Transformers

Machine Learning

Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

MC886/MO444, November 17, 2022

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

The Illustrated Transformer

Discussions: Hacker News (65 points, 4 comments), Reddit r/MachineLearning (29 points, 3 comments)

Translations: Chinese (Simplified), French, Japanese, Korean, Russian, Spanish

Watch: MIT's Deep Learning State of the Art lecture referencing this post

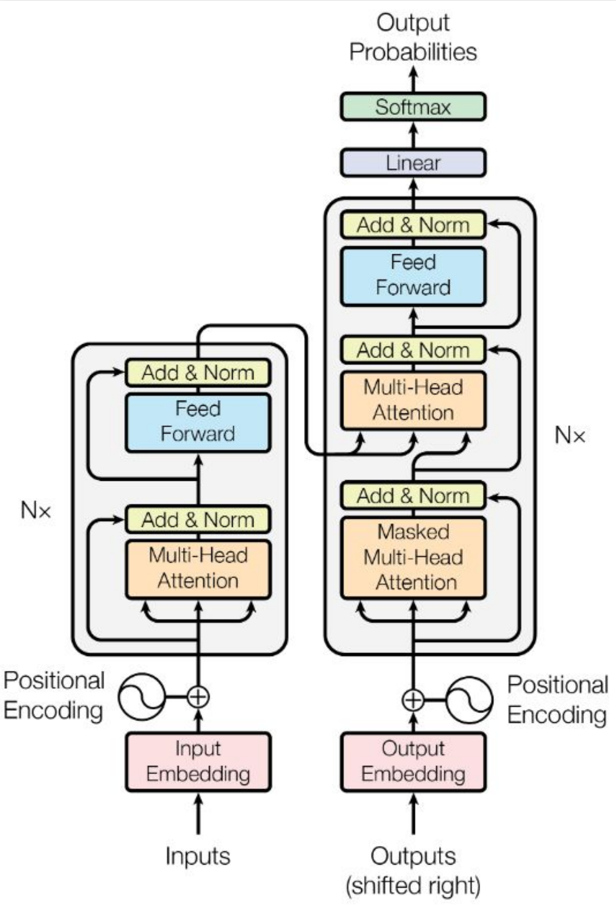
In the previous post, we looked at Attention – a ubiquitous method in modern deep learning models. Attention is a concept that helped improve the performance of neural machine translation applications. In this post, we will look at **The Transformer** – a model that uses attention to boost the speed with which these models can be trained. The Transformers outperforms the Google Neural Machine Translation model in specific tasks. The biggest benefit, however, comes from how The Transformer lends itself to parallelization. It is in fact Google Cloud's recommendation to use The Transformer as a reference model to use their Cloud TPU offering. So let's try to break the model apart and look at how it functions.

The Transformer was proposed in the paper Attention is All You Need. A TensorFlow implementation of it is available as a part of the Tensor2Tensor package. Harvard's NLP group created a guide annotating the paper with PyTorch implementation. In this post, we will attempt to oversimplify things a bit and introduce the concepts one by one to hopefully make it easier to understand to people without in-depth knowledge of the subject matter.

2020 Update: I've created a "Narrated Transformer" video which is a gentler approach to the topic:

Transformer

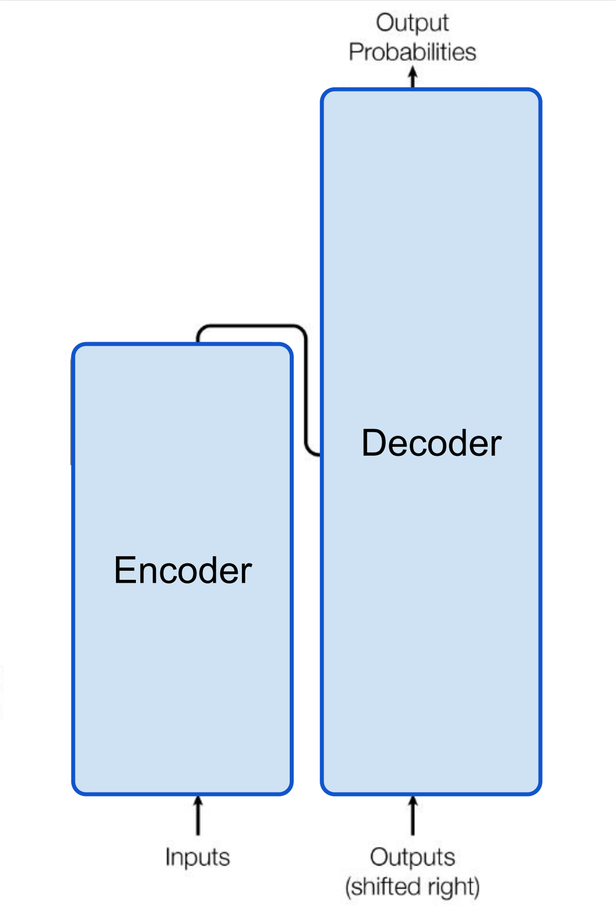
- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer

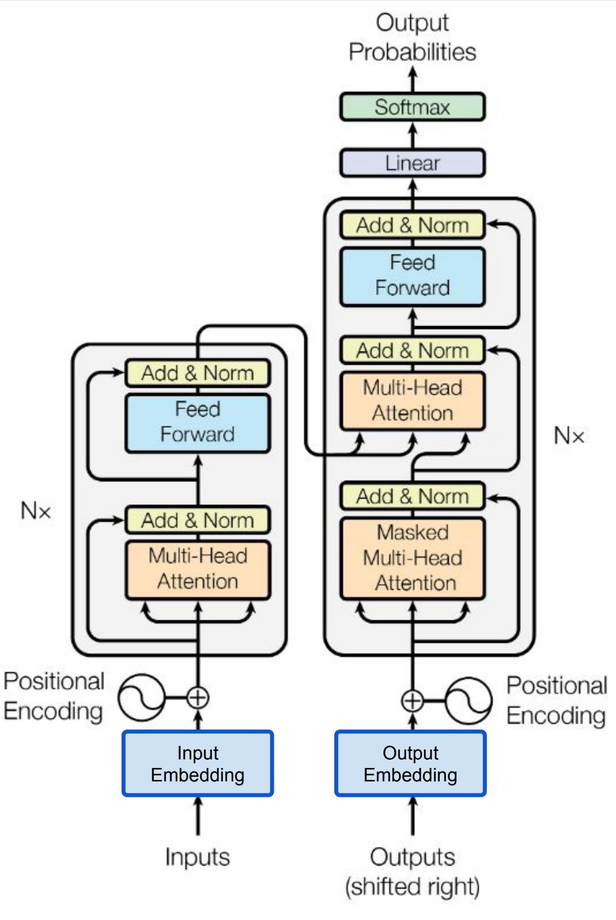
- Transformer Architecture
    - Encoder & Decoder
    - Input & output embedding
    - Positional encoding
    - Self-attention
    - Multi-head attention
    - Masked multi-head attention
    - Residual connections
    - Layer Normalization
    - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer

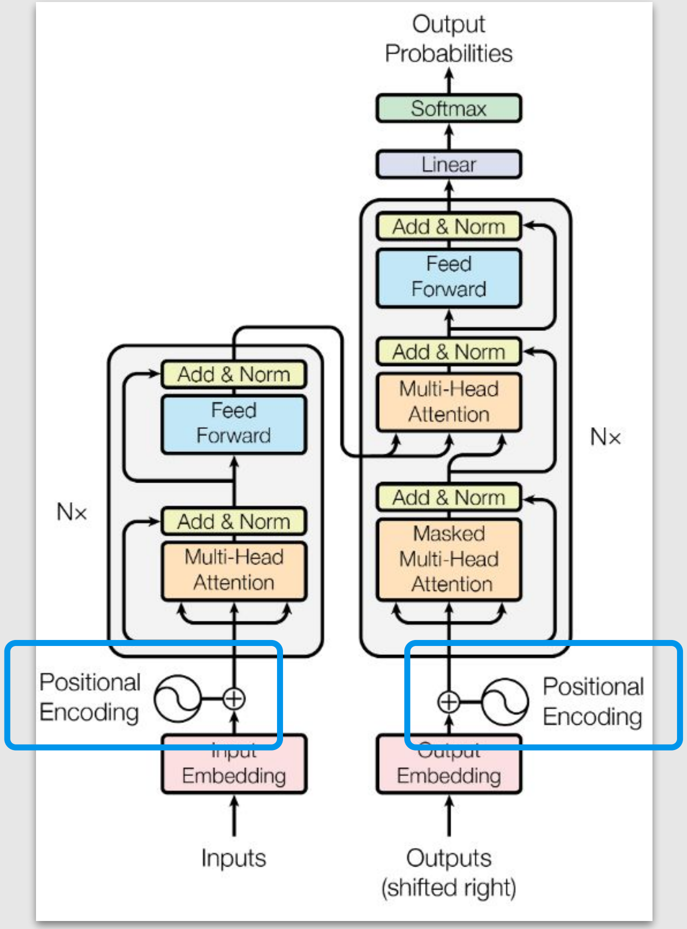
- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer

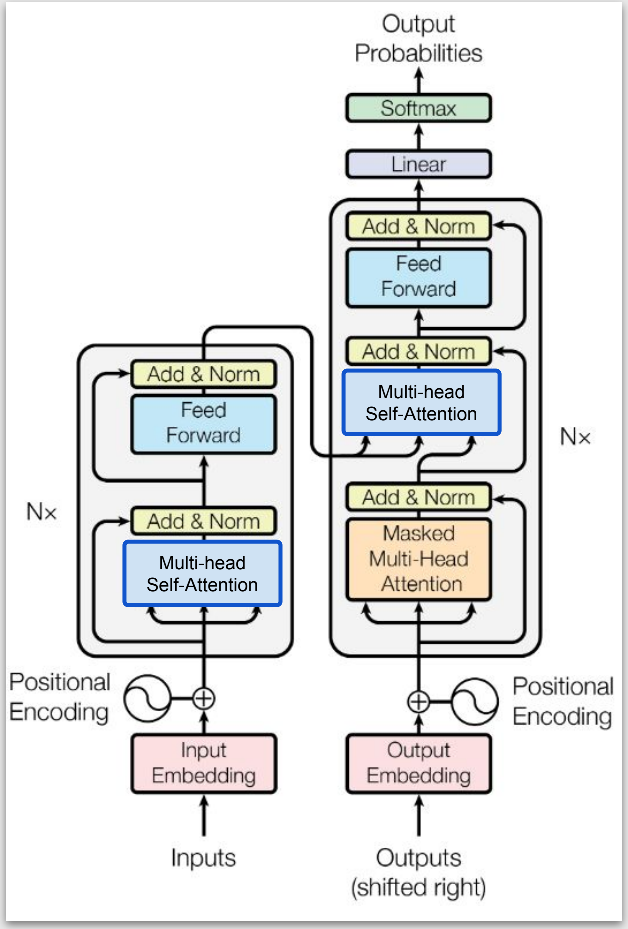
- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - **Positional encoding**
  - Self-attention
  - Multi-head attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer

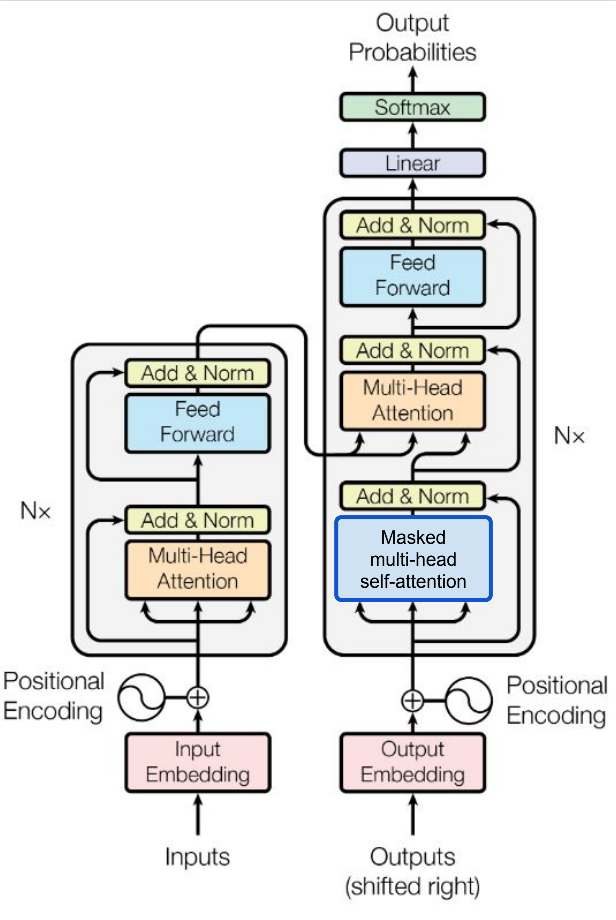
- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer

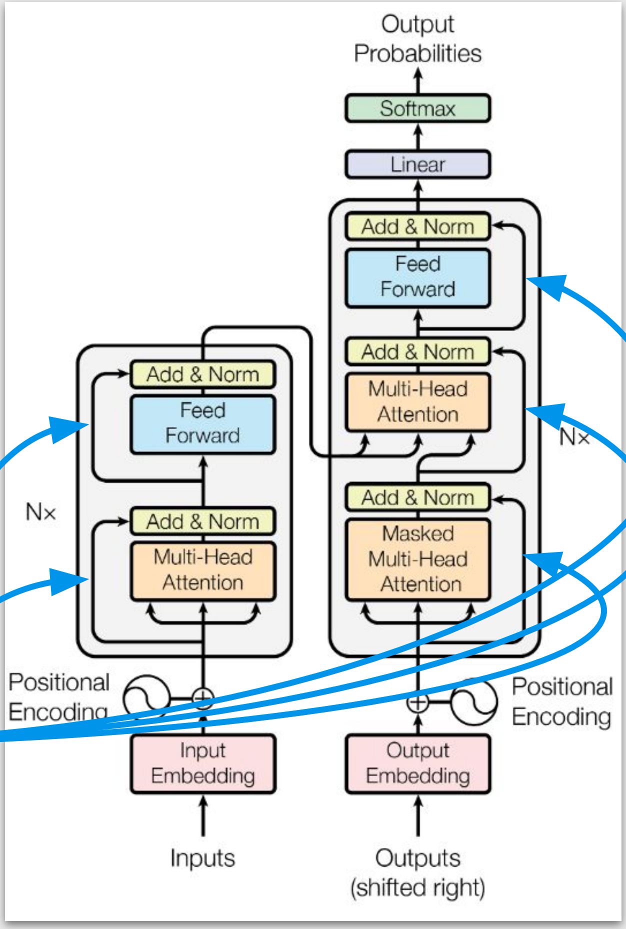
- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head attention
  - **Masked multi-head attention**
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer

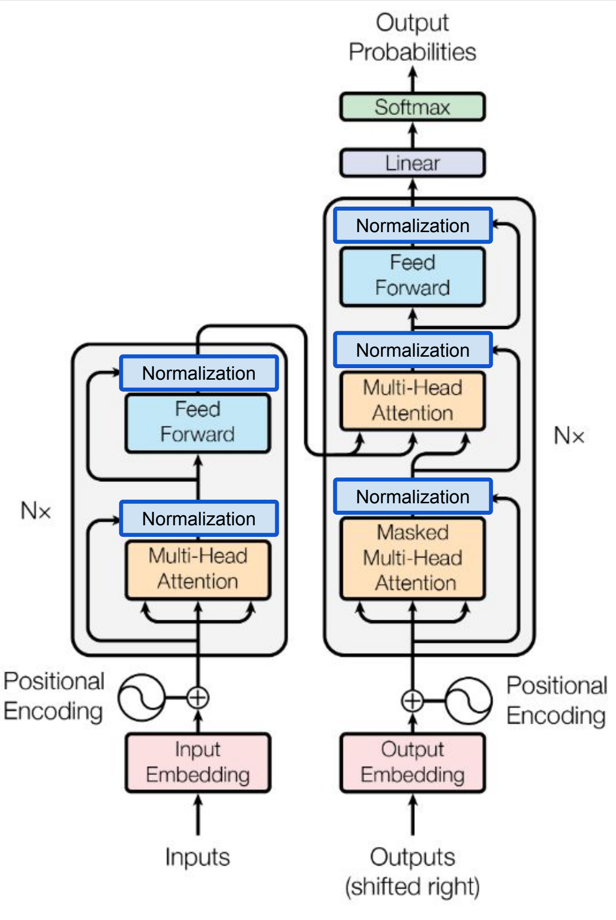
- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head attention
  - Masked multi-head attention
  - **Residual connections**
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer

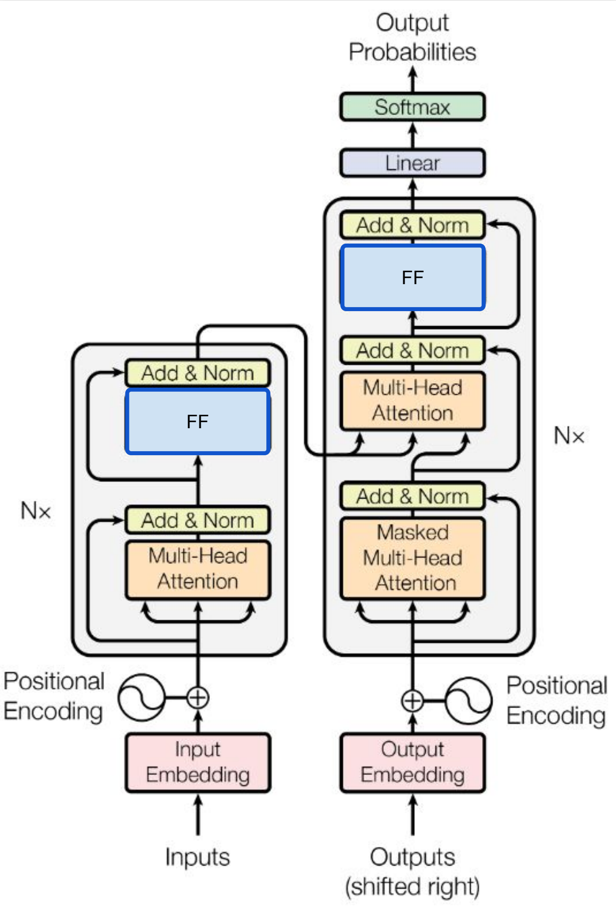
- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer

- Transformer Architecture
    - Encoder & Decoder
    - Input & output embedding
    - Positional encoding
    - Self-attention
    - Multi-head attention
    - Masked multi-head attention
    - Residual connections
    - Layer Normalization
    - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

README.md

https://github.com/huggingface/transformers

Transformers

build passing | license Apache-2.0 | website online | release v4.0.0 | Contributor Covenant v2.0 adopted

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

Transformers provides thousands of pretrained models to perform tasks on texts such as classification, information extraction, question answering, summarization, translation, text generation, etc in 100+ languages. Its aim is to make cutting-edge NLP easier to use for everyone.

Transformers provides APIs to quickly download and use those pretrained models on a given text, fine-tune them on your own datasets then share them with the community on our model hub. At the same time, each python module defining an architecture can be used as a standalone and modified to enable quick research experiments.

Transformers is backed by the two most popular deep learning libraries, PyTorch and TensorFlow, with a seamless integration between them, allowing you to train your models with one then load it for inference with the other.

Online demos

You can test most of our models directly on their pages from the model hub. We also offer an inference API to use those models.

Here are a few examples:

- Masked word completion with BERT
- Name Entity Recognition with Electra
- Text generation with GPT-2
- Natural Langugage Inference with RoBERTa
- Summarization with BART
- Question answering with DistilBERT
- Translation with T5

Transformers (HuggingFace)

- Available `pipelines` https://github.com/huggingface/transformers:
  - `feature-extraction`
  - `fill-mask`
  - `ner (named entity recognition)`
  - `question-answering`
  - `sentiment-analysis`
  - `summarization`
  - `text-generation`
  - `translation`
  - `zero-shot-classification`

```
!pip install datasets transformers[sentencepiece]
```

```python
from transformers import pipeline

classifier = pipeline("sentiment-analysis")
classifier("I've been waiting for a Transformer course my whole life.")
```

```
No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-english
(https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english)

[{'label': 'POSITIVE', 'score': 0.914790689945221}]
```

```python
classifier(
    ["I've been waiting for a Transformer course my whole life.",
     "I hate to wait so much!"]
)
```

```
[{'label': 'POSITIVE', 'score': 0.914790689945221},
 {'label': 'NEGATIVE', 'score': 0.7451478838920593}]
```

```python
from transformers import pipeline

generator = pipeline("text-generation", model="distilgpt2")
generator(
    "In this course, we will teach you how to",
    max_length=30,
    num_return_sequences=2,
)
```

```
[{'generated_text': 'In this course, we will teach you how to set up an online
marketing campaign to help them understand the challenges you face working online
without fear. If'},
 {'generated_text': 'In this course, we will teach you how to read an original story
by reading the script in a specific language. The first half of the course will'}]
```

```python
from transformers import pipeline

classifier = pipeline("zero-shot-classification")
classifier(
   "This is a course about the Transformers.",
   candidate_labels=["education", "politics", "business"],
)
```

```
No model was supplied, defaulted to facebook/bart-large-mnli
(https://huggingface.co/facebook/bart-large-mnli)

{'labels': ['education', 'business', 'politics'],
 'scores': [0.9185112714767456, 0.06004317477345466, 0.02144558541476726 5],
 'sequence': 'This is a course about the Transformers.'}
```

```python
from transformers import pipeline

translator =
    pipeline('text2text-generation', model="unicamp-dl/translation-pt-en-t5")

translator("Esse curso é produzido pela Unicamp.")
```

```
[{'generated_text': 'This course is produced by Unicamp.'}]
```

Lite Training Strategies for Portuguese-English and English-Portuguese Translation

Alexandre Lopes[1] Rodrigo Nogueira[2,3,4] Roberto Lotufo[2,3] Helio Pedrini[1]

[1]Institute of Computing, University of Campinas, Brazil
[2]School of Electrical and Computer Engineering, University of Campinas, Brazil
[3]NeuralMind Inteligência Artificial, Brazil
[4]David R. Cheriton School of Computer Science, University of Waterloo, Canada

https://huggingface.co/unicamp-dl/translation-pt-en-t5

```python
from transformers import pipeline

unmasker = pipeline("fill-mask")
unmasker("This course will teach you all about <mask> models.", top_k=2)
```

```
No model was supplied, defaulted to distilroberta-base
(https://huggingface.co/distilroberta-base)

[{'score': 0.196198508143425,
  'sequence': 'This course will teach you all about mathematical models.',
  'token': 30412,
  'token_str': ' mathematical'},
 {'score': 0.040527332574129105,
  'sequence': 'This course will teach you all about computational models.',
  'token': 38163,
  'token_str': ' computational'}]
```

```python
from transformers import pipeline

unmasker = pipeline("fill-mask")
result = unmasker("This man works as a <mask>.")
print([r["token_str"] for r in result])

result = unmasker("This woman works as a <mask>.")
print([r["token_str"] for r in result])
```

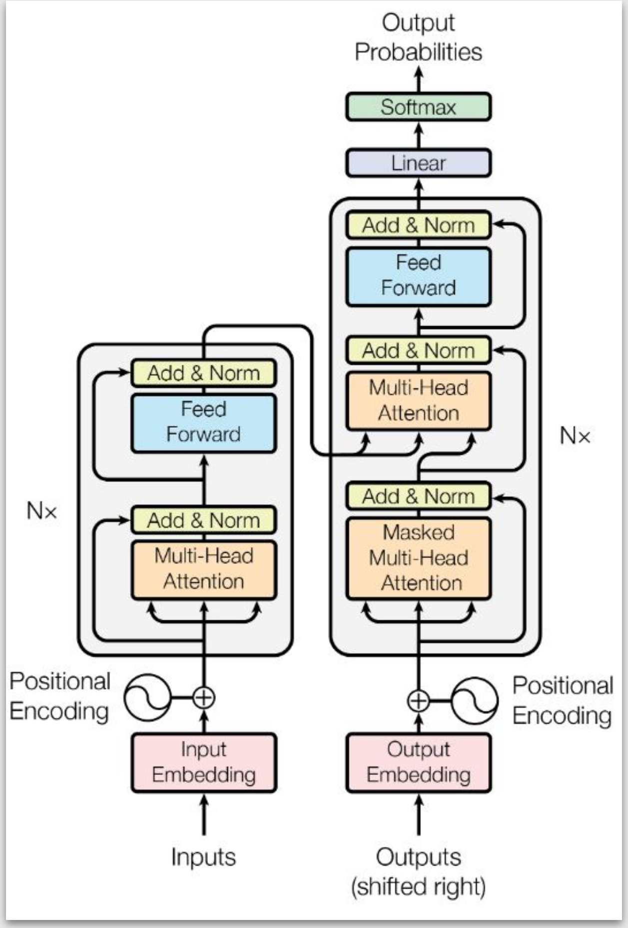
```
No model was supplied, defaulted to distilroberta-base
(https://huggingface.co/distilroberta-base)

[' translator', ' consultant', ' bartender', ' waiter', ' courier']
[' waitress', ' translator', ' nurse', ' bartender', ' consultant']
```

```python
from transformers import pipeline

unmasker = pipeline("fill-mask", model="bert-base-uncased")
result = unmasker("This man works as a [MASK].")
print([r["token_str"] for r in result])

result = unmasker("This woman works as a [MASK].")
print([r["token_str"] for r in result])
```

```
['carpenter', 'lawyer', 'farmer', 'businessman', 'doctor']
['nurse', 'maid', 'teacher', 'waitress', 'prostitute']
```

Transformer

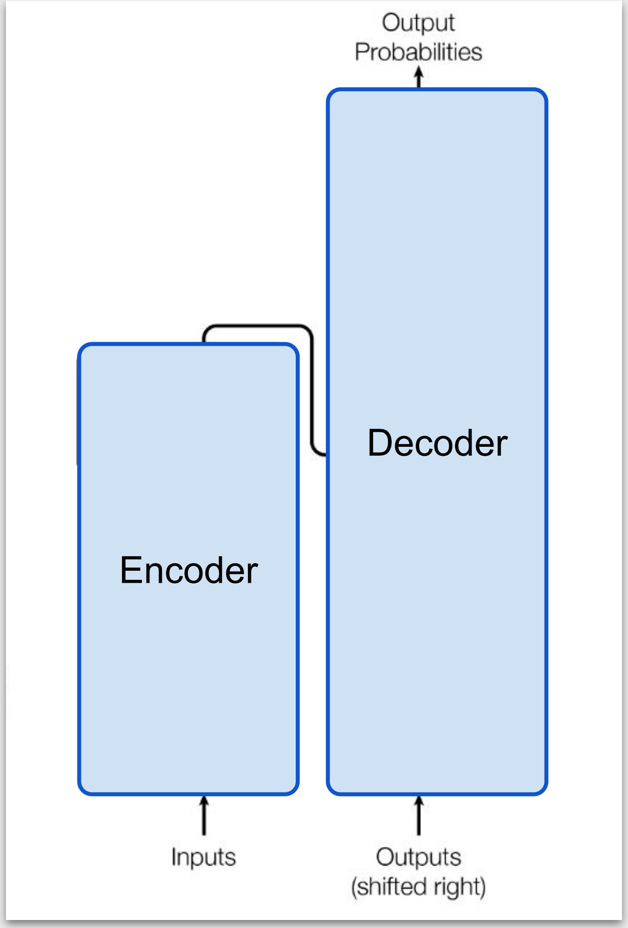
- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head self-attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer

- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head self-attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward

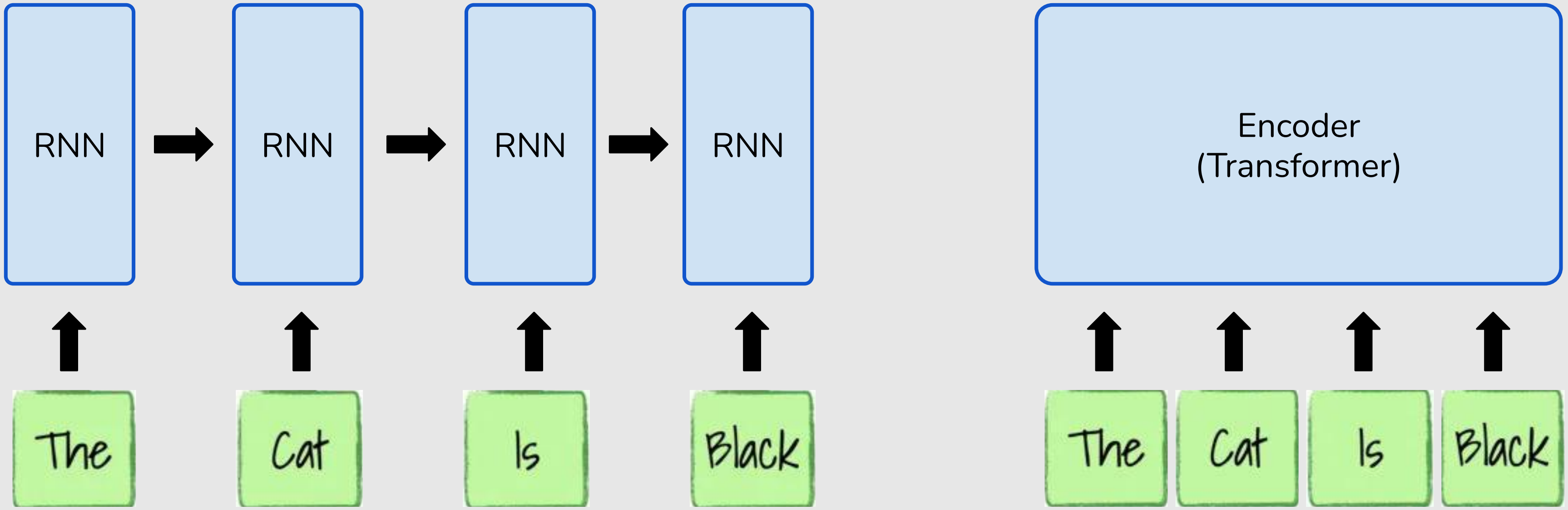


https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer: Encoder & Decoder

- The model is primarily composed of two blocks:
  - The **encoder** receives an input and builds a representation of it (its features). This means that the model is optimized to acquire understanding from the input. **6x**
  - The **decoder** uses the encoder’s representation (features) along with other inputs to generate a target sequence. This means that the model is optimized for generating outputs. **6x**

Transformers vs. RNNs

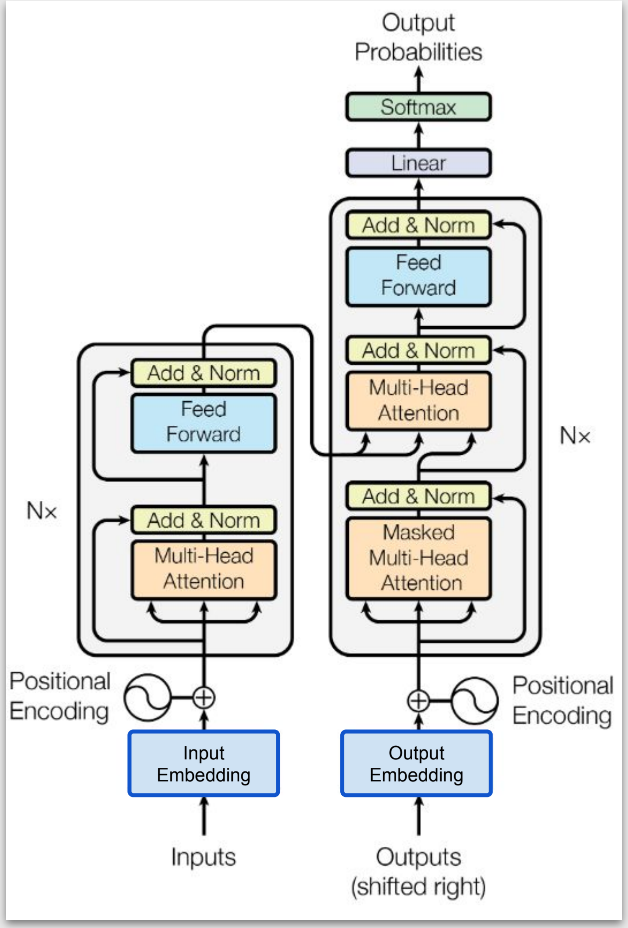


Transformer: Encoder & Decoder

- **Encoder-only models**: ALBERT, BERT, DistilBERT, ELECTRA, RoBERTa
- **Decoder-only models**: CTRL, GPT, GPT-2, GPT-3, Transformer XL.
- **Encoder-decoder models or sequence-to-sequence models**: BART, mBART, Marian, T5.

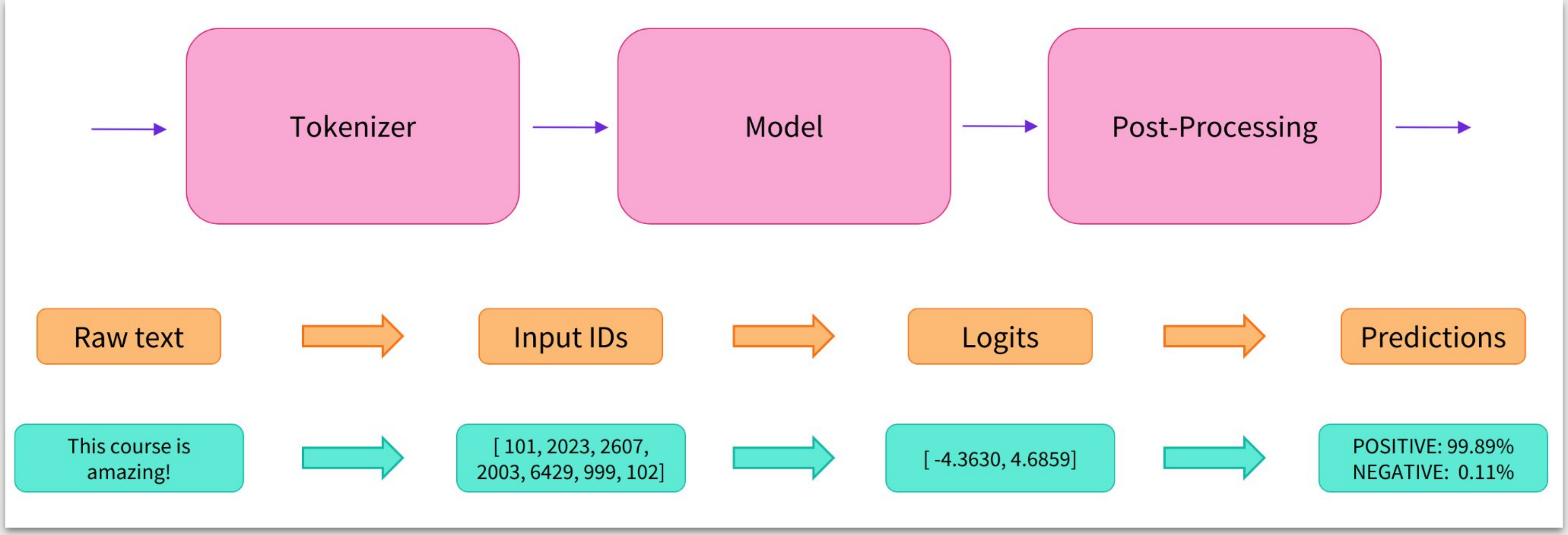
Transformer

- Transformer Architecture
  - Encoder & Decoder
  - **Input & output embedding**
  - Positional encoding
  - Self-attention
  - Multi-head self-attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer: Input & output embedding



Transformer: Input & output embedding

- Word-based

Let’s	do	tokenization!

Let	's	do	tokenization	!

- Character-based

L	e	t	'	s	d	o	t	o	k	e	n	i	z	a	t	i	o	n	!

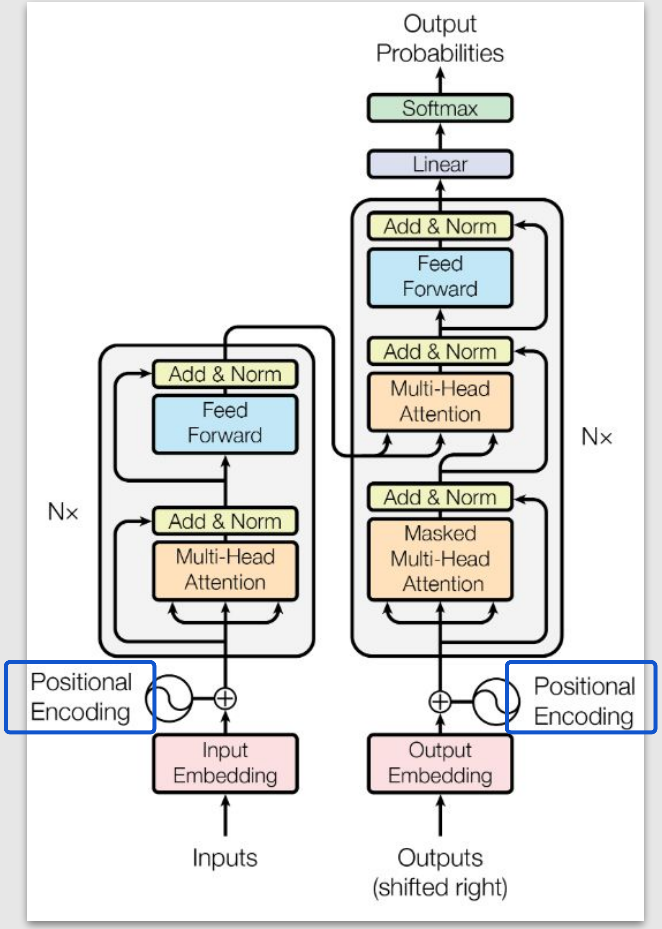
- Subword tokenization

Let's</w>	do</w>	token	ization</w>	!</w>

https://huggingface.co/transformers/tokenizer_summary.html

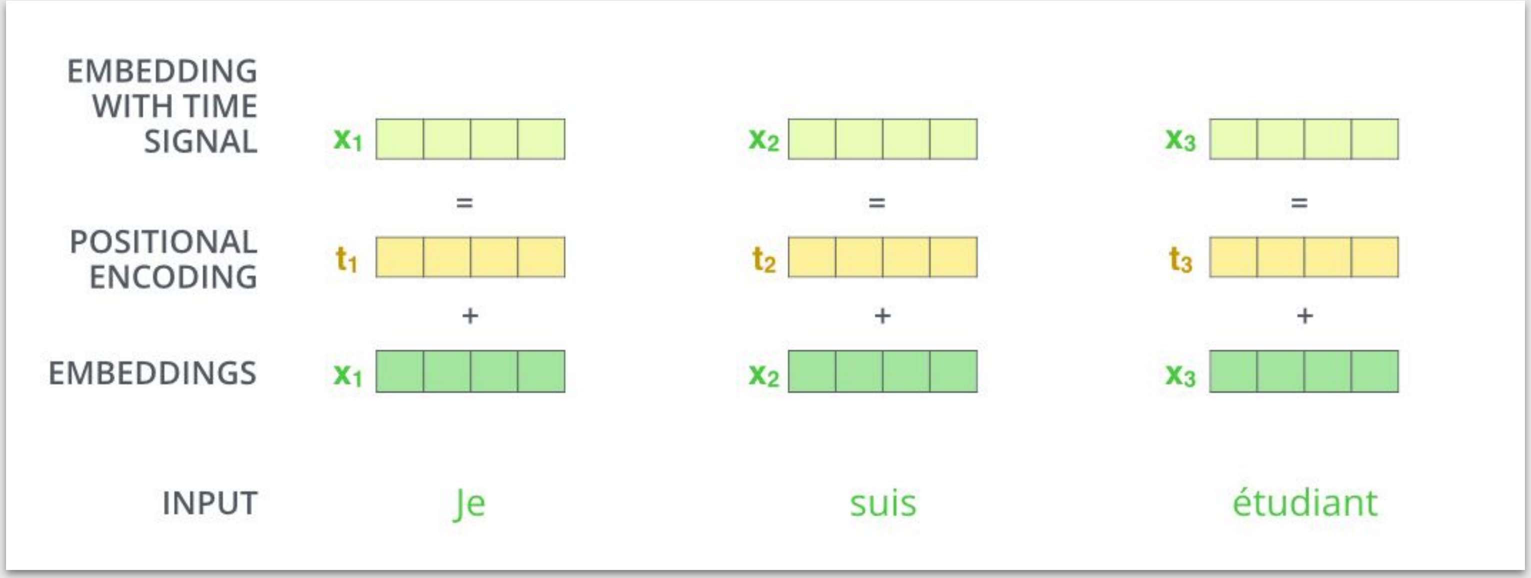
Transformer

- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head self-attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer: Positional Encoding

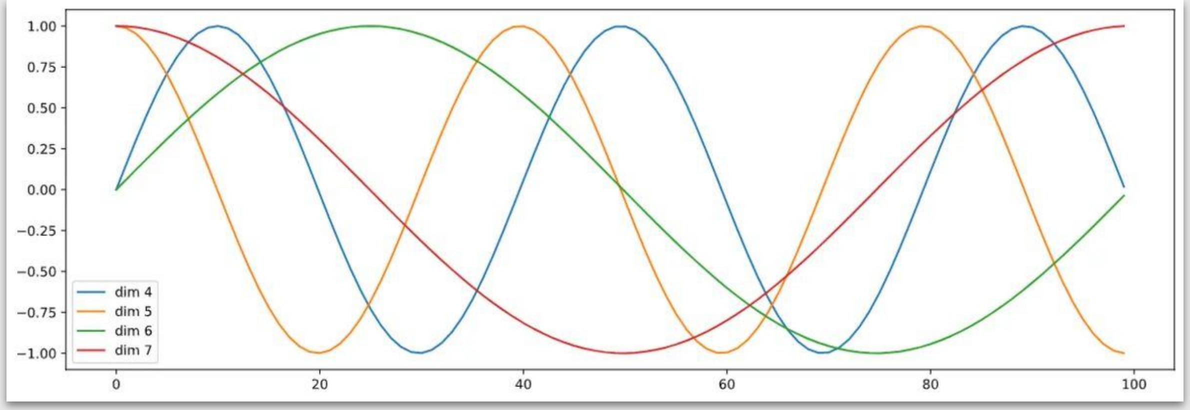


https://huggingface.co/transformers/tokenizer_summary.html

Transformer: Positional Encoding

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

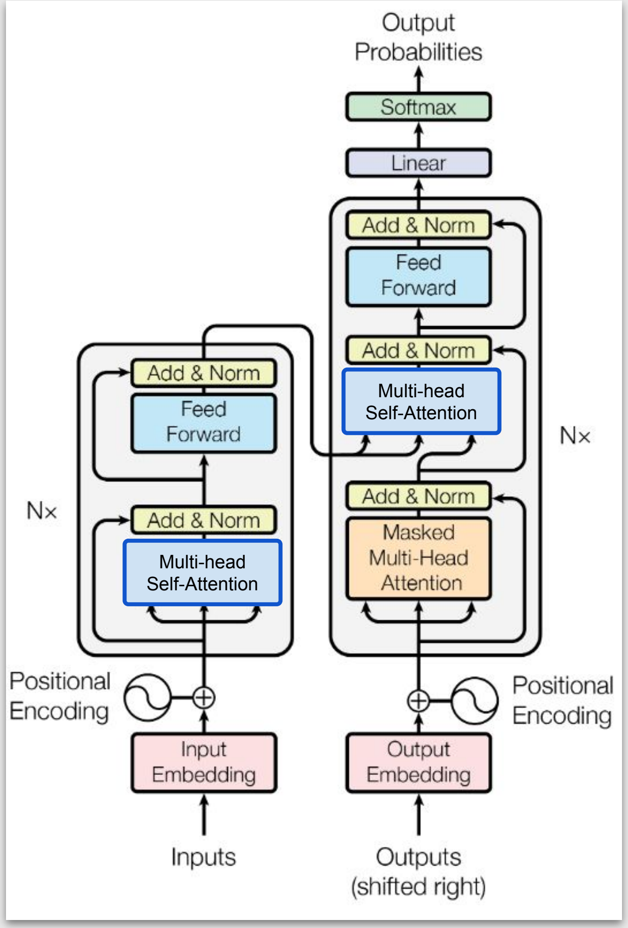
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



https://huggingface.co/transformers/tokenizer_summary.html

Transformer

- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head self-attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

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

The Illustrated Transformer

Discussions: Hacker News (65 points, 4 comments), Reddit r/MachineLearning (29 points, 3 comments)

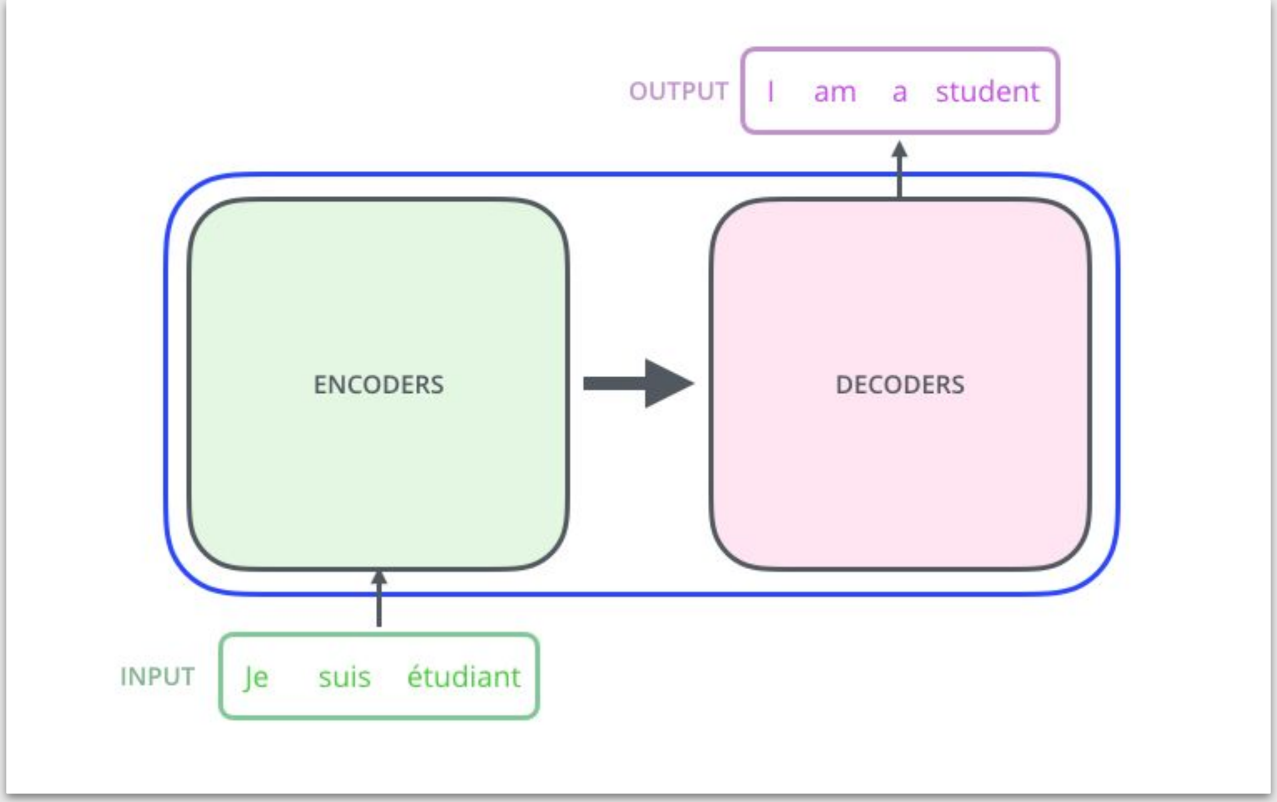
Translations: Chinese (Simplified), French, Japanese, Korean, Russian, Spanish

Watch: MIT's Deep Learning State of the Art lecture referencing this post

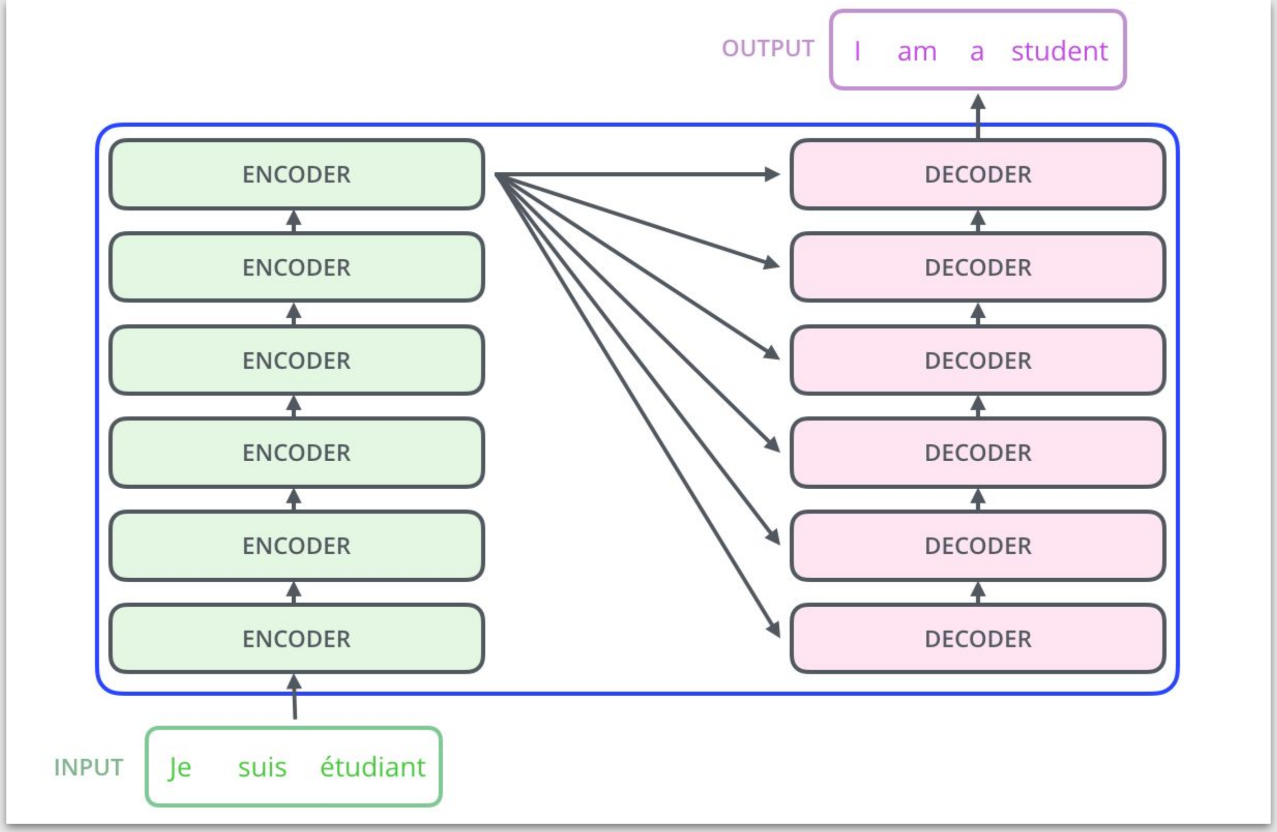
In the previous post, we looked at Attention – a ubiquitous method in modern deep learning models. Attention is a concept that helped improve the performance of neural machine translation applications. In this post, we will look at **The Transformer** – a model that uses attention to boost the speed with which these models can be trained. The Transformers outperforms the Google Neural Machine Translation model in specific tasks. The biggest benefit, however, comes from how The Transformer lends itself to parallelization. It is in fact Google Cloud's recommendation to use The Transformer as a reference model to use their Cloud TPU offering. So let's try to break the model apart and look at how it functions.

The Transformer was proposed in the paper Attention is All You Need. A TensorFlow implementation of it is available as a part of the Tensor2Tensor package. Harvard's NLP group created a guide annotating the paper with PyTorch implementation. In this post, we will attempt to oversimplify things a bit and introduce the concepts one by one to hopefully make it easier to understand to people without in-depth knowledge of the subject matter.

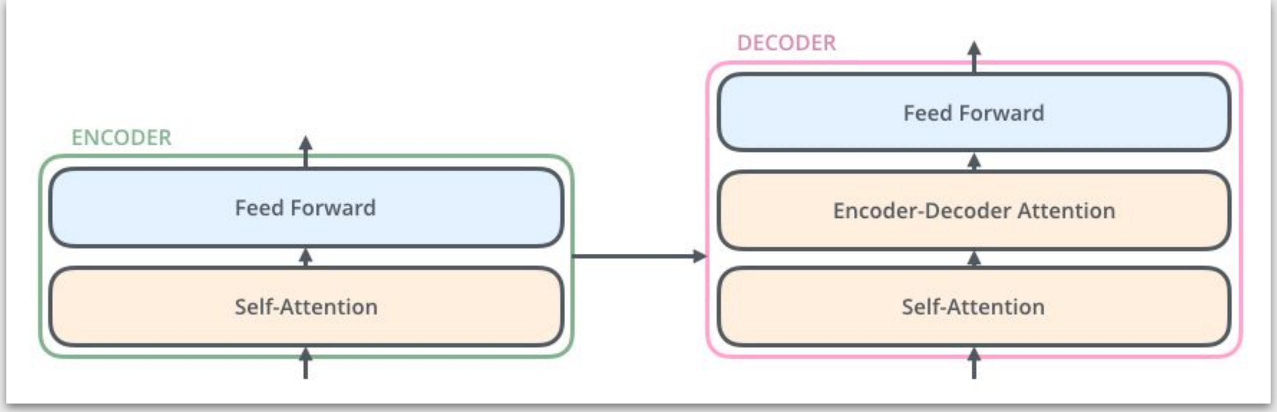
2020 Update: I've created a "Narrated Transformer" video which is a gentler approach to the topic:



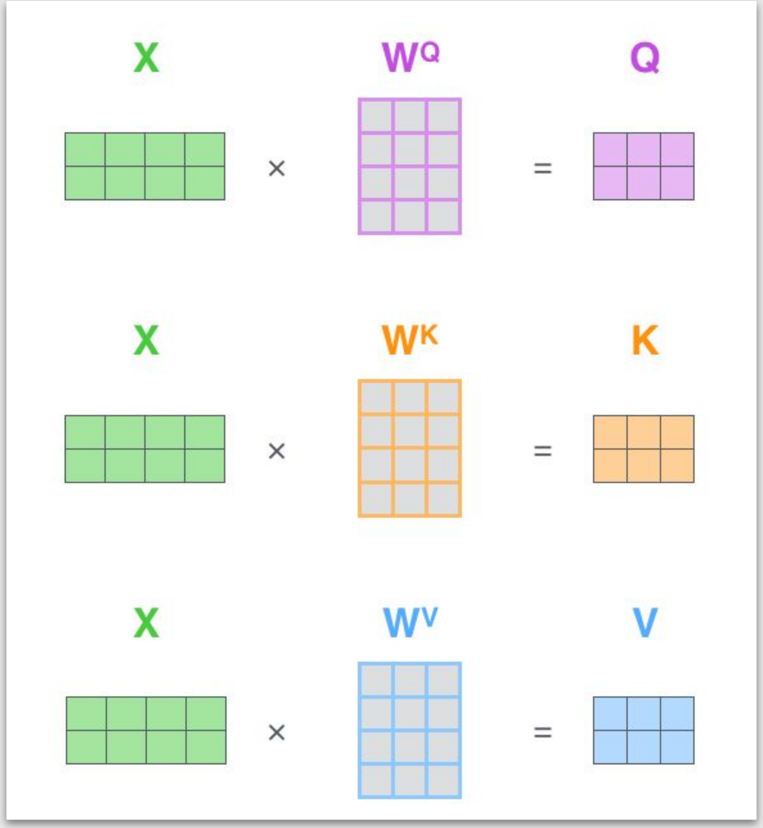
https://jalammar.github.io/illustrated-transformer

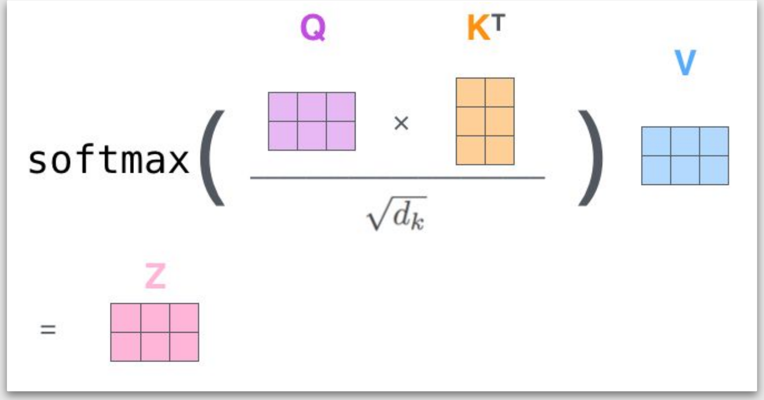


https://jalammar.github.io/illustrated-transformer

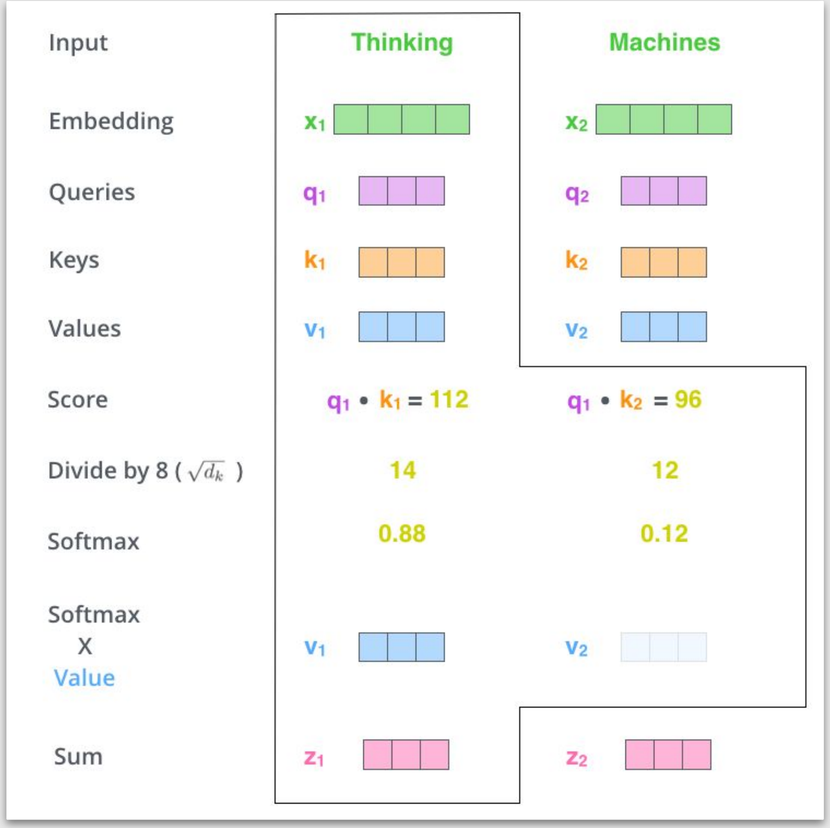


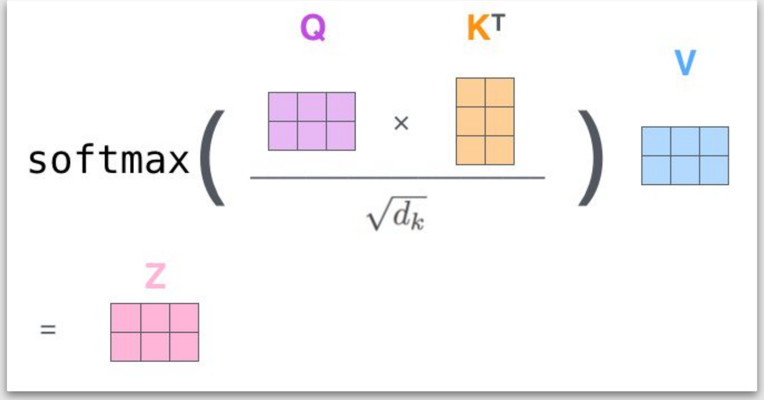
https://jalammar.github.io/illustrated-transformer



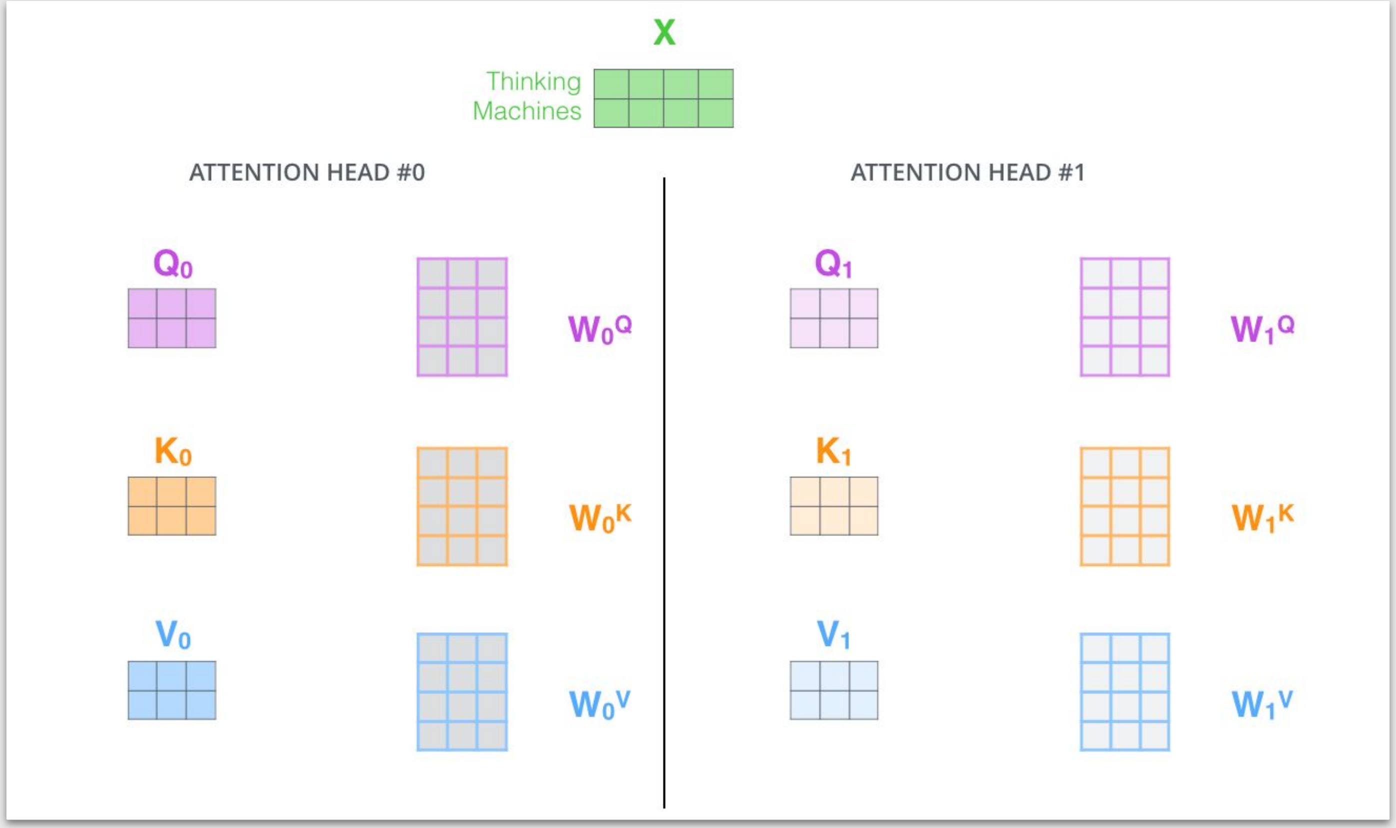


https://jalammar.github.io/illustrated-transformer

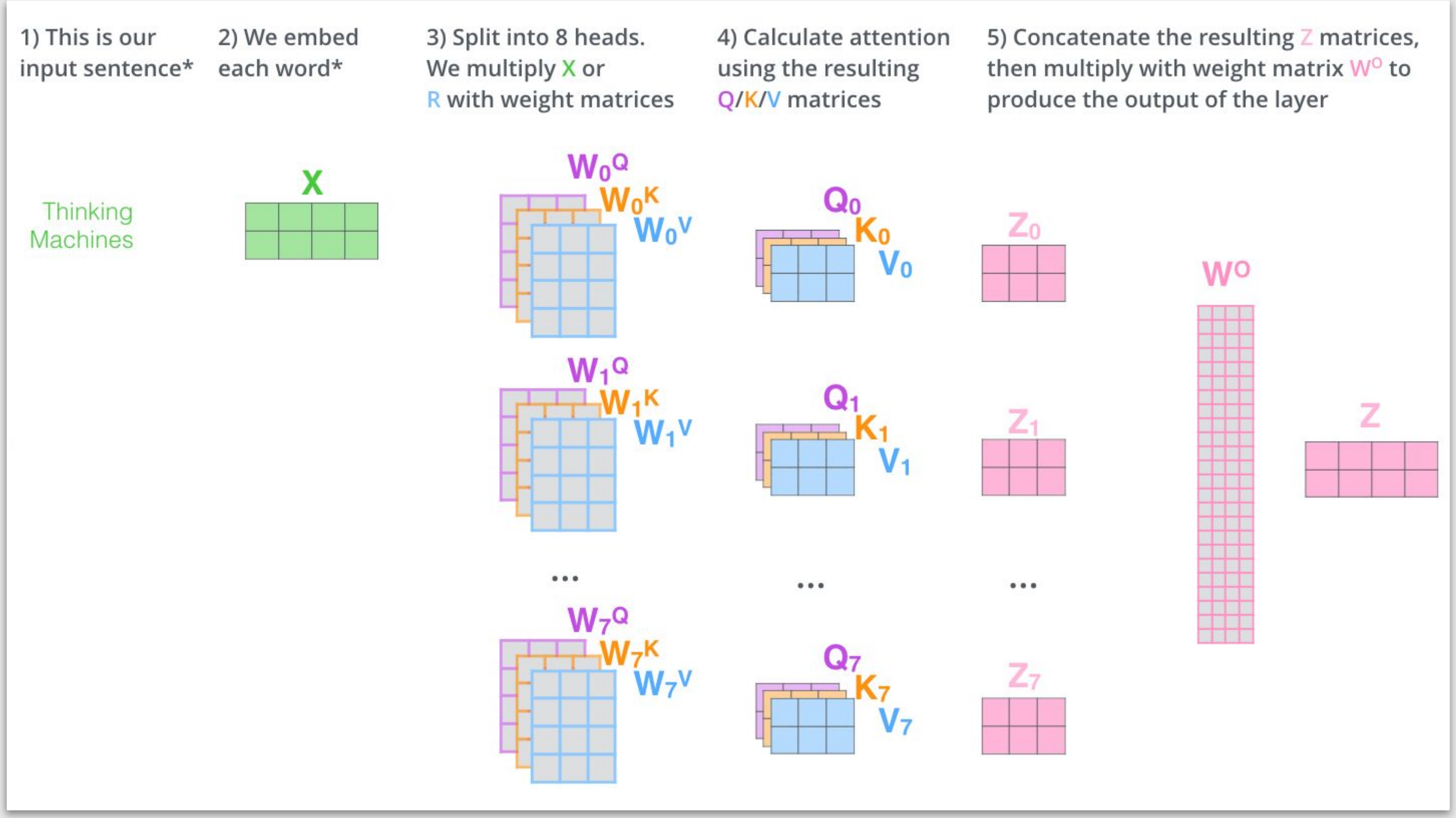




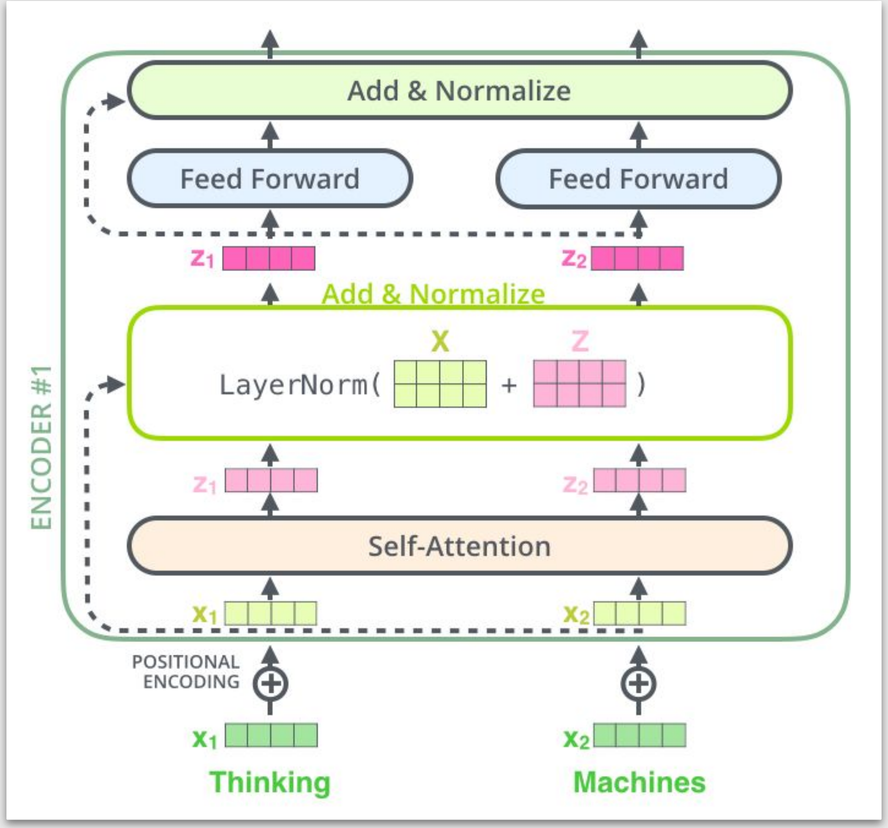
https://jalammar.github.io/illustrated-transformer



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



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



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

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

The Illustrated Transformer

Discussions: Hacker News (65 points, 4 comments), Reddit r/MachineLearning (29 points, 3 comments)

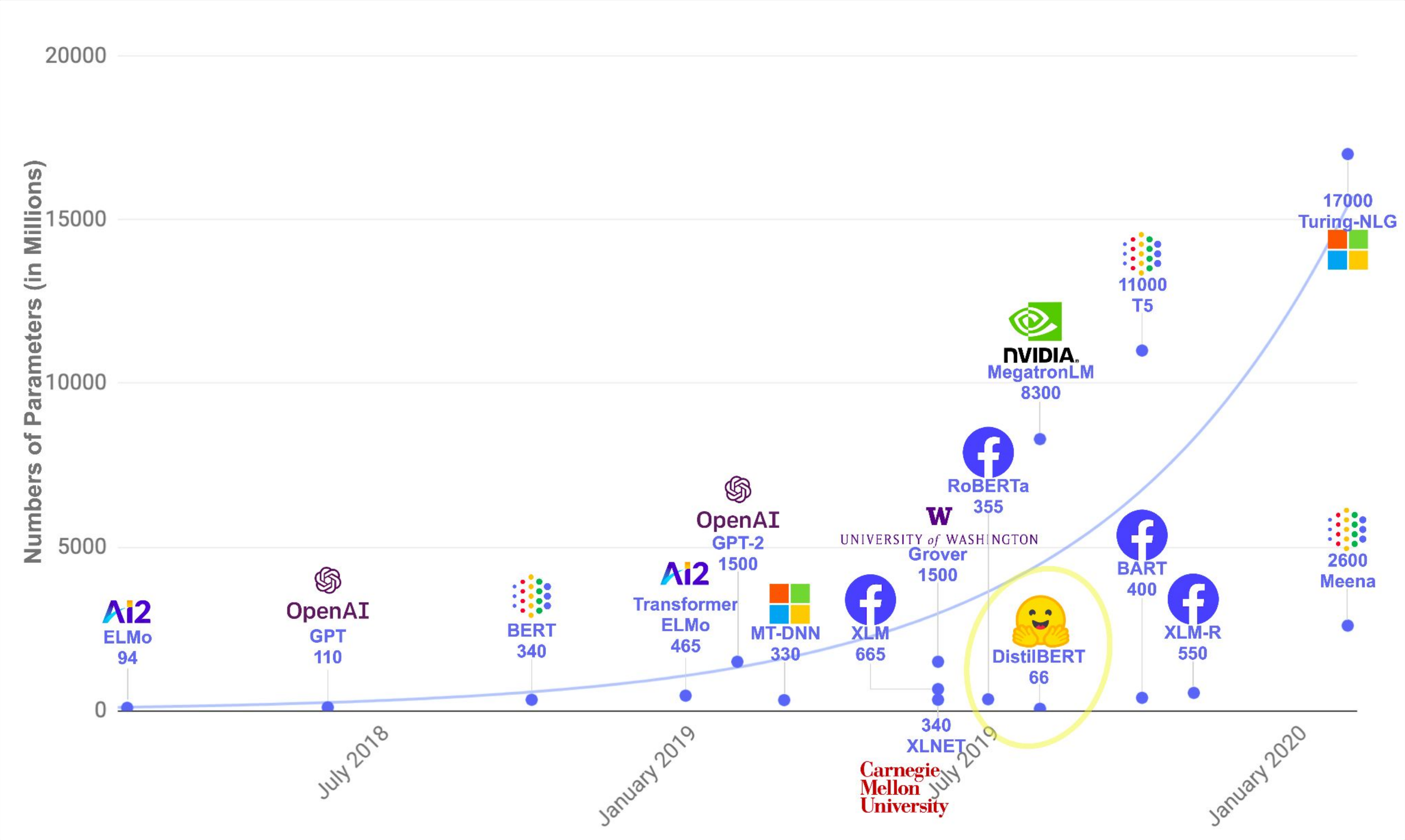
Translations: Chinese (Simplified), French, Japanese, Korean, Russian, Spanish

Watch: MIT's Deep Learning State of the Art lecture referencing this post

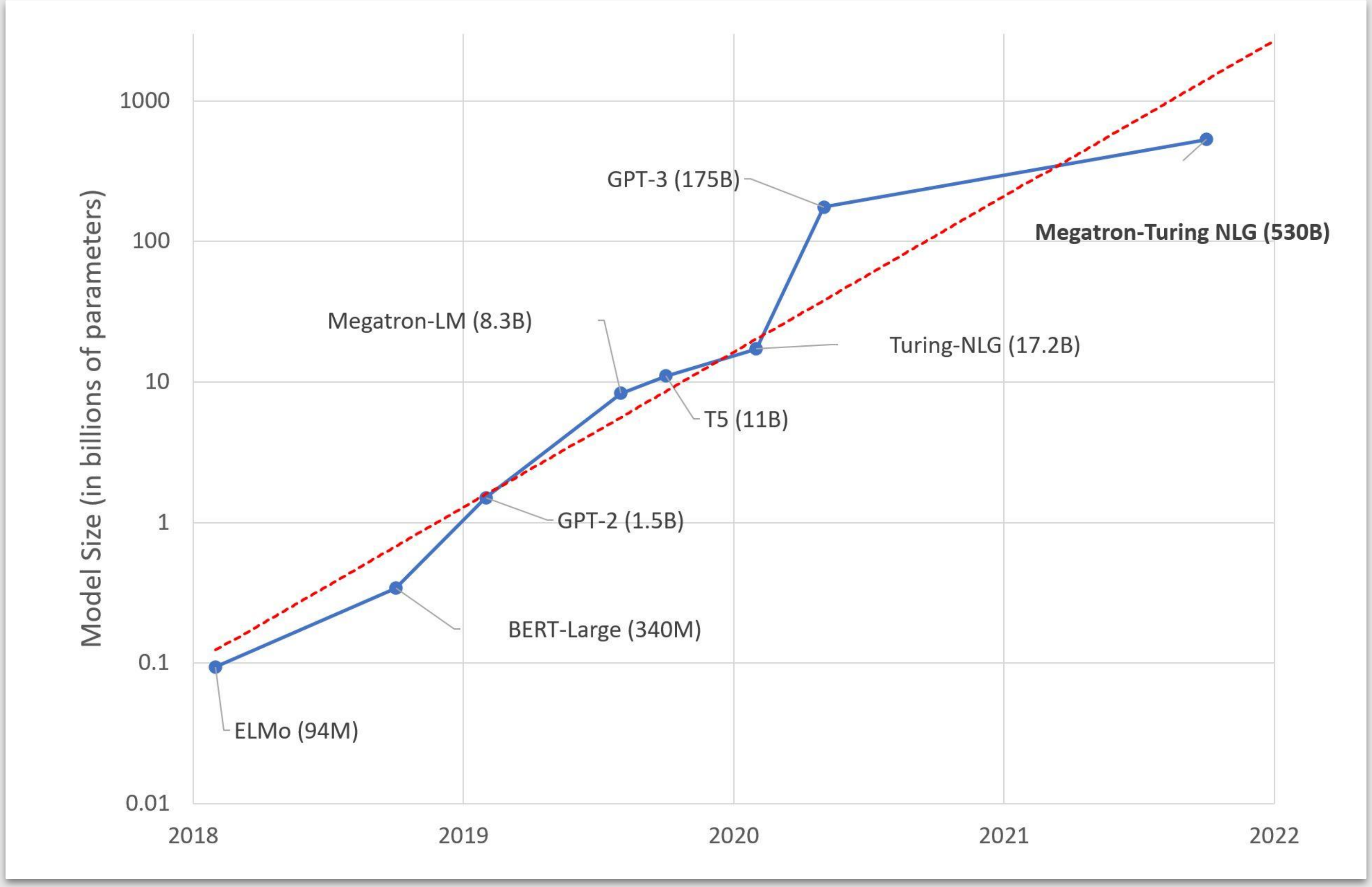
In the previous post, we looked at Attention – a ubiquitous method in modern deep learning models. Attention is a concept that helped improve the performance of neural machine translation applications. In this post, we will look at **The Transformer** – a model that uses attention to boost the speed with which these models can be trained. The Transformers outperforms the Google Neural Machine Translation model in specific tasks. The biggest benefit, however, comes from how The Transformer lends itself to parallelization. It is in fact Google Cloud's recommendation to use The Transformer as a reference model to use their Cloud TPU offering. So let's try to break the model apart and look at how it functions.

The Transformer was proposed in the paper Attention is All You Need. A TensorFlow implementation of it is available as a part of the Tensor2Tensor package. Harvard's NLP group created a guide annotating the paper with PyTorch implementation. In this post, we will attempt to oversimplify things a bit and introduce the concepts one by one to hopefully make it easier to understand to people without in-depth knowledge of the subject matter.

2020 Update: I've created a "Narrated Transformer" video which is a gentler approach to the topic:



https://huggingface.co/course/chapter1/4?fw=pt



https://huggingface.co/course/chapter1/4?fw=pt

References

- “Attention Is All You Need”, https://arxiv.org/pdf/1706.03762.pdf
- “Attention, please! A survey of Neural Attention Models in Deep Learning”, A. Correia and E. Colombini, https://arxiv.org/pdf/2103.16775.pdf
- “Transformers: State-of-the-Art Natural Language Processing”, https://aclanthology.org/2020.emnlp-demos.6.pdf
- “The Illustrated Transformer”, Jay Alammar, https://jalammar.github.io/illustrated-transformer
- “Transformers from Scratch”, http://peterbloem.nl/blog/transformers