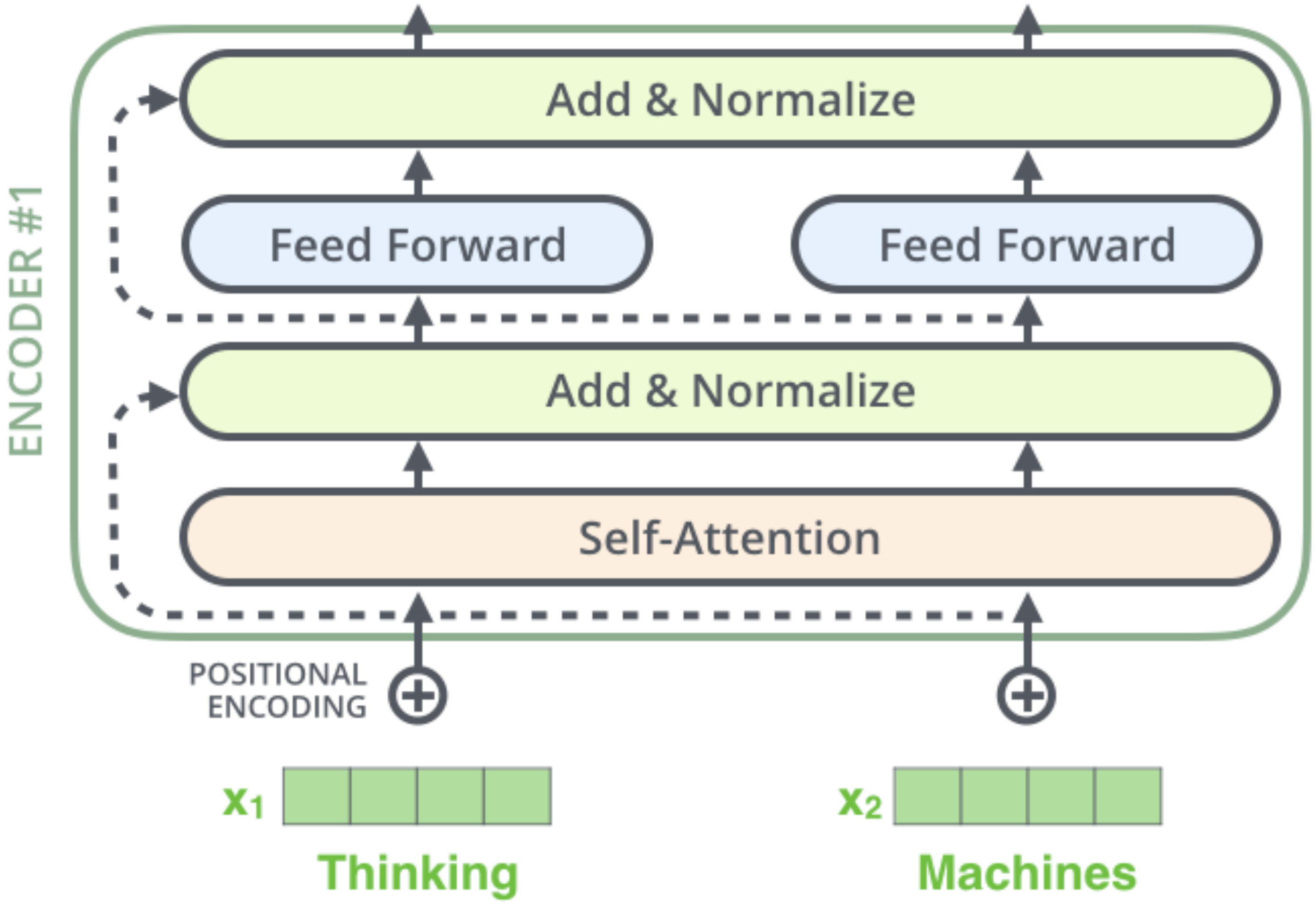
Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformers Positional Encoding

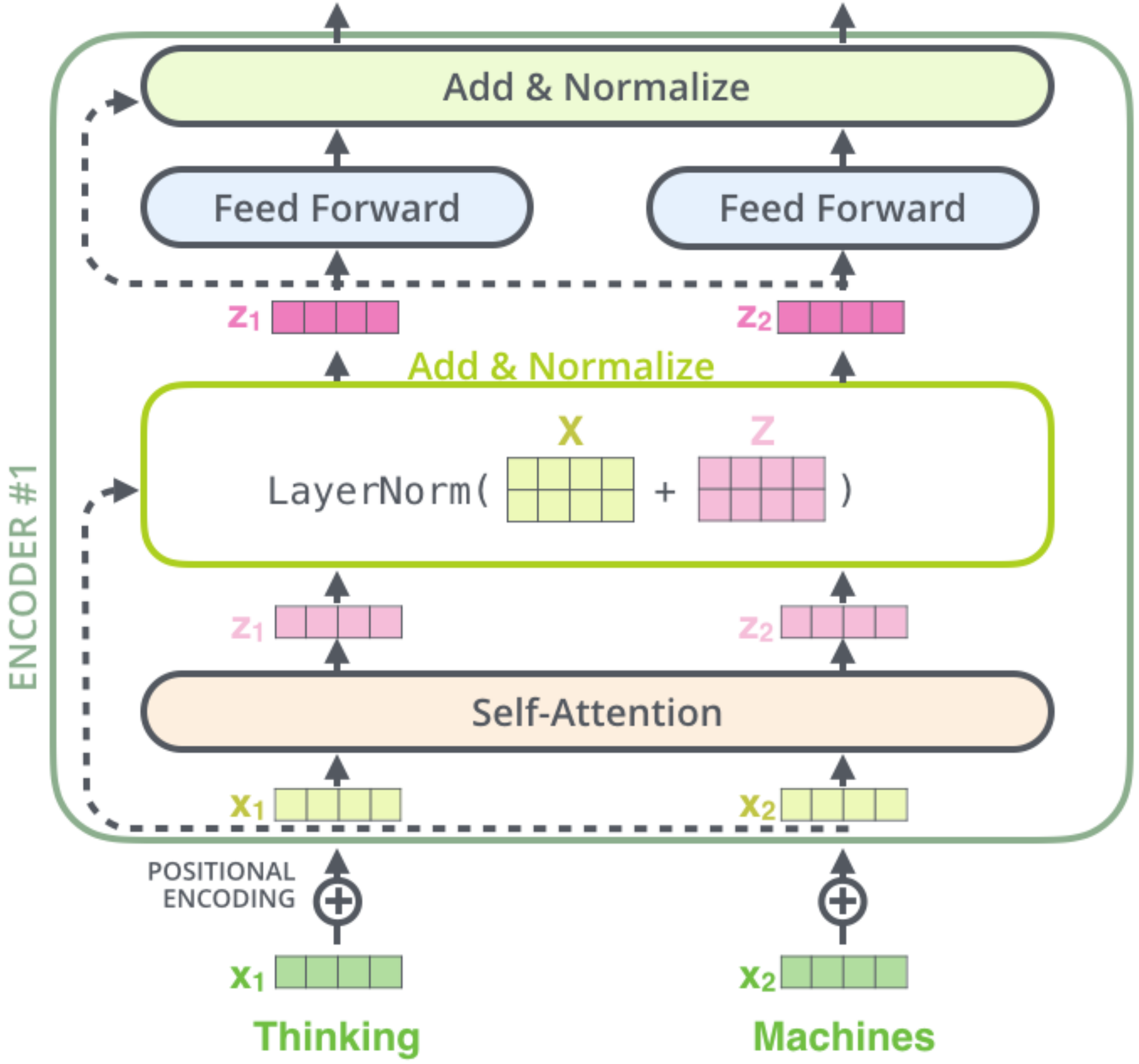


Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

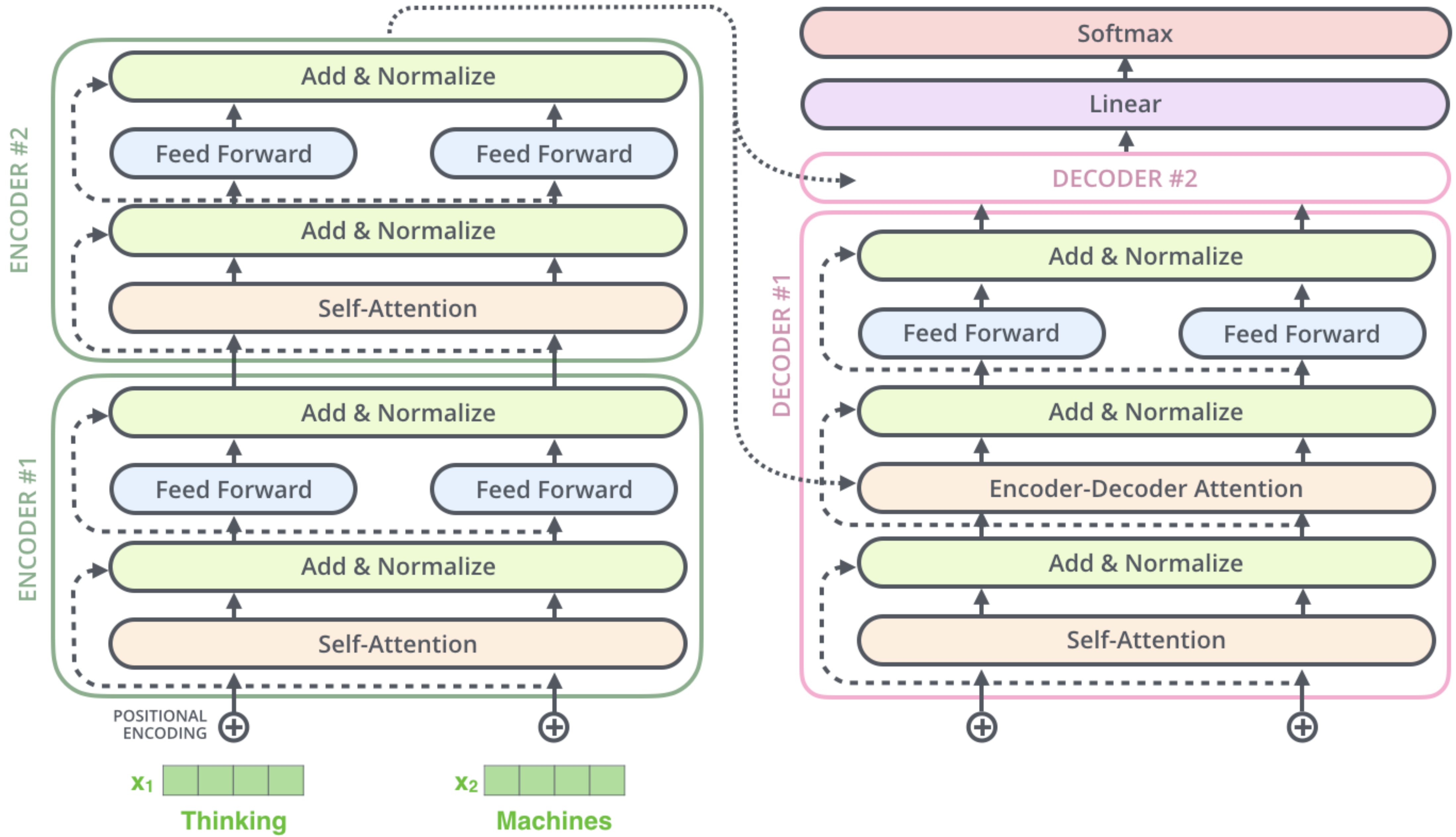
The Residuals



Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

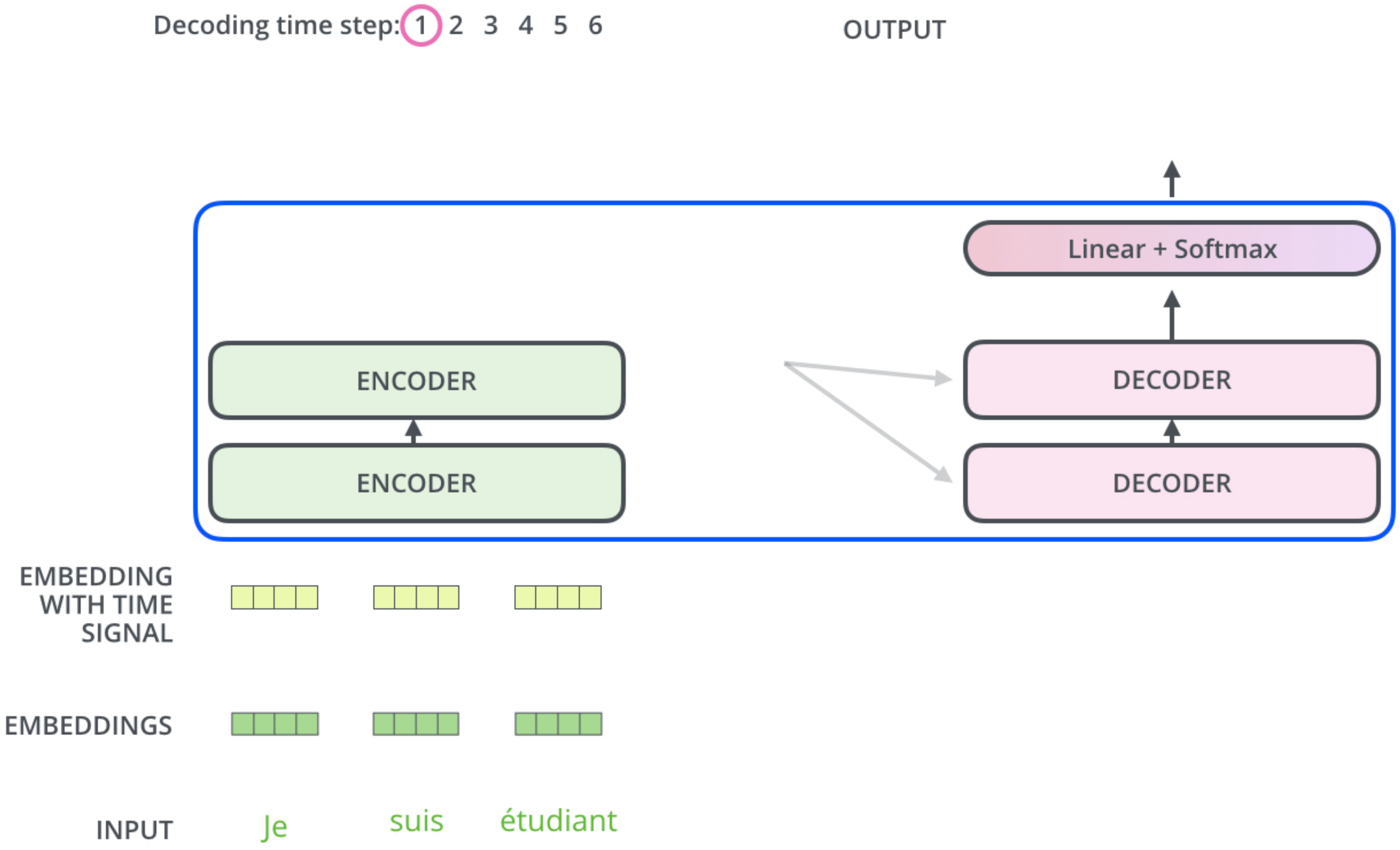


Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/



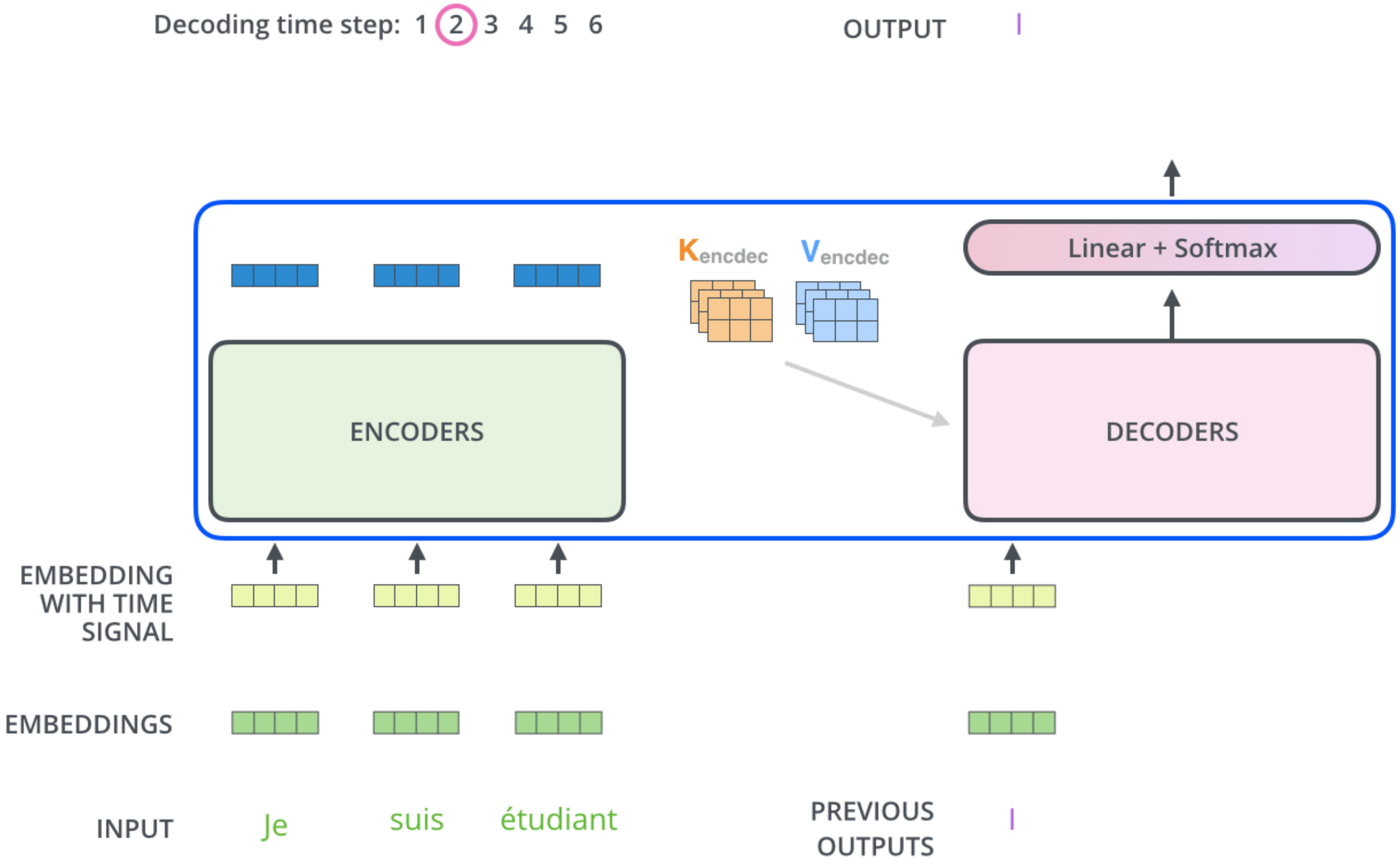
Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

The Decoder Side



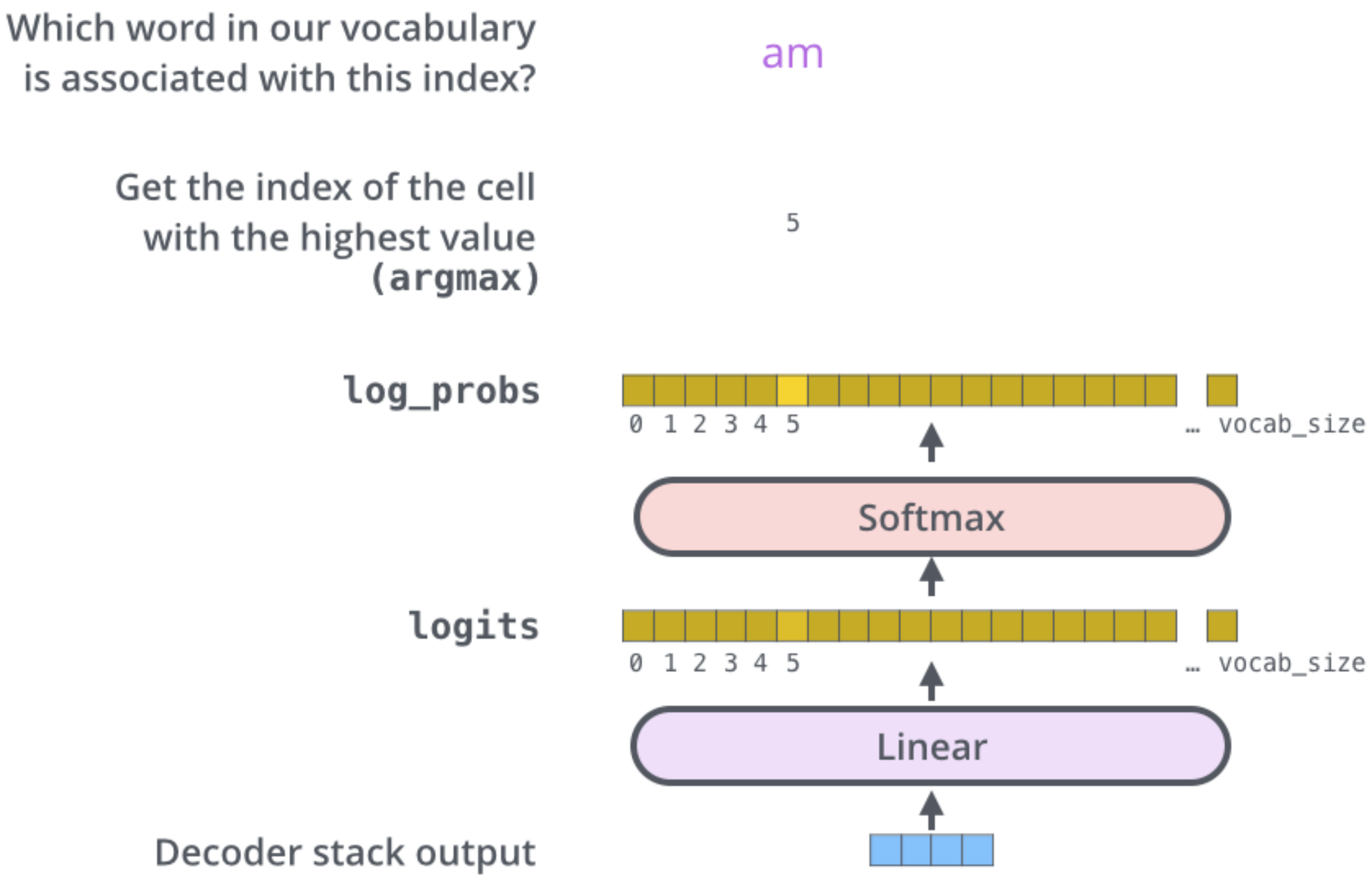
Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

The Decoder Side



Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

The Final Linear and Softmax Layer



Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

The output vocabulary

Output Vocabulary

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

The output vocabulary of our model is created in the preprocessing phase before we even begin training.

Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

Example: one-hot encoding of output vocabulary

Output Vocabulary

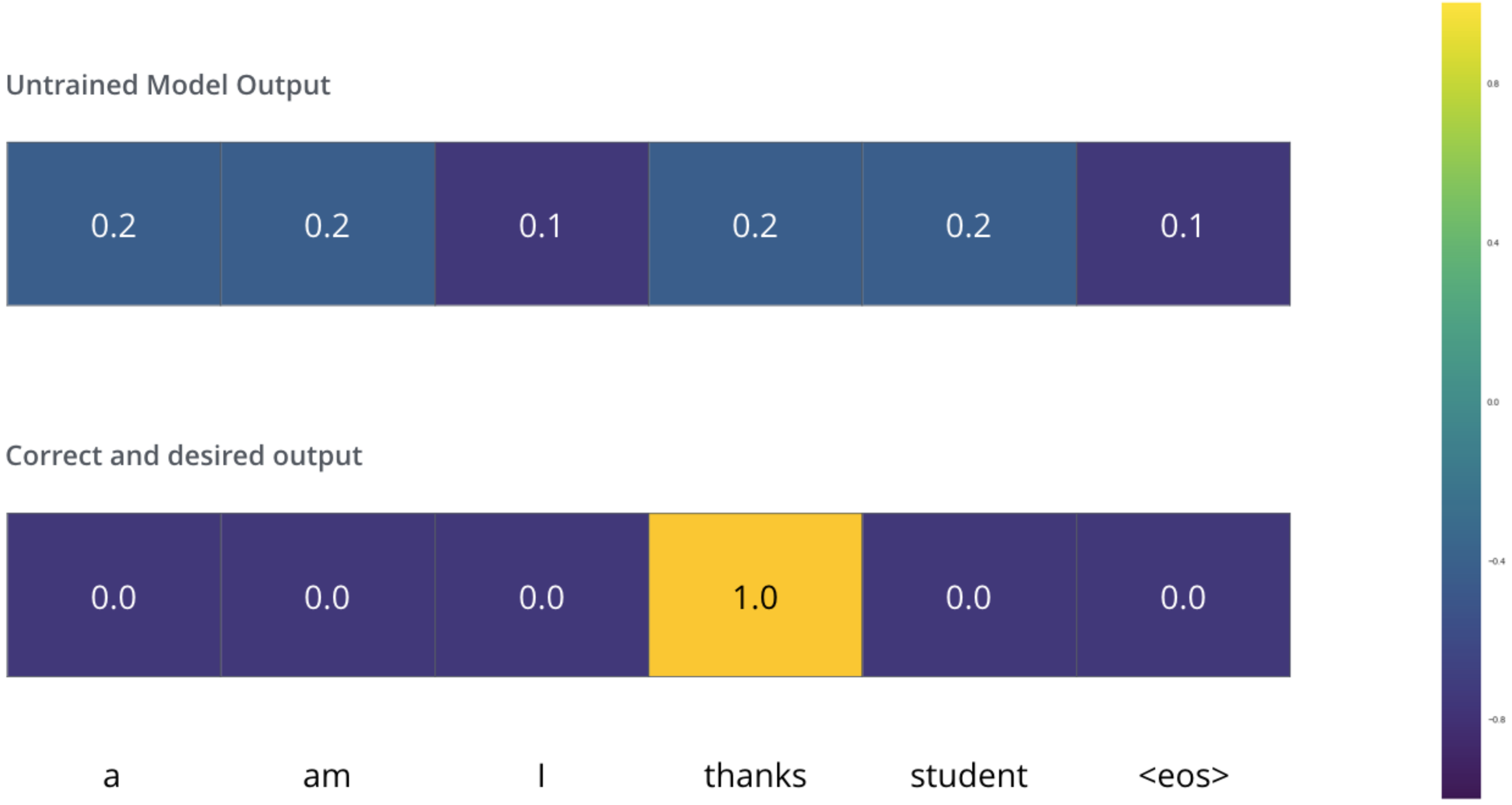
WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

One-hot encoding of the word “am”

0.0	1.0	0.0	0.0	0.0	0.0

Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

The Loss Function



Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

Target Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.0	0.0	1.0	0.0	0.0	0.0
position #2	0.0	1.0	0.0	0.0	0.0	0.0
position #3	1.0	0.0	0.0	0.0	0.0	0.0
position #4	0.0	0.0	0.0	0.0	1.0	0.0
position #5	0.0	0.0	0.0	0.0	0.0	1.0
	a	am	I	thanks	student	<eos>

Source: Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Trained Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001
position #4	0.001	0.002	0.001	0.02	0.94	0.01
position #5	0.01	0.01	0.001	0.001	0.001	0.98
	a	am	I	thanks	student	<eos>

Source: Jay Alammar (2018), The Illustrated Transformer,
http://jalammar.github.io/illustrated-transformer/

Transformers

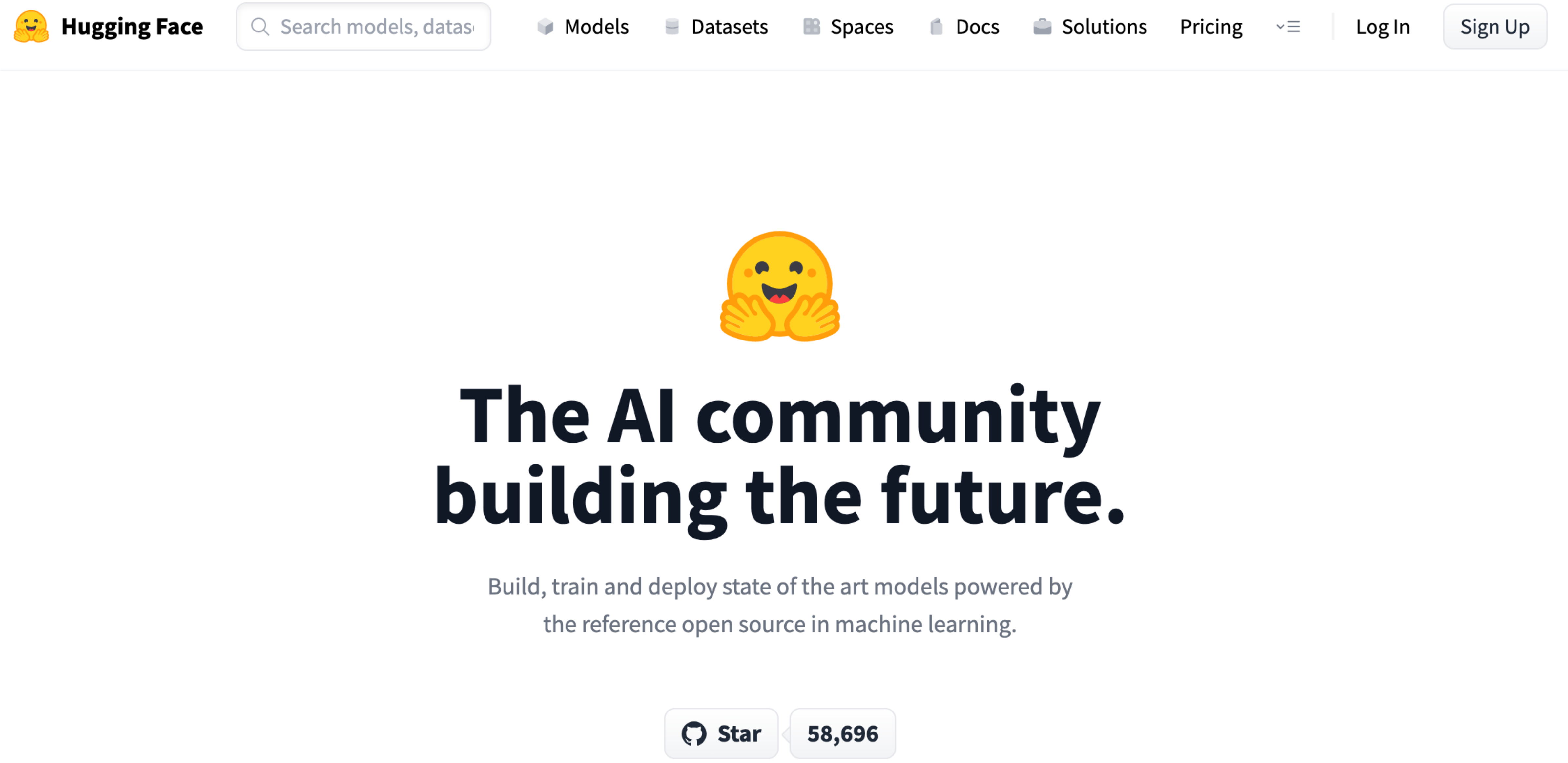
Transformers

State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch

- Transformers
  - pytorch-transformers
  - pytorch-pretrained-bert
- provides state-of-the-art general-purpose architectures
  - (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL...)
  - for Natural Language Understanding (NLU) and
    Natural Language Generation (NLG)
    with over 32+ pretrained models
    in 100+ languages
    and deep interoperability between
    TensorFlow 2.0 and
    PyTorch.

Source: https://github.com/huggingface/transformers

Hugging Face



https://huggingface.co/

Hugging Face Transformers

🤗 **Hugging Face** Search models, datasets, users... Models Datasets Spaces Docs Solutions Pricing Log In Sign Up

- **Transformers**

Search documentation ⌘K

V4.16.2 EN 58,697

GET STARTED

- 🤗 Transformers
- Quick tour
- Installation
- Philosophy
- Glossary

USING 🤗 TRANSFORMERS

- Summary of the tasks
- Summary of the models
- Preprocessing data
- Fine-tuning a pretrained model
- Distributed training with 🤗 Accelerate

🤗 Transformers

State-of-the-art Machine Learning for Jax, Pytorch and TensorFlow

🤗 Transformers (formerly known as *pytorch-transformers* and *pytorch-pretrained-bert*) provides thousands of pretrained models to perform tasks on different modalities such as text, vision, and audio.

These models can applied on:

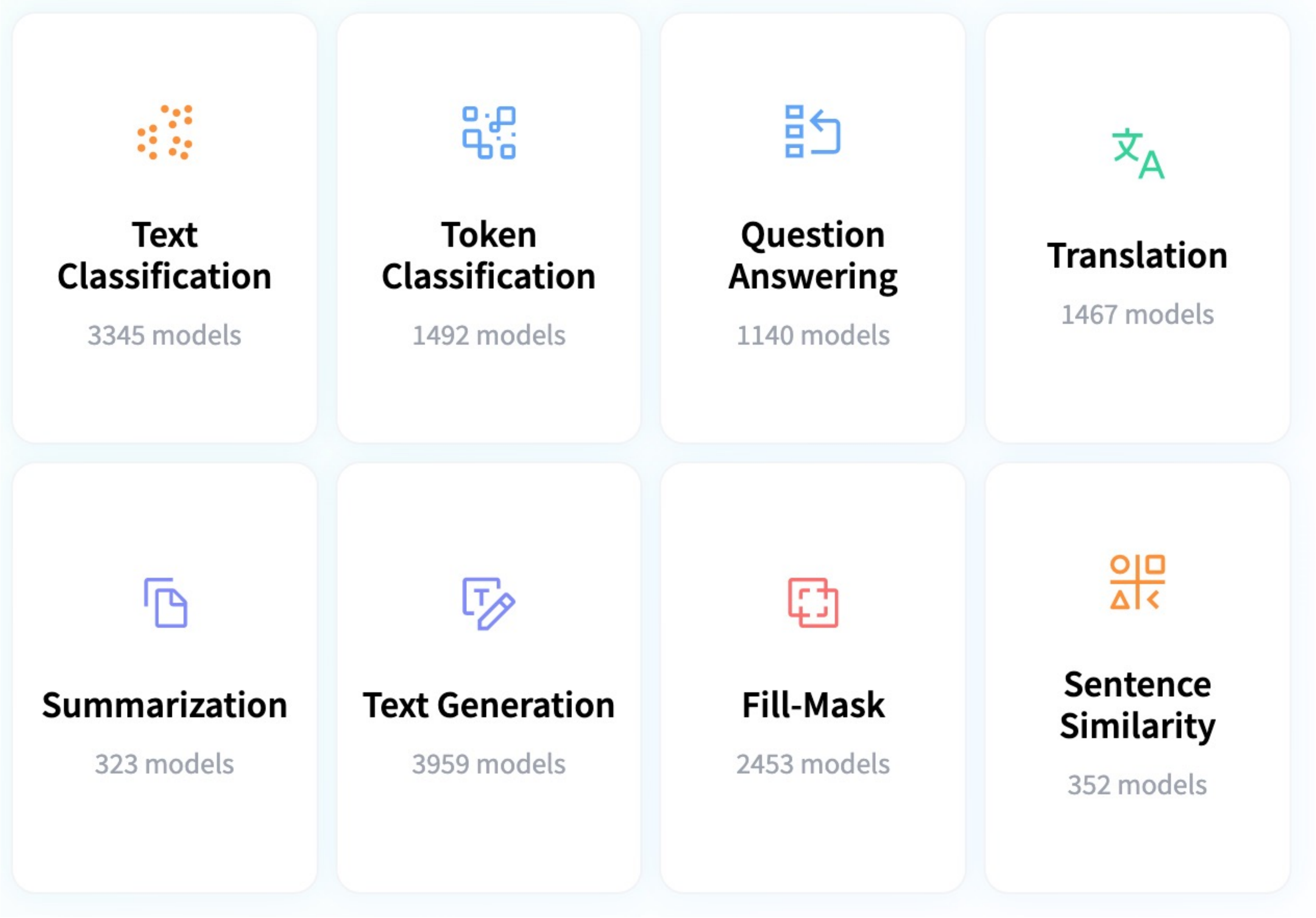
- 📝 Text, for tasks like text classification, information extraction, question answering, summarization, translation, text generation, in over 100 languages.
- 🖼️ Images, for tasks like image classification, object detection, and segmentation.
- 🗣️ Audio, for tasks like speech recognition and audio classification.

Transformer models can also perform tasks on **several modalities combined**, such as table question answering, optical character recognition, information extraction from scanned documents, video classification, and visual question answering.

- 🤗 Transformers
  - If you are looking for custom support from the Hugging Face team
  - Features
  - Contents
    - Supported models
    - Supported frameworks

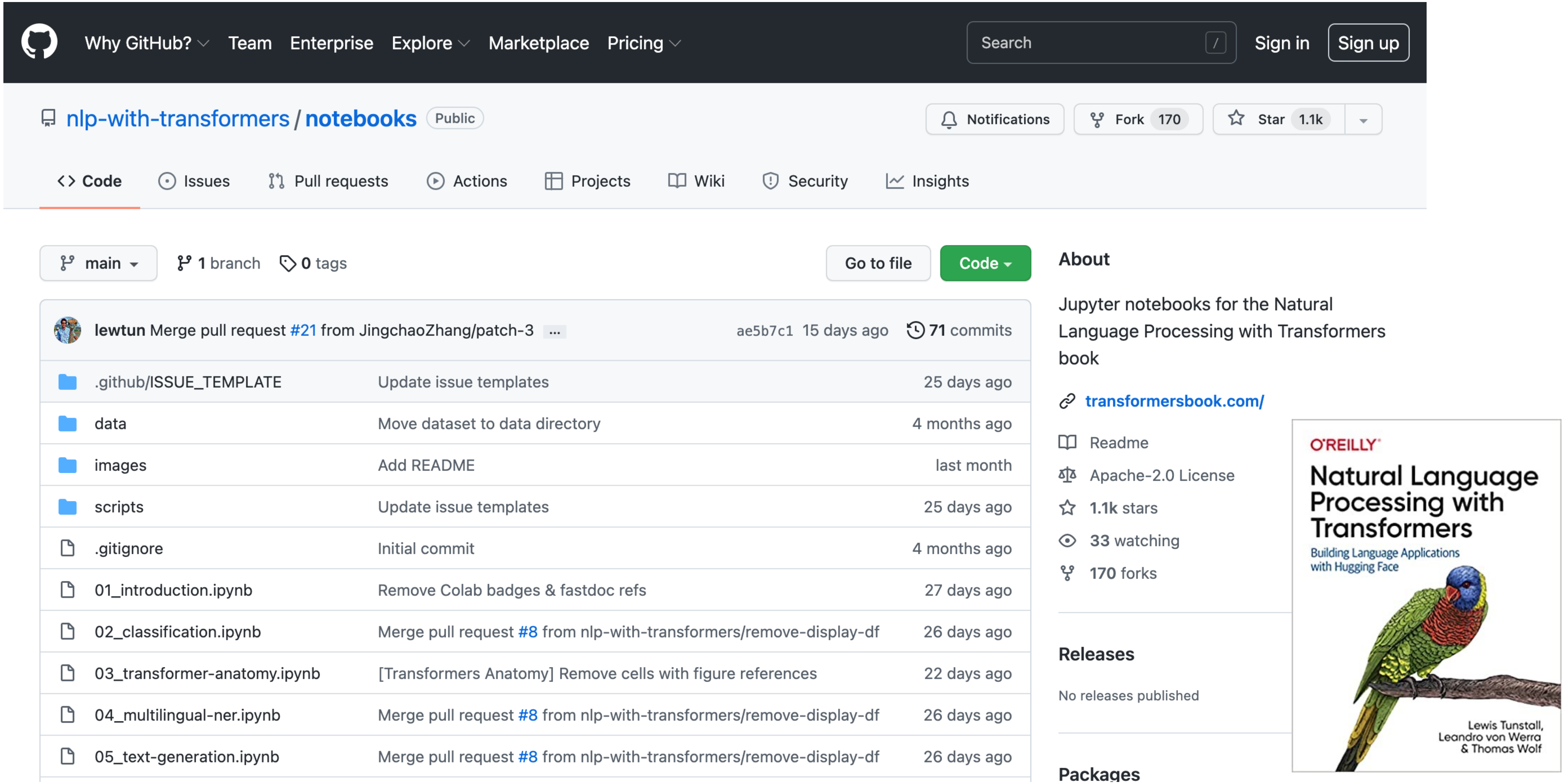
https://huggingface.co/docs/transformers/index

Hugging Face Tasks
Natural Language Processing



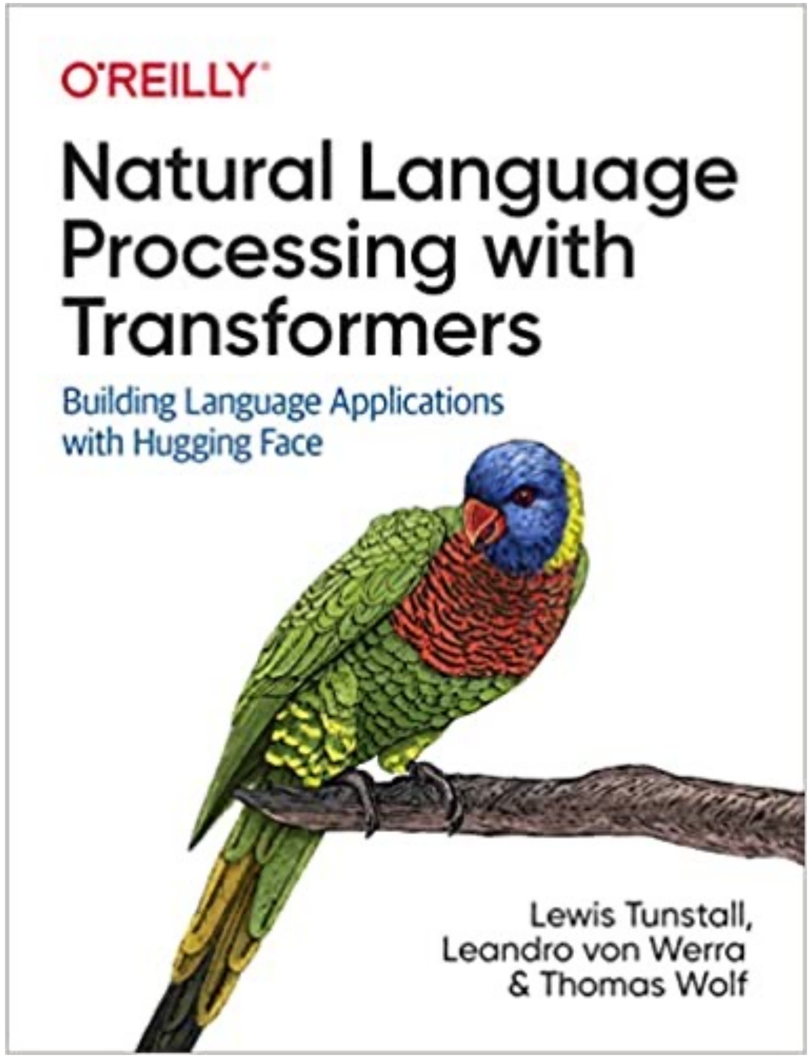
https://huggingface.co/tasks

NLP with Transformers Github



https://github.com/nlp-with-transformers/notebooks

NLP with Transformers Github Notebooks



Running on a cloud platform

To run these notebooks on a cloud platform, just click on one of the badges in the table below:

Chapter	Colab	Kaggle	Gradient	Studio Lab
Introduction	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Classification	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Transformer Anatomy	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Multilingual Named Entity Recognition	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Generation	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Summarization	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Question Answering	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Making Transformers Efficient in Production	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Dealing with Few to No Labels	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Training Transformers from Scratch	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Future Directions	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab

Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using Kaggle, Gradient, or SageMaker Studio Lab. These platforms tend to provide more performant GPUs like P100s, all for free!

https://github.com/nlp-with-transformers/notebooks

NLP with Transformers

```
!git clone https://github.com/nlp-with-transformers/notebooks.git
%cd notebooks
from install import *
install_requirements()
```

```
from utils import *
setup_chapter()
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Text Classification

```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Text Classification

```python
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

```python
from transformers import pipeline
classifier = pipeline("text-classification")
```

```python
import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

	label	score
0	NEGATIVE	0.901546

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Text Classification

```
from transformers import pipeline
classifier = pipeline("text-classification")
```

```
import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

	label	score
0	NEGATIVE	0.901546

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Named Entity Recognition

```python
ner_tagger = pipeline("ner", aggregation_strategy="simple")
outputs = ner_tagger(text)
pd.DataFrame(outputs)
```

	entity_group	score	word	start	end
0	ORG	0.879010	Amazon	5	11
1	MISC	0.990859	Optimus Prime	36	49
2	LOC	0.999755	Germany	90	97
3	MISC	0.556570	Mega	208	212
4	PER	0.590256	##tron	212	216
5	ORG	0.669692	Decept	253	259
6	MISC	0.498349	##icons	259	264
7	MISC	0.775362	Megatron	350	358
8	MISC	0.987854	Optimus Prime	367	380
9	PER	0.812096	Bumblebee	502	511

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Question Answering

```
reader = pipeline("question-answering")
question = "What does the customer want?"
outputs = reader(question=question, context=text)
pd.DataFrame([outputs])
```

	score	start	end	answer
0	0.631292	335	358	an exchange of Megatron

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Summarization

```
summarizer = pipeline("summarization")
outputs = summarizer(text, max_length=45, clean_up_tokenization_spaces=True)
print(outputs[0]['summary_text'])
```

Bumblebee ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead.

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Translation

```
translator = pipeline("translation_en_to_de",
                      model="Helsinki-NLP/opus-mt-en-de")
outputs = translator(text, clean_up_tokenization_spaces=True, min_length=100)
print(outputs[0]['translation_text'])
```

```
Sehr geehrter Amazon, letzte Woche habe ich eine Optimus Prime Action Figur aus
Ihrem Online-Shop in Deutschland bestellt. Leider, als ich das Paket öffnete,
entdeckte ich zu meinem Entsetzen, dass ich stattdessen eine Action Figur von
Megatron geschickt worden war! Als lebenslanger Feind der Decepticons, Ich
hoffe, Sie können mein Dilemma verstehen. Um das Problem zu lösen, Ich fordere
einen Austausch von Megatron für die Optimus Prime Figur habe ich bestellt.
Anbei sind Kopien meiner Aufzeichnungen über diesen Kauf. Ich erwarte, bald von
Ihnen zu hören. Aufrichtig, Bumblebee.
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Text Generation

```
from transformers import set_seed
set_seed(42) # Set the seed to get reproducible results
```

```
generator = pipeline("text-generation")
response = "Dear Bumblebee, I am sorry to hear that your order was mixed up."
prompt = text + "\n\nCustomer service response:\n" + response
outputs = generator(prompt, max_length=200)
print(outputs[0]['generated_text'])
```

```
Customer service response:
Dear Bumblebee, I am sorry to hear that your order was mixed up. The
order was completely mislabeled, which is very common in our online
store, but I can appreciate it because it was my understanding from this
site and our customer service of the previous day that your order was
not made correct in our mind and that we are in a process of resolving
this matter. We can assure you that your order
```

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

Text Generation

Dear Amazon, last week I ordered an Optimus Prime action figure from your online store in Germany. Unfortunately, when I opened the package, I discovered to my horror that I had been sent an action figure of Megatron instead! As a lifelong enemy of the Decepticons, I hope you can understand my dilemma. To resolve the issue, I demand an exchange of Megatron for the Optimus Prime figure I ordered. Enclosed are copies of my records concerning this purchase. I expect to hear from you soon. Sincerely, Bumblebee.

Customer service response:
Dear Bumblebee, I am sorry to hear that your order was mixed up. The order was completely mislabeled, which is very common in our online store, but I can appreciate it because it was my understanding from this site and our customer service of the previous day that your order was not made correct in our mind and that we are in a process of resolving this matter. We can assure you that your order

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
https://github.com/nlp-with-transformers/notebooks

NLTK Gensim Word2Vec Visualization

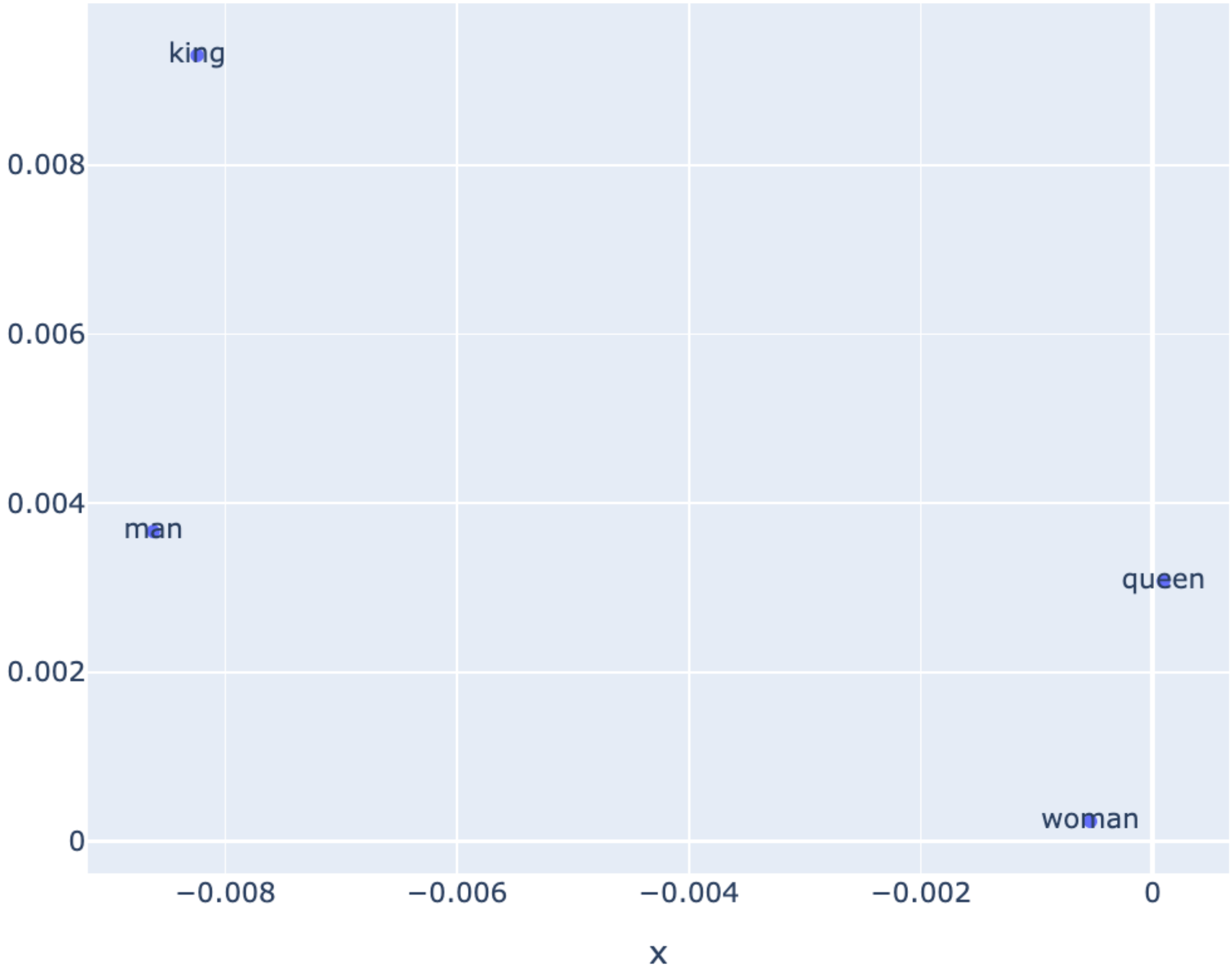
```
import nltk
import gensim
import plotly.express as px
nltk.download('punkt')
text = 'king queen man woman'
data = [nltk.word_tokenize(text)]
model = gensim.models.Word2Vec(sentences=data, min_count=1,
vector_size=100, window=5)

words = list(model.wv.index_to_key)
vectors = model.wv[words]

fig = px.scatter(x=vectors[:, 0], y=vectors[:, 1], text=words)
fig.show()
```

https://tinyurl.com/aintpupython101

NLTK Gensim Word2Vec Visualization



https://tinyurl.com/aintpupython101

Transformers Tokenizer Embeddings

```
!pip install transformers
!pip install torch
import torch
from transformers import BertTokenizer, BertModel
model_name = 'bert-base-uncased' #'bert-base-chinese'
model = BertModel.from_pretrained(model_name)
tokenizer = BertTokenizer.from_pretrained(model_name)
def get_bert_embeddings(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True,
    padding=True, max_length=512)
    with torch.no_grad():
        outputs = model(**inputs)
    embeddings = outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
    return embeddings
text = "I love apple."
embeddings = get_bert_embeddings(text)
print(embeddings)
```

https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

Table of contents

Natural Language Processing with Transformers

- Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Github: https://github.com/nlp-with-transformers/notebooks

[1]
```
!git clone https://github.com/nlp-with-transformers/notebooks.git
%cd notebooks
from install import *
install_requirements()
```

[3]
```
from utils import *
setup_chapter()
```

[12]
```
text = """Dear Amazon, last week I ordered an Optimus Prime action figure \
from your online store in Germany. Unfortunately, when I opened the package, \
I discovered to my horror that I had been sent an action figure of Megatron \
instead! As a lifelong enemy of the Decepticons, I hope you can understand my \
dilemma. To resolve the issue, I demand an exchange of Megatron for the \
Optimus Prime figure I ordered. Enclosed are copies of my records concerning \
this purchase. I expect to hear from you soon. Sincerely, Bumblebee."""
```

Text Clssification

[13]
```
from transformers import pipeline
classifier = pipeline("text-classification")
```

[14]
```
import pandas as pd
outputs = classifier(text)
pd.DataFrame(outputs)
```

https://tinyurl.com/aintpupython101

Summary

- **Natural Language Processing with Transformers**
  - **Transformer (Attention is All You Need)**
    - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
    - ChatGPT: Large Language Models (LLMs), Foundation Models
  - **Encoder-Decoder**
  - **Attention Mechanisms**
  - **Transfer Learning in NLP: Pre-train, Fine-tune**

References

- Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Denis Rothman (2021), Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more, Packt Publishing.
- Savaş Yıldırım and Meysam Asgari-Chenaghlu (2021), Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques, Packt Publishing.
- Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta (2020), Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems, O'Reilly Media.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov et al. (2023) "Llama 2: Open Foundation and Fine-Tuned Chat Models." arXiv preprint arXiv:2307.09288 (2023).
- Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun (2023). "A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT." arXiv preprint arXiv:2303.04226.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. (2023) "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing." ACM Computing Surveys 55, no. 9 (2023): 1-35.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min et al. (2023) "A Survey of Large Language Models." arXiv preprint arXiv:2303.18223.
- The Super Duper NLP Repo, https://notebooks.quantumstat.com/
- Jay Alammar (2018), The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/
- Jay Alammar (2019), A Visual Guide to Using BERT for the First Time, http://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/
- NLP with Transformer, https://github.com/nlp-with-transformers/notebooks
- Min-Yuh Day (2023), Python 101, https://tinyurl.com/aintpupython101