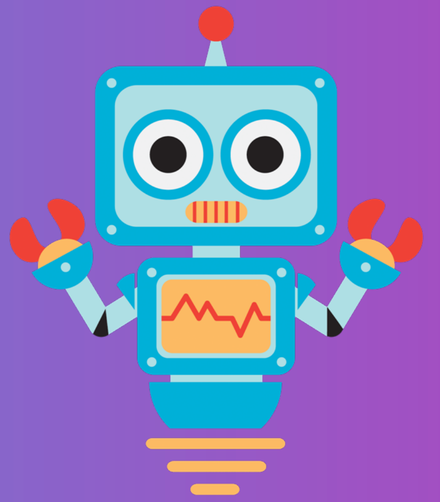
Dr. Abulkarim Albanna

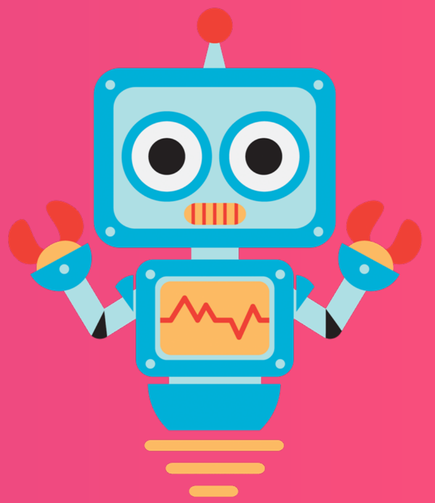
# Natural Language Processing Applications with Transformers and Generative Models



## Day 2

January 14th 2024

# Transformers



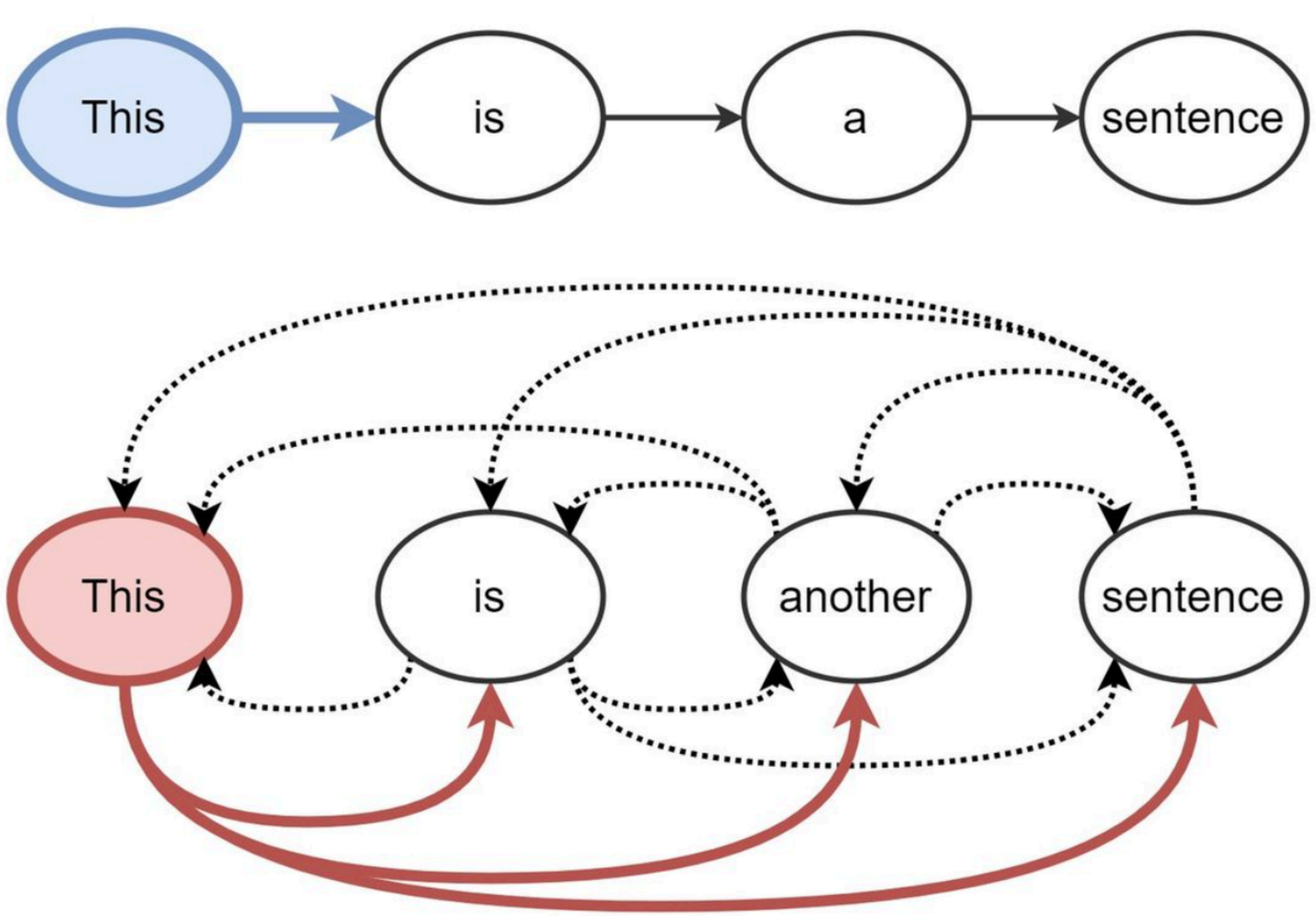
# Transformers

## History

- LSTMs, GRUs and other flavors of RNNs were the essential building blocks of NLP models for two decades since 1990s.
- CNNs were the essential building blocks of vision (and some NLP) models for three decades since the 1980s.
- In 2017, Transformers (proposed in the "Attention Is All You Need" paper) demonstrated that recurrence and/or convolutions are not essential for building high-performance natural language models.
- In 2020, Vision Transformer (ViT) (An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale) demonstrated that convolutions are not essential for building high-performance vision models.

# Transformers

## History




**RNN**

Translation?

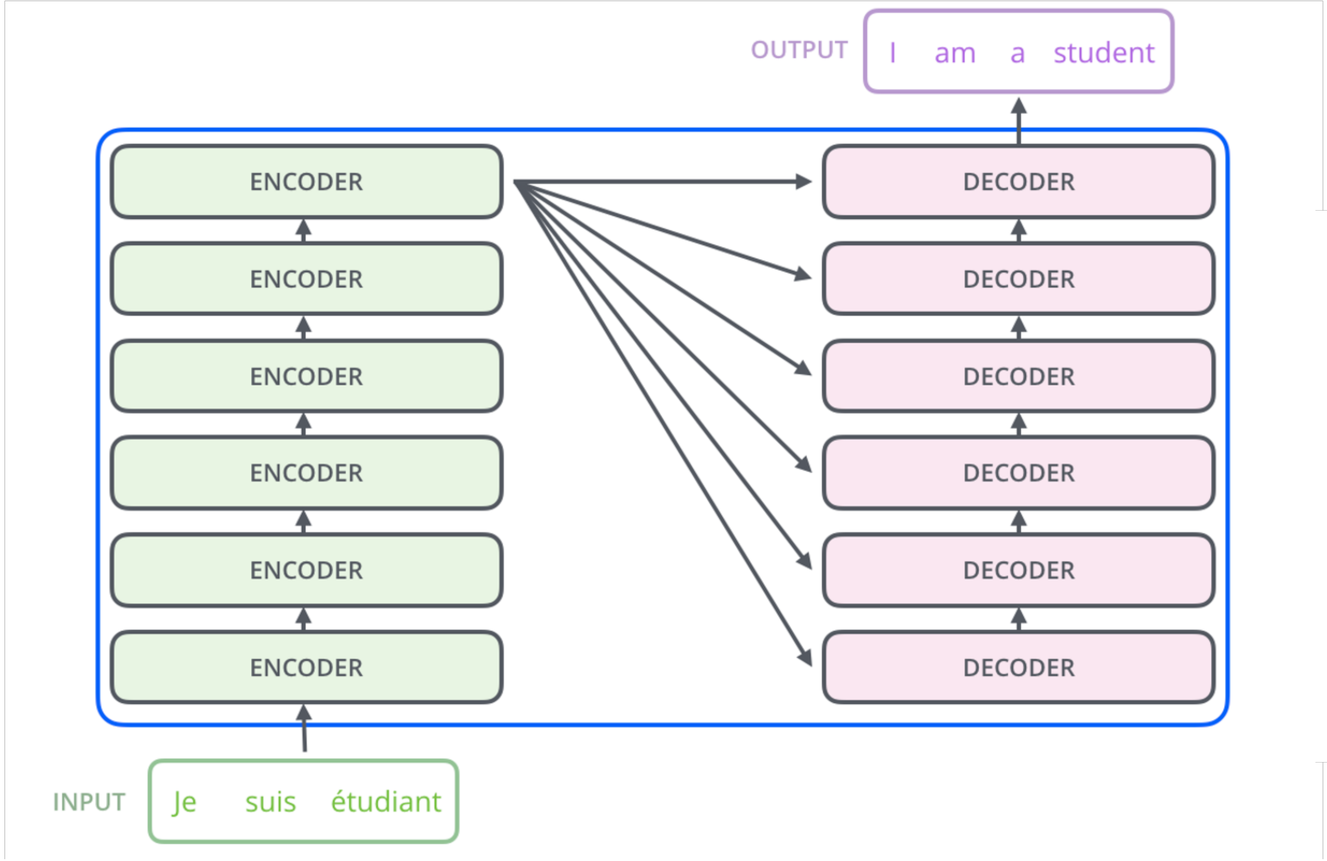
Sentiment?

Next word?

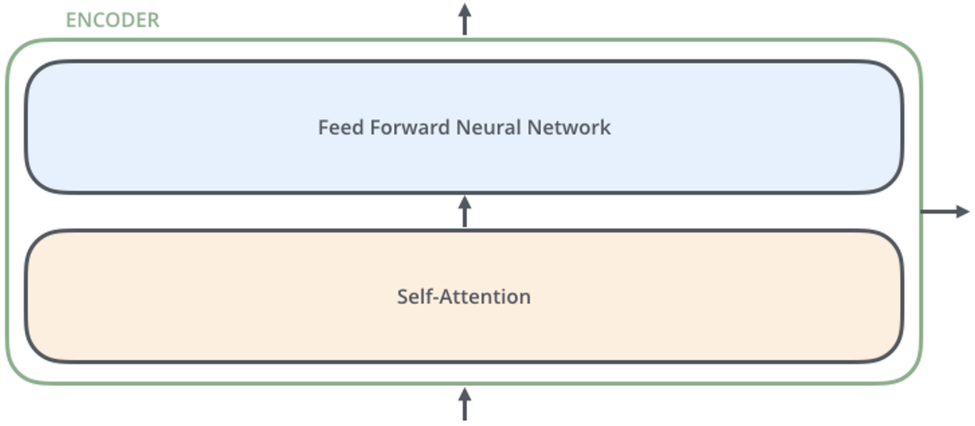
**Transf.**

Part-of-speech tags?

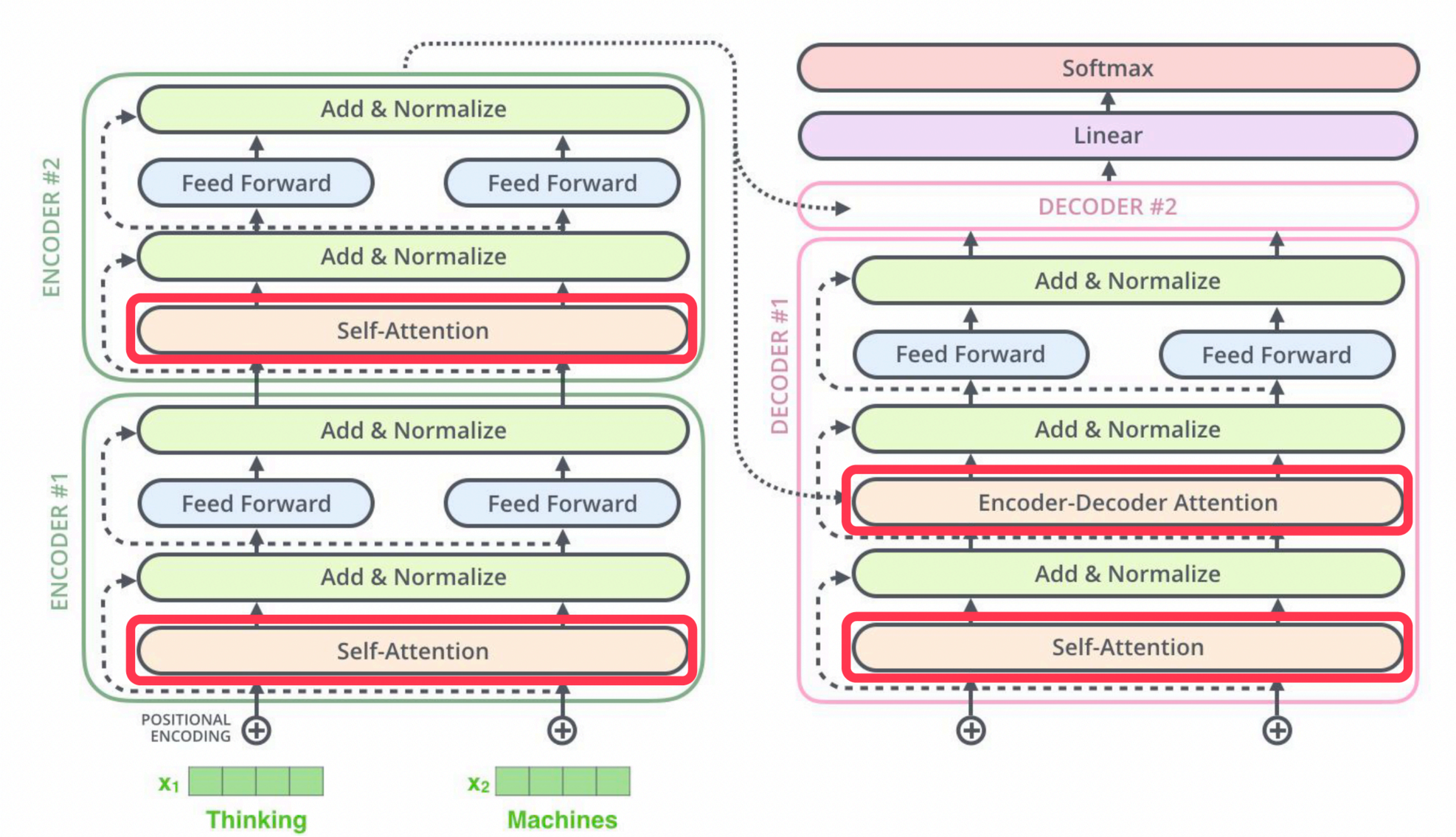
# Transformers




An Encoder Block: same structure, different parameters



# Transformers



# Transformers

## Self-Attention

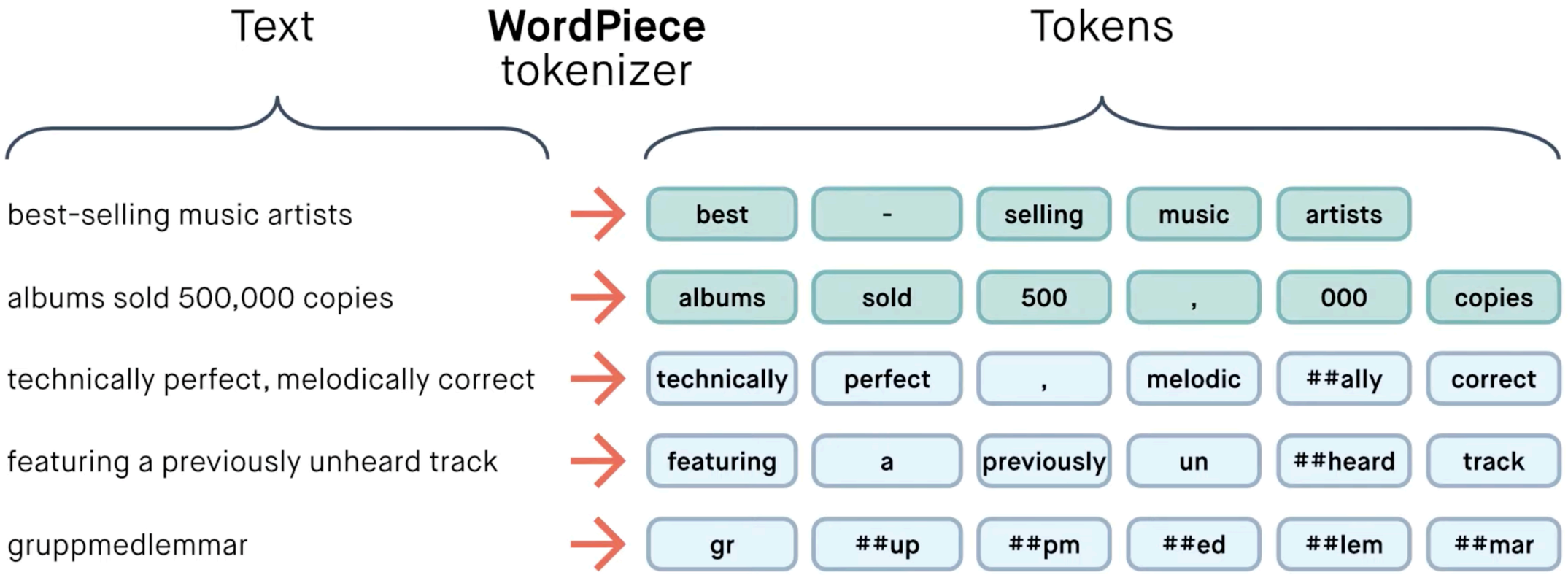
### Tokenization

- LSTMs, GRUs and other flavors of RNNs were the essential building blocks of NLP models for two decades since 1990s.
- Before processing, the raw input text is tokenized into smaller units, often subwords or words. This process breaks down the input into chunks that the model can recognize.
- This step is crucial because the model has a fixed vocabulary, and tokenization ensures the input is in a format that matches this vocabulary.

# Transformers

## Self-Attention

### Tokenization



# Transformers

## Self-Attention

### Embedding

- Dense vector representations of words or sentences that capture semantic and syntactic properties of words or sentences.
- Provide a way to convert textual information into a form that machine learning algorithms can process

- Obtained by training models, such as BERT and its variants, Word2Vec, GloVe, or FastText.

- Encapsulate the semantic meaning of words (which are internally represented as or more tokens) or semantic and syntactic properties of sentences by representing the dense, low-dimensional vectors.

# Transformers

## Self-Attention

### Contextualized vs. Non-Contextualized Embeddings

- Encoder models, like the Transformer-based BERT (Bidirectional Encoder Representations from Transformers), are designed to generate contextualized embeddings.

- Traditional word embeddings that assign a static vector to each word (such as Word2Vec or GloVe), these models consider the context of a word (i.e., the words that surround it).

# Transformers

## Self-Attention

### Contextualized vs. Non-Contextualized Embeddings

- With embeddings, you can perform various arithmetic operations to carry out specific tasks:

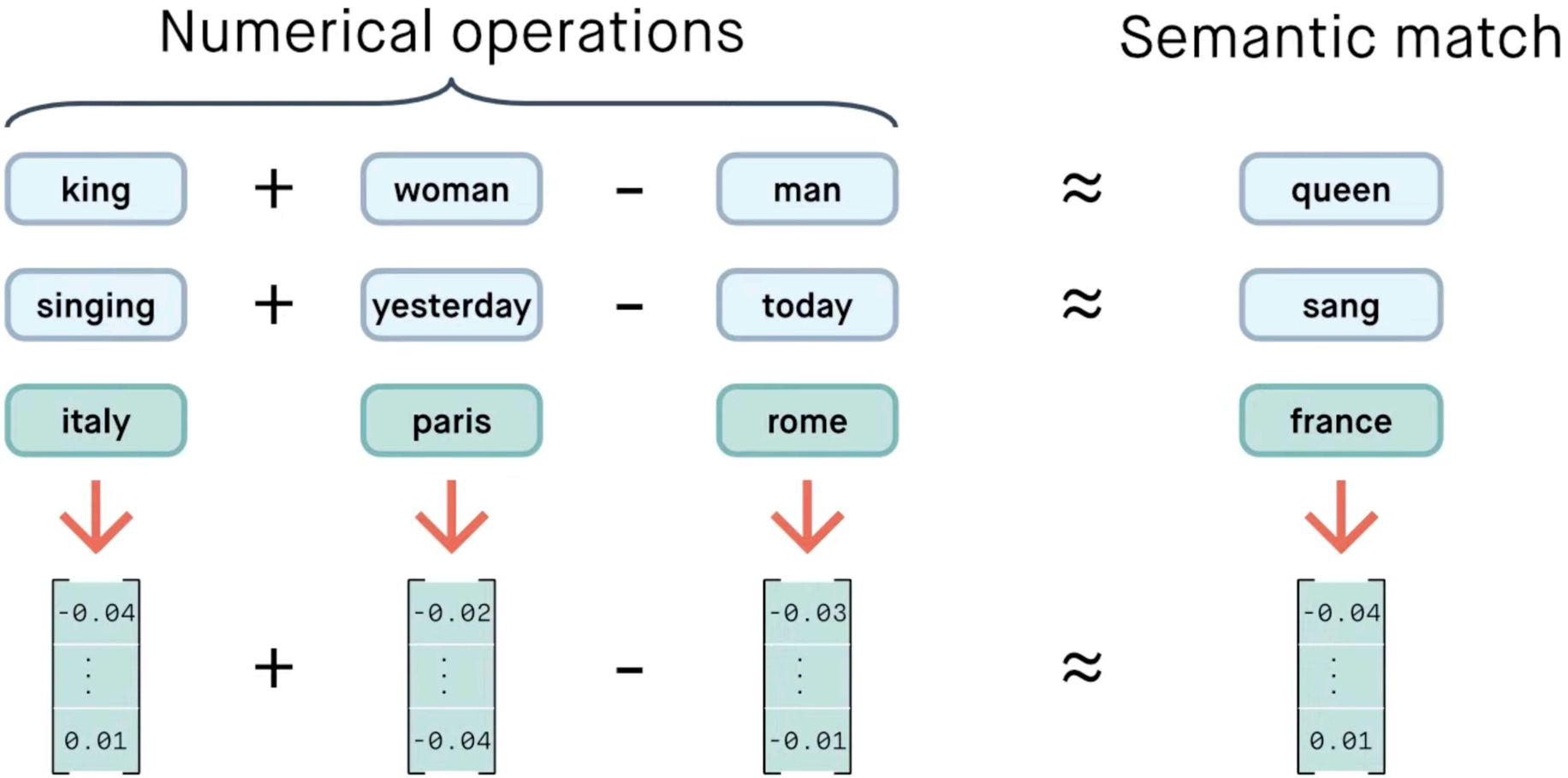
- **Word similarity**: You can compare the embeddings of two words to understand their similarity. This is often done using cosine similarity

- **Word analogy:** Vector arithmetic can be used to solve word analogy tasks

# Transformers

## Self-Attention

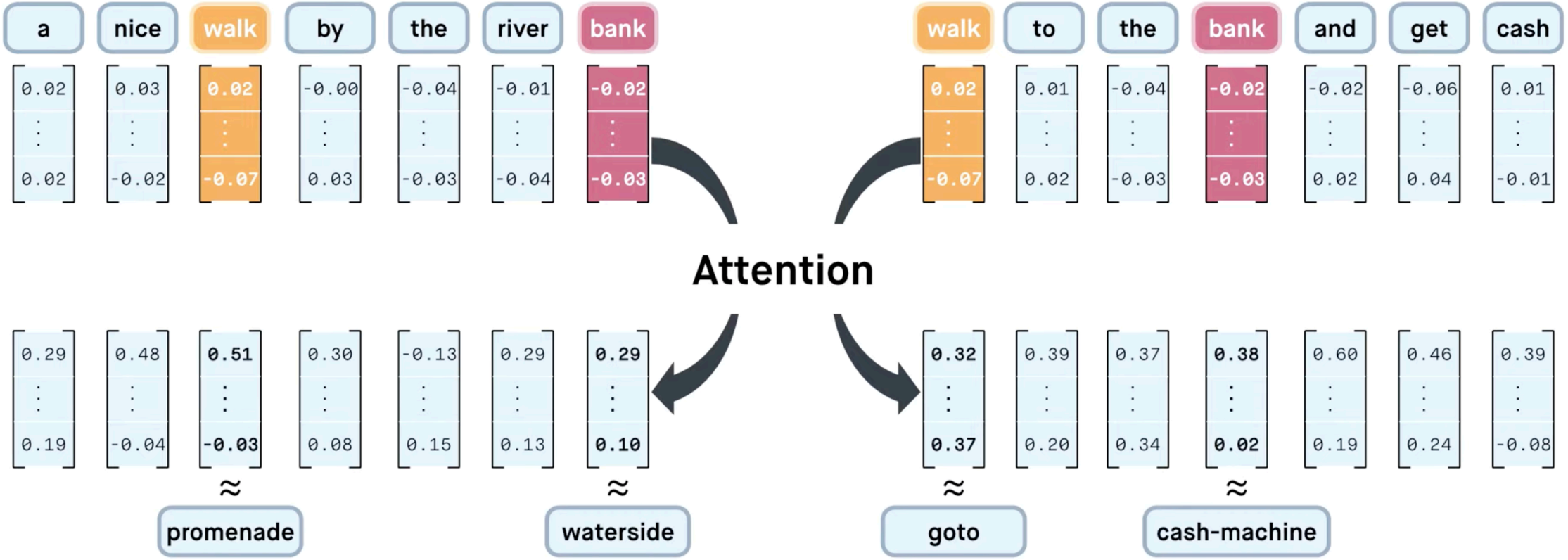
### Contextualized vs. Non-Contextualized Embeddings



# Transformers

## Self-Attention

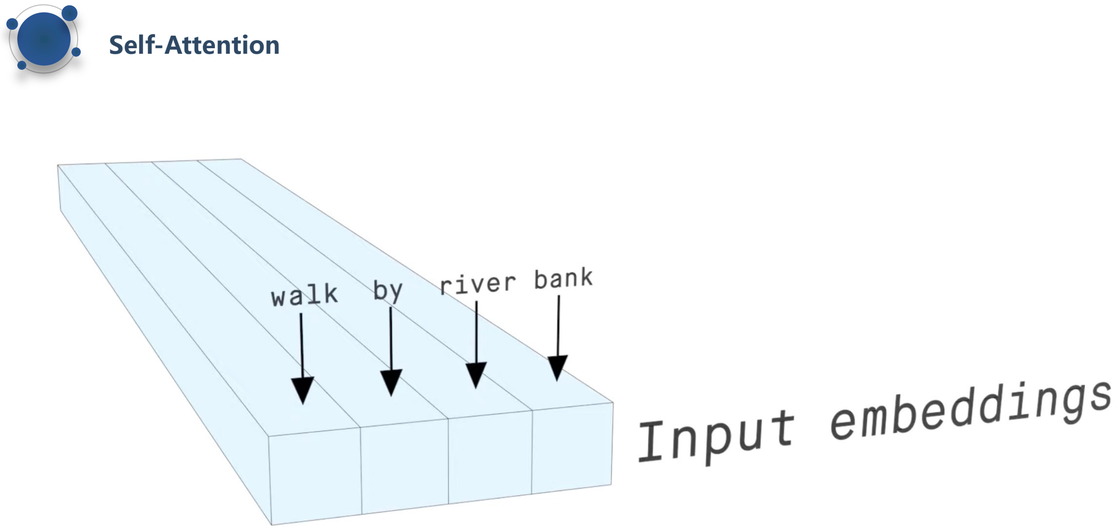
### Contextualized vs. Non-Contextualized Embeddings

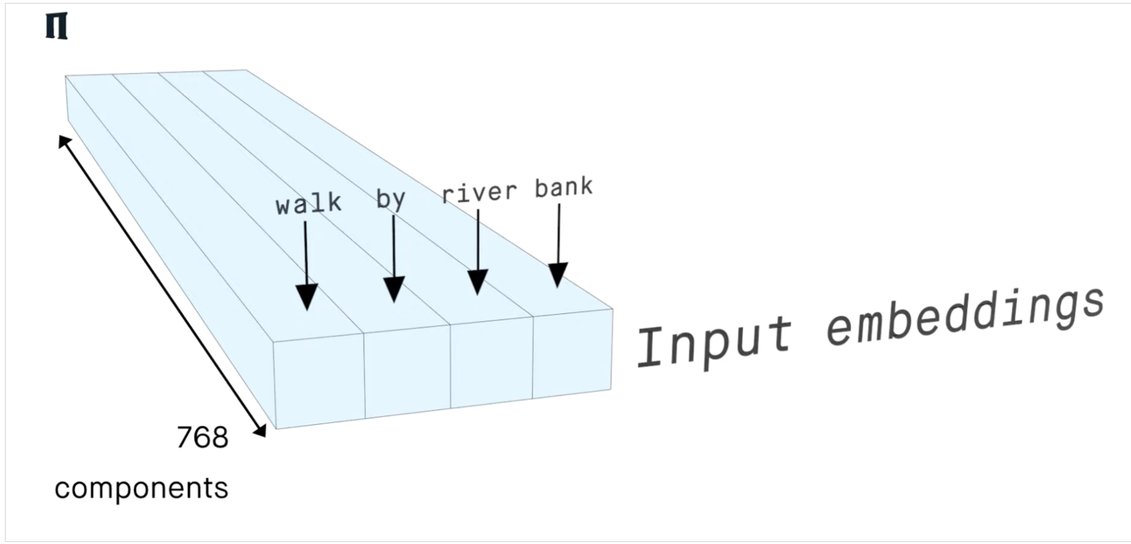


# Transformers

## Self-Attention

## Contextualized vs. Non-Contextualized Embeddings

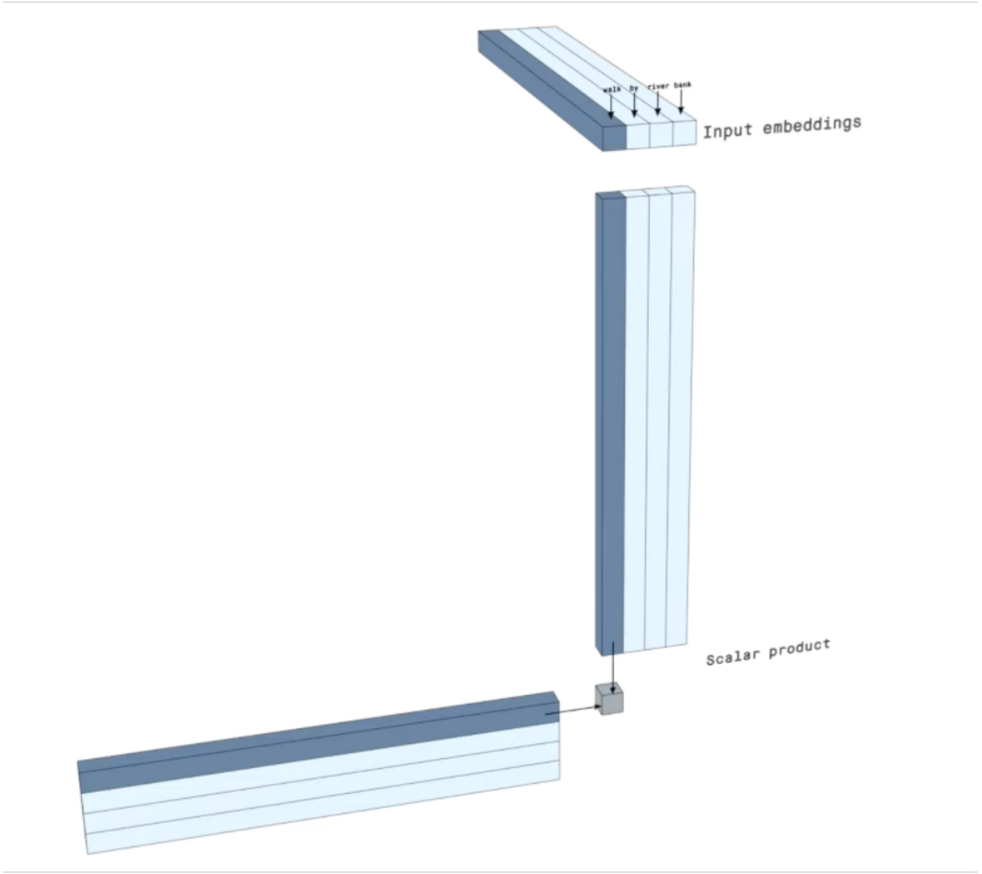


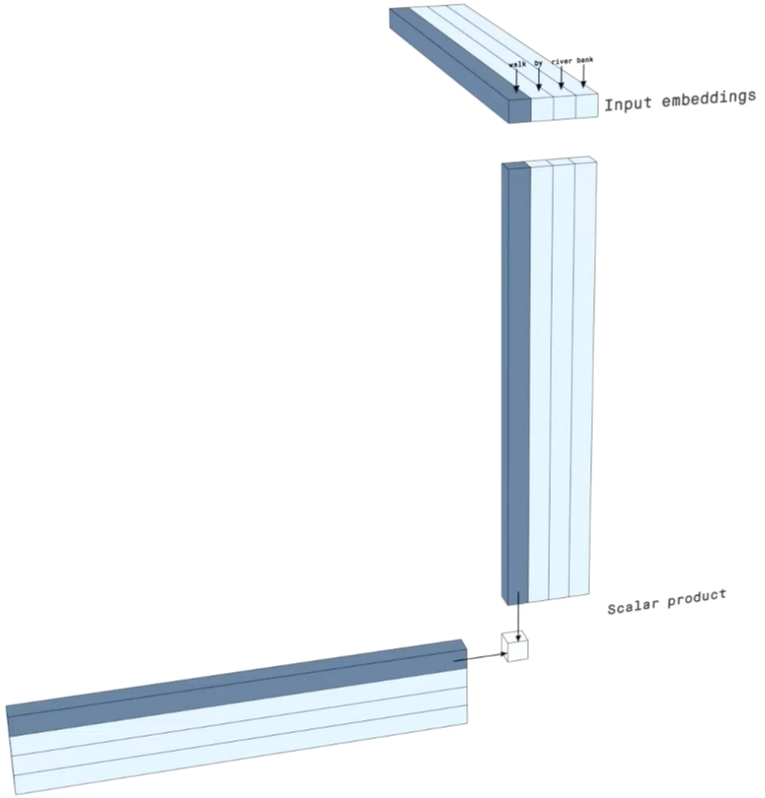



# Transformers

## Self-Attention

## Contextualized vs. Non-Contextualized Embeddings



# Transformers

## Self-Attention

### Scaled Dot-Product Attention

Scaled dot-product attention takes three matrices as input

Similarity is simply the dot product between Q and K

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

The output is simply a scaling of V

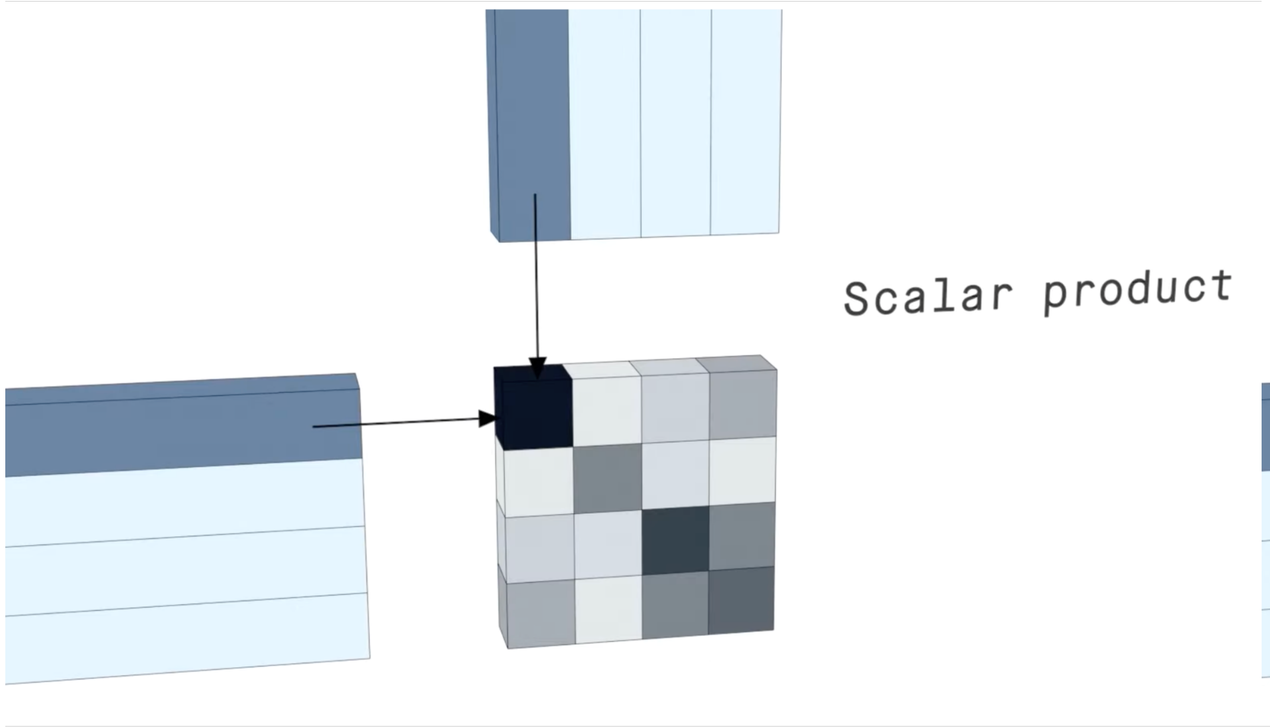
A softmax normalizes similarities → [0, 1]

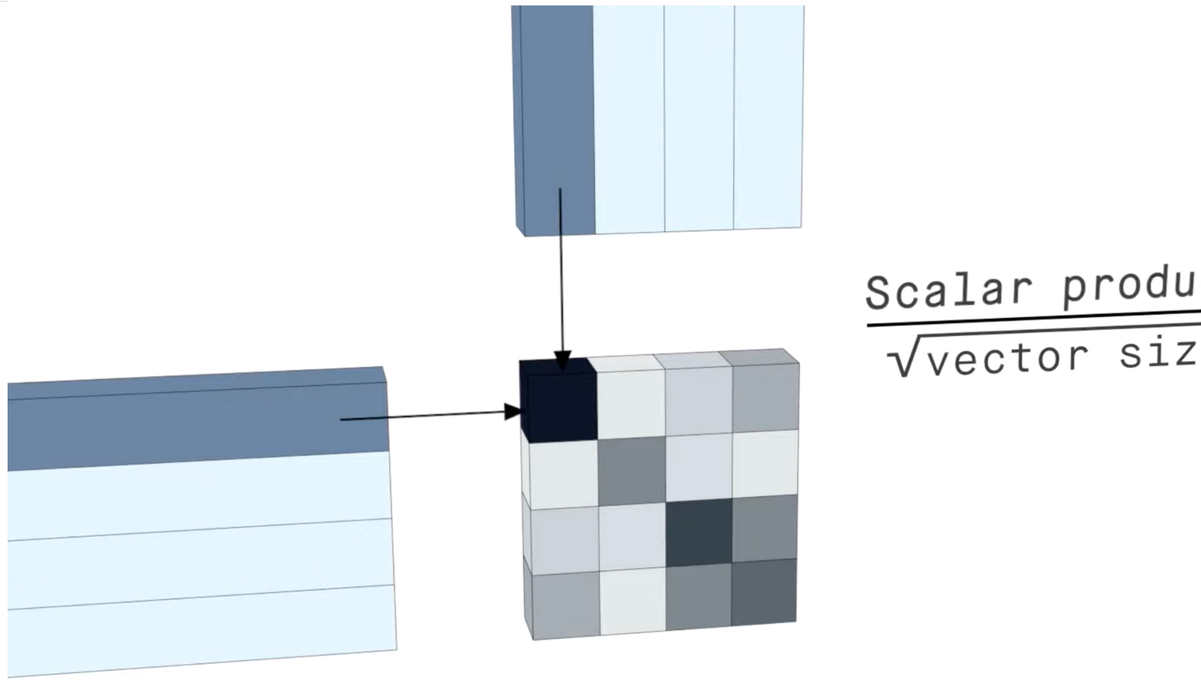
**Attention** maps a query, Q and a set of key-value (K, V) pairs to an output. The output is **computed** as a **weighted sum** of the **values**, where the weight assigned to each value is computed by a **compatibility function** of the **query** with the **corresponding key**.

# Transformers

## Self-Attention

### Contextualized vs. Non-Contextualized Embeddings

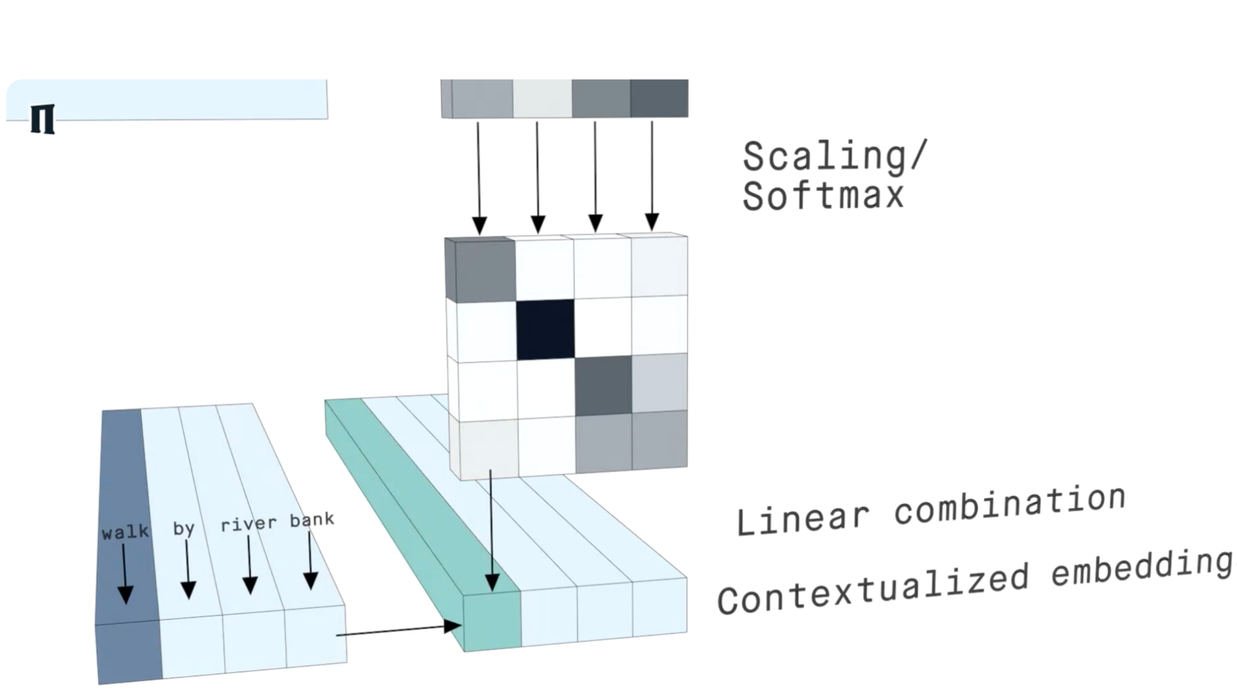


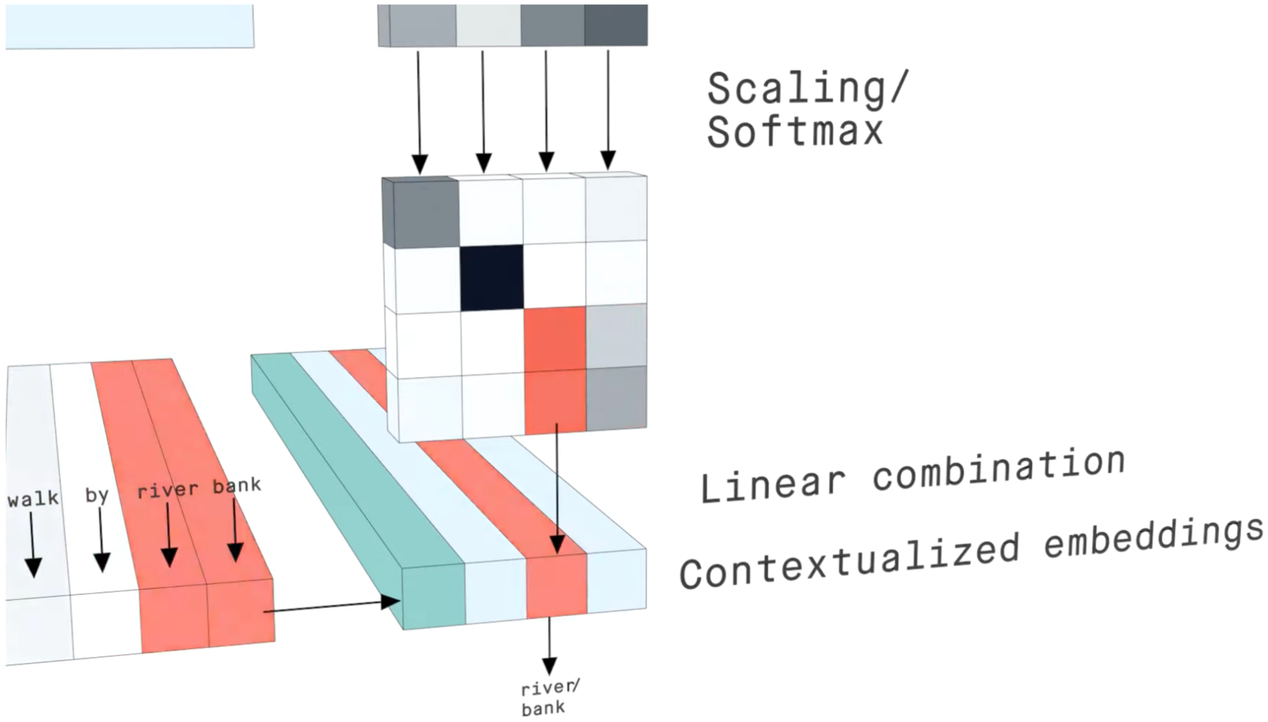



# Transformers

## Self-Attention

### Contextualized vs. Non-Contextualized Embeddings

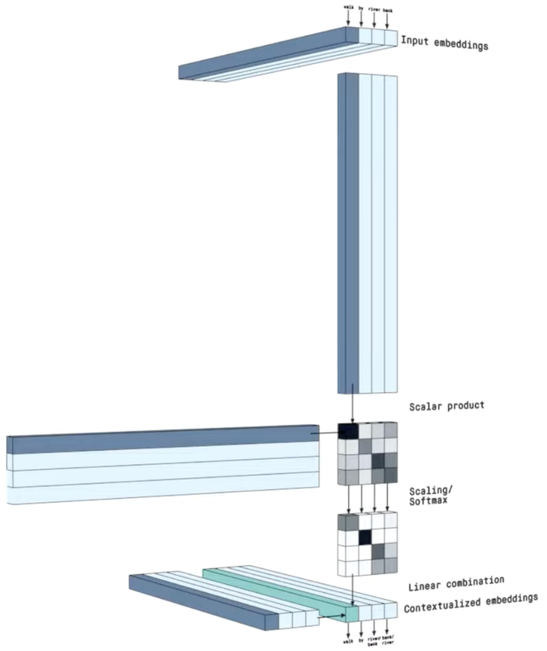


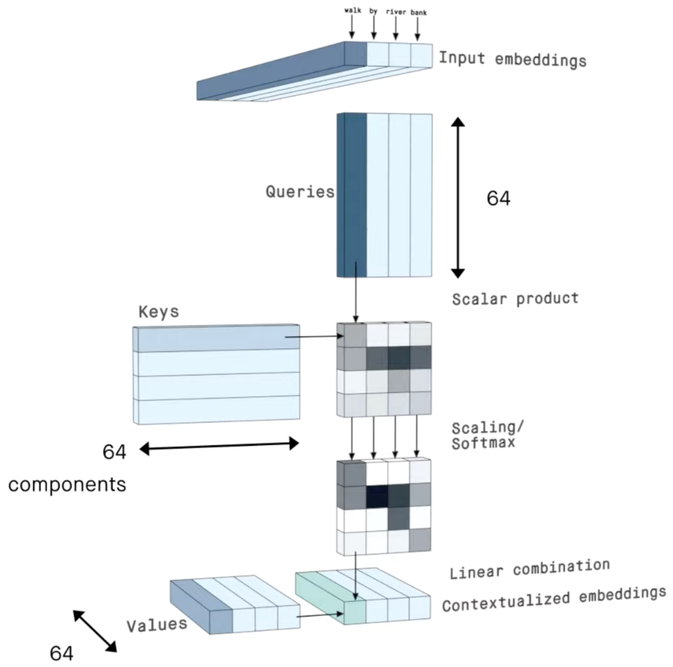



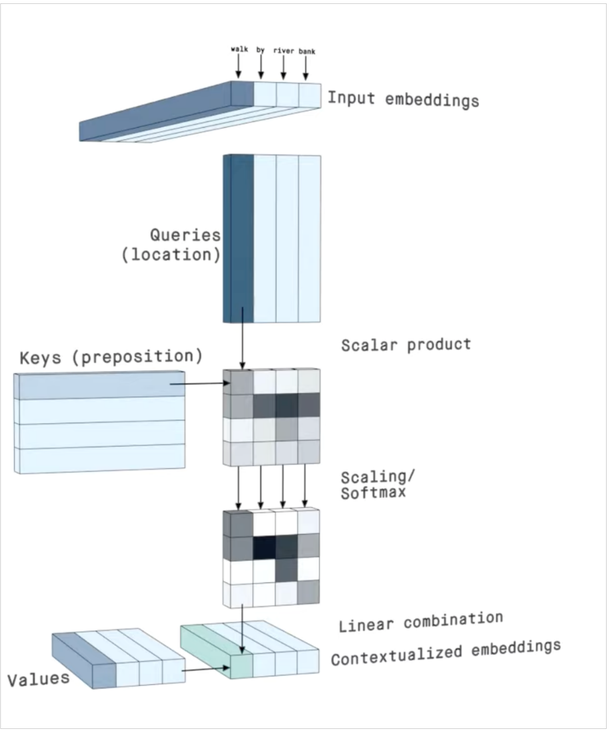
# Transformers

## Self-Attention

## Contextualized vs. Non-Contextualized Embeddings

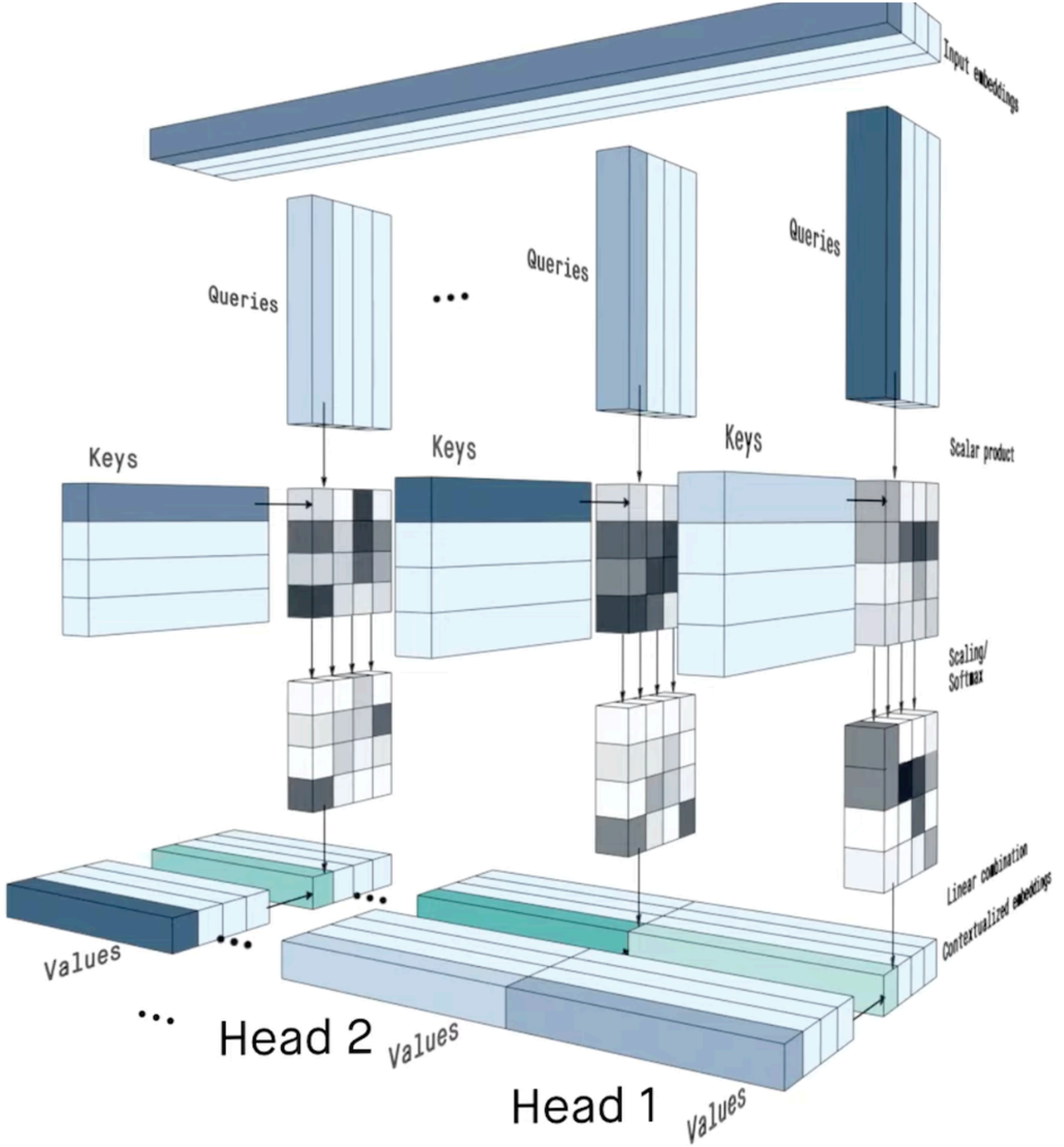


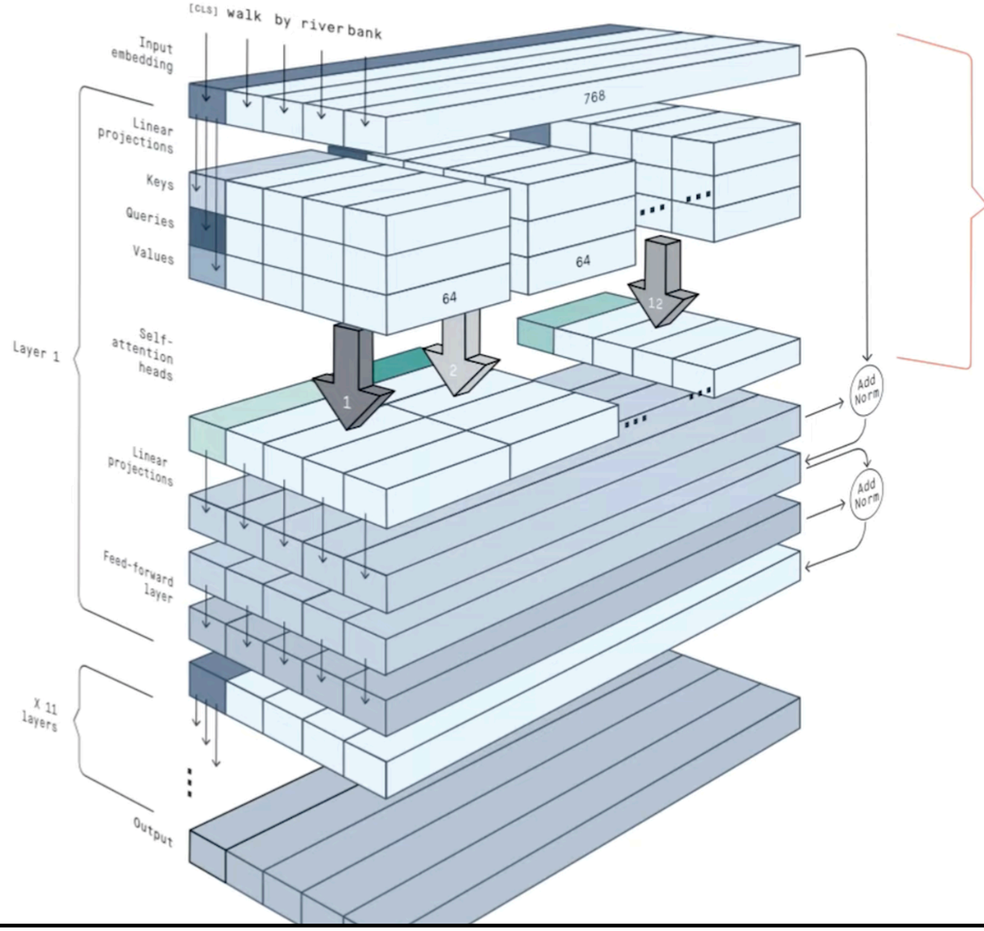
# Transformers

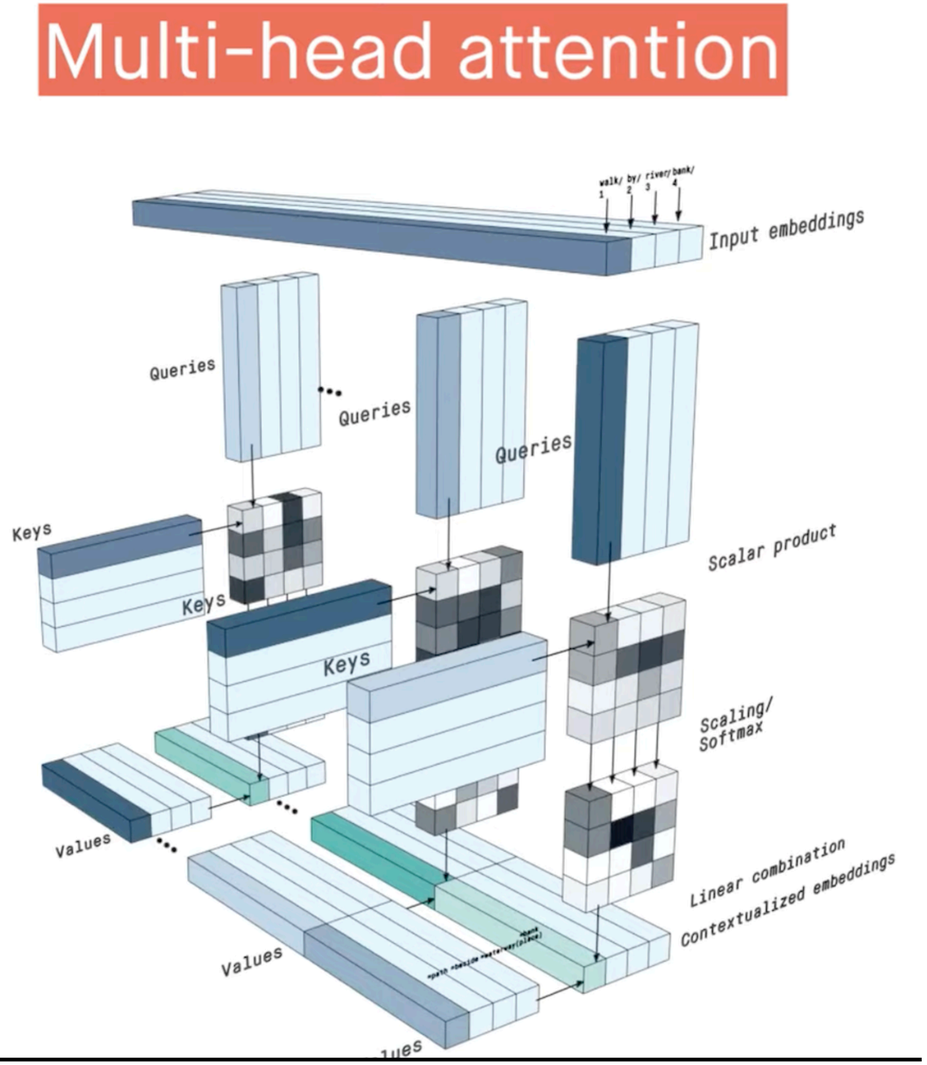
## Self-Attention



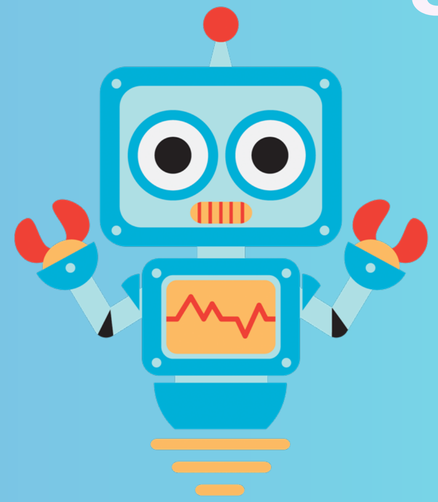
# Transformers

## Self-Attention



# Pretrained Language Models and Self-Supervised Learning



# LLM

## Wishlist

**We want to train transformers as LMs**

Learn general properties of language that can be transferred to **downstream** tasks

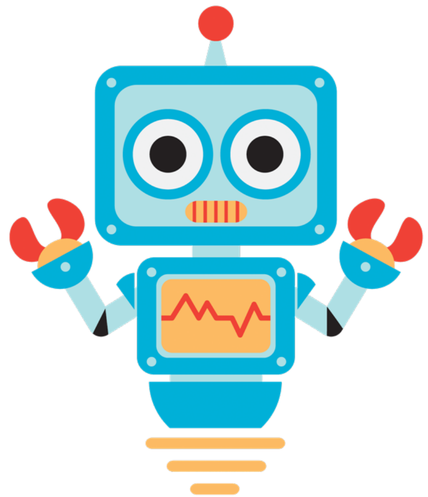
**Ideally, we could train LMs using unlabeled text**

Leverage unsupervised or Self-Supervised Learning (SSL)

- **(At least) two paradigms have emerged**

**G**enerative **P**retrained **T**ransformer (**GPT**)
- Next-token prediction, decoder only transformer

**B**idirectional **E**ncoder **R**epresentations from **T**ransformers (**BERT**)
- Masked language modelling, encoder only transformer



# LLM

## Generative Pretrained Transformer (GPT)

Once pretrained, GPT can be used for any “text in, text out” task

### Grammar correction

Transformation | Generation

Corrects sentences into standard English.

**Prompt**

```
Correct this to standard English:

She no went to the market.
```

**Sample response**

```
She didn't go to the market.
```

### English to other languages

Transformation | Generation

Translates English text into French, Spanish and Japanese.

**Prompt**

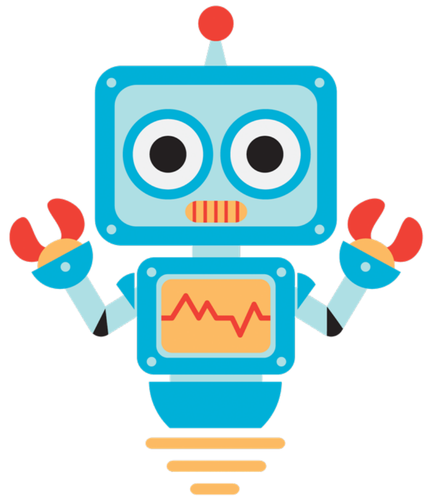
```
Translate this into 1. French, 2. Spanish and 3. Japanese:

What rooms do you have available?

1.
```

**Sample response**

```
Quels sont les chambres disponibles?
2. ¿Cuáles son las habitaciones disponibles?
3. 何室がありますか?
```



https://beta.openai.com/examples

# LLM

## Generative Pretrained Transformer (GPT)

Once pretrained, GPT can be used for any “text in, text out” task

### Grammar correction

Transformation | Generation

Corrects sentences into standard English.

**Prompt**

```
Correct this to standard English:

She no went to the market.
```

**Sample response**

```
She didn't go to the market.
```

### English to other languages

Transformation | Generation

Translates English text into French, Spanish and Japanese.

**Prompt**

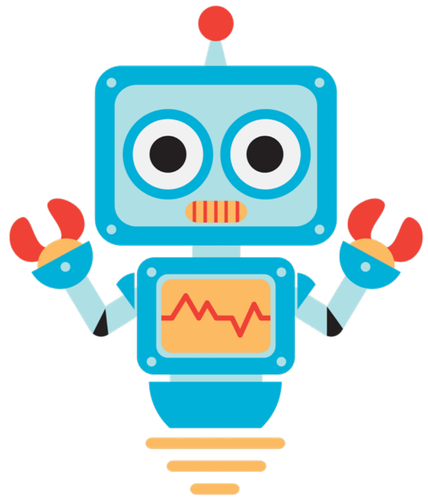
```
Translate this into 1. French, 2. Spanish and 3. Japanese:

What rooms do you have available?

1.
```

**Sample response**

```
Quels sont les chambres disponibles?
2. ¿Cuáles son las habitaciones disponibles?
3. 何室がありますか?
```



https://beta.openai.com/examples

# LLMs

## Bidirectional Encoder Representations from Transformers (BERT)

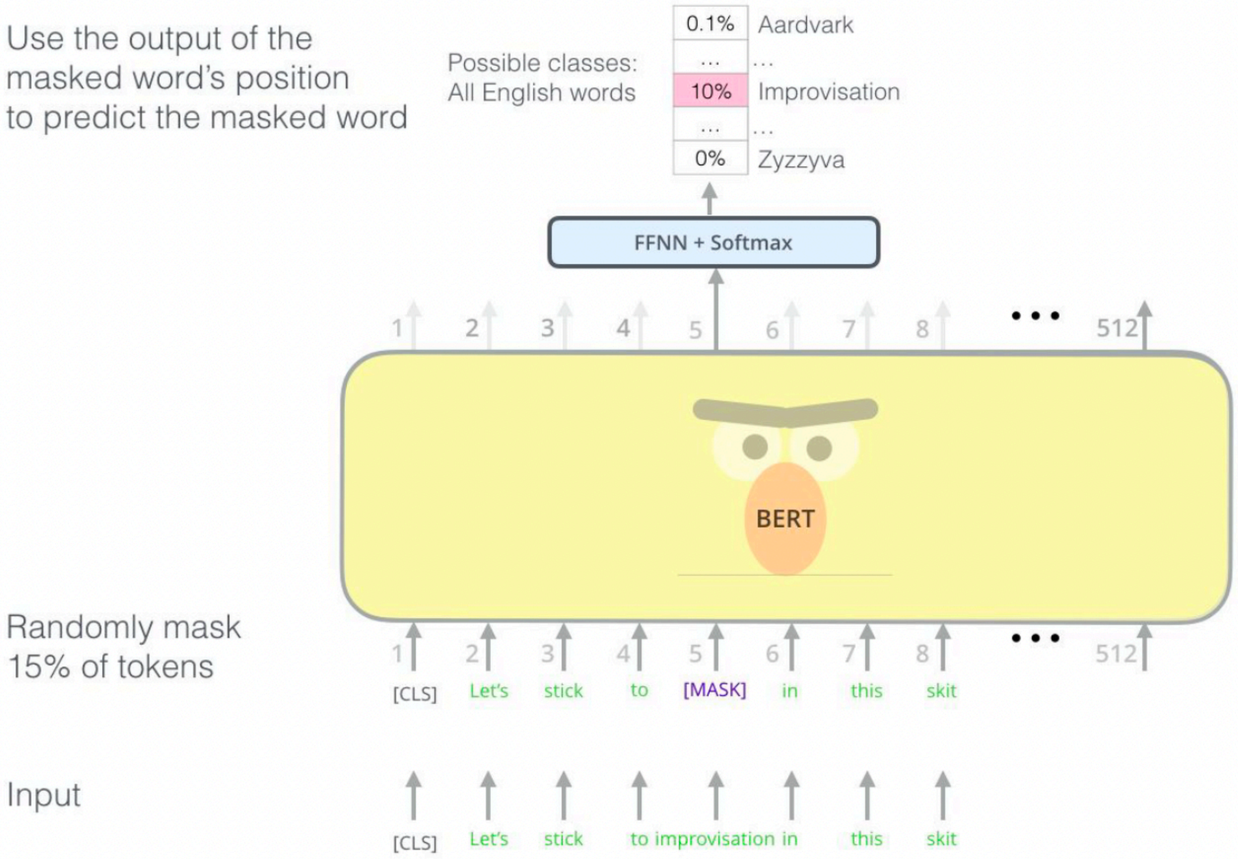
GPT is a unidirectional LM, incorporating context from previous tokens

- This is likely sub-optimal for many token- or sentence-level tasks
- BERT proposes a **bidirectional** LM based on a transformer **encoder**

- BERT is pretrained with two self-supervised objectives:
  - Masked Language Modelling (MLM)
  - Next Sentence Prediction (NSP)

# LLMs

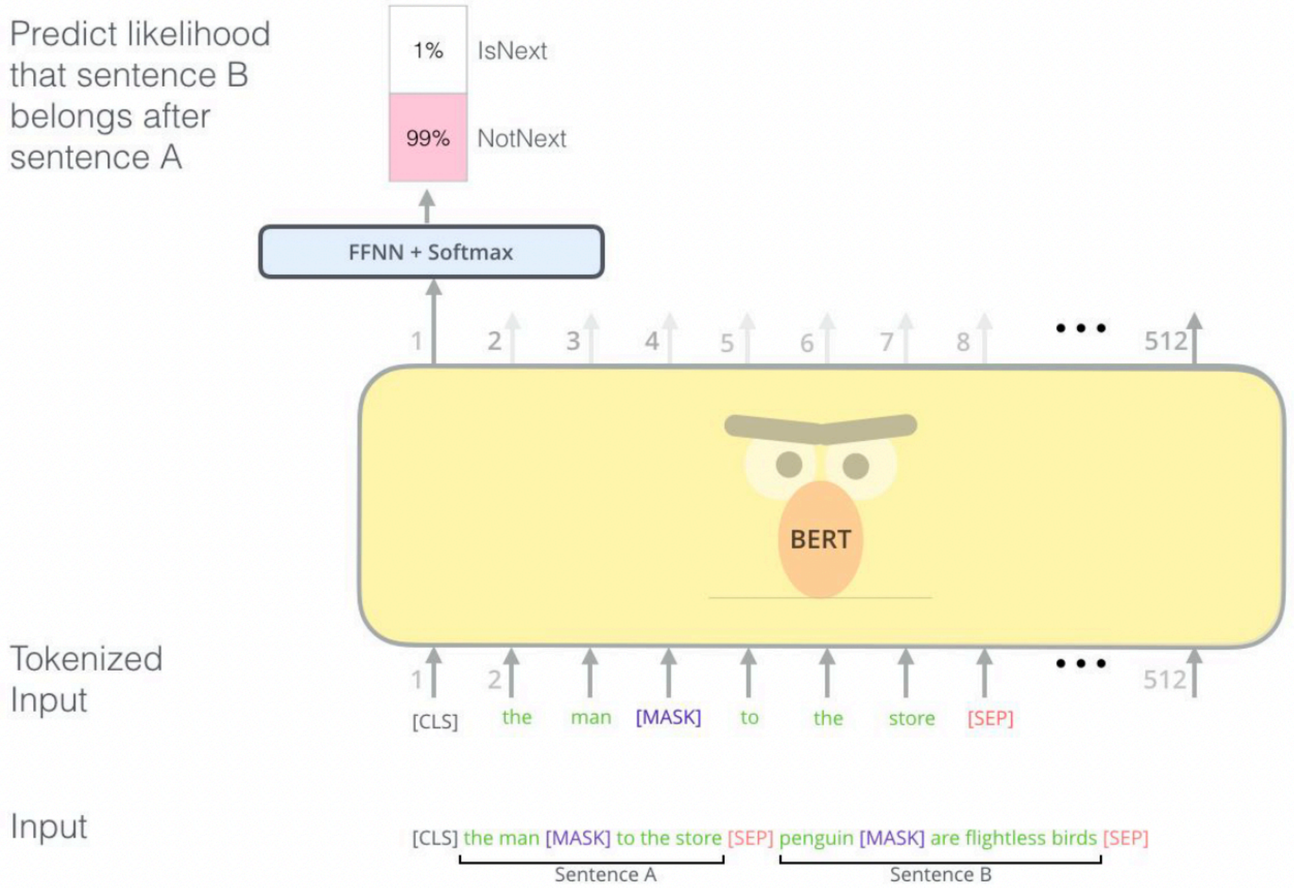
## Masked Language Modelling (MLM)




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

# LLMs

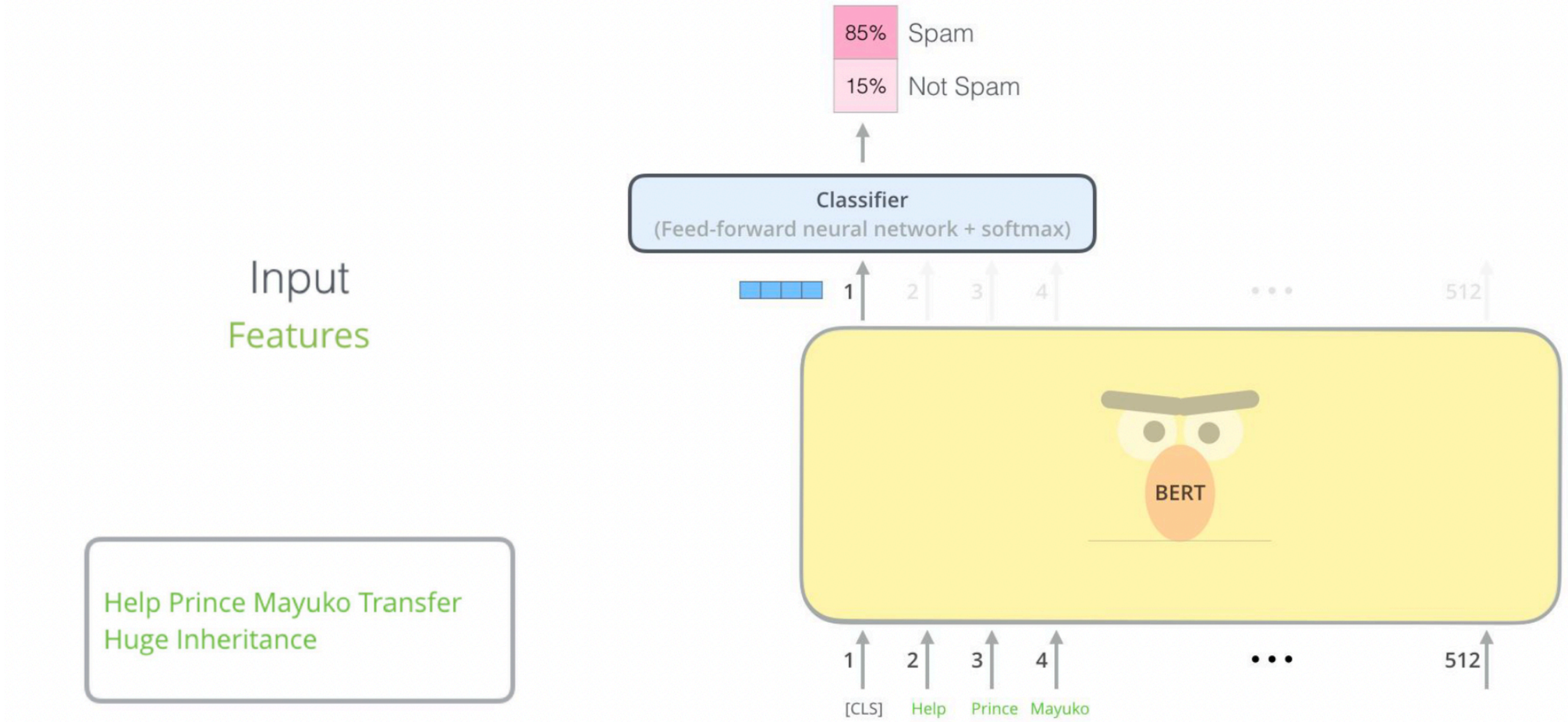
## Masked Language Modelling (MLM)




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

# LLMs

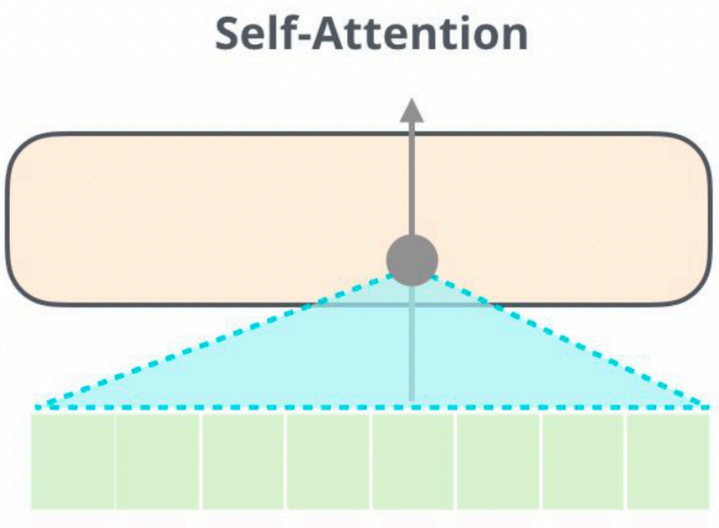
## Fine-tuning BERT

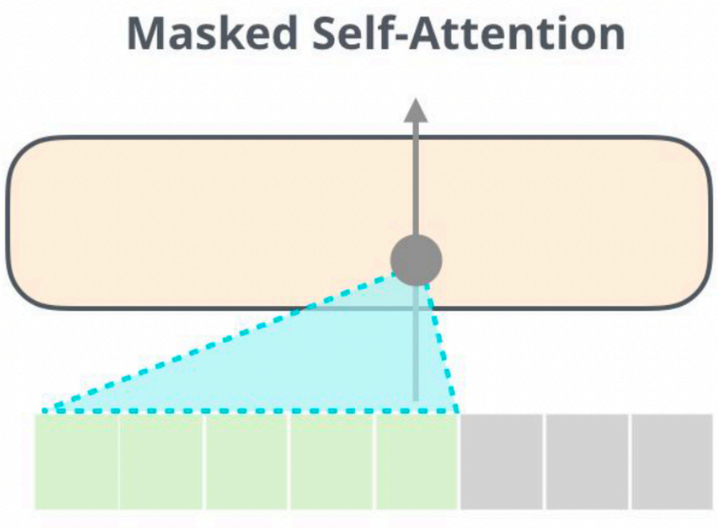


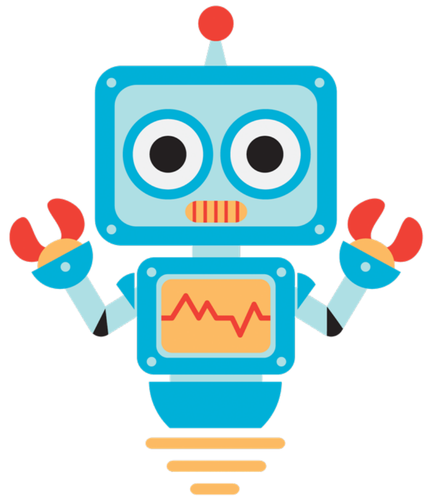

https://jalammar.github.io/illustrated-bert/

# Masked Self-Attention

Future timesteps are masked to prevent decoder from “peaking”



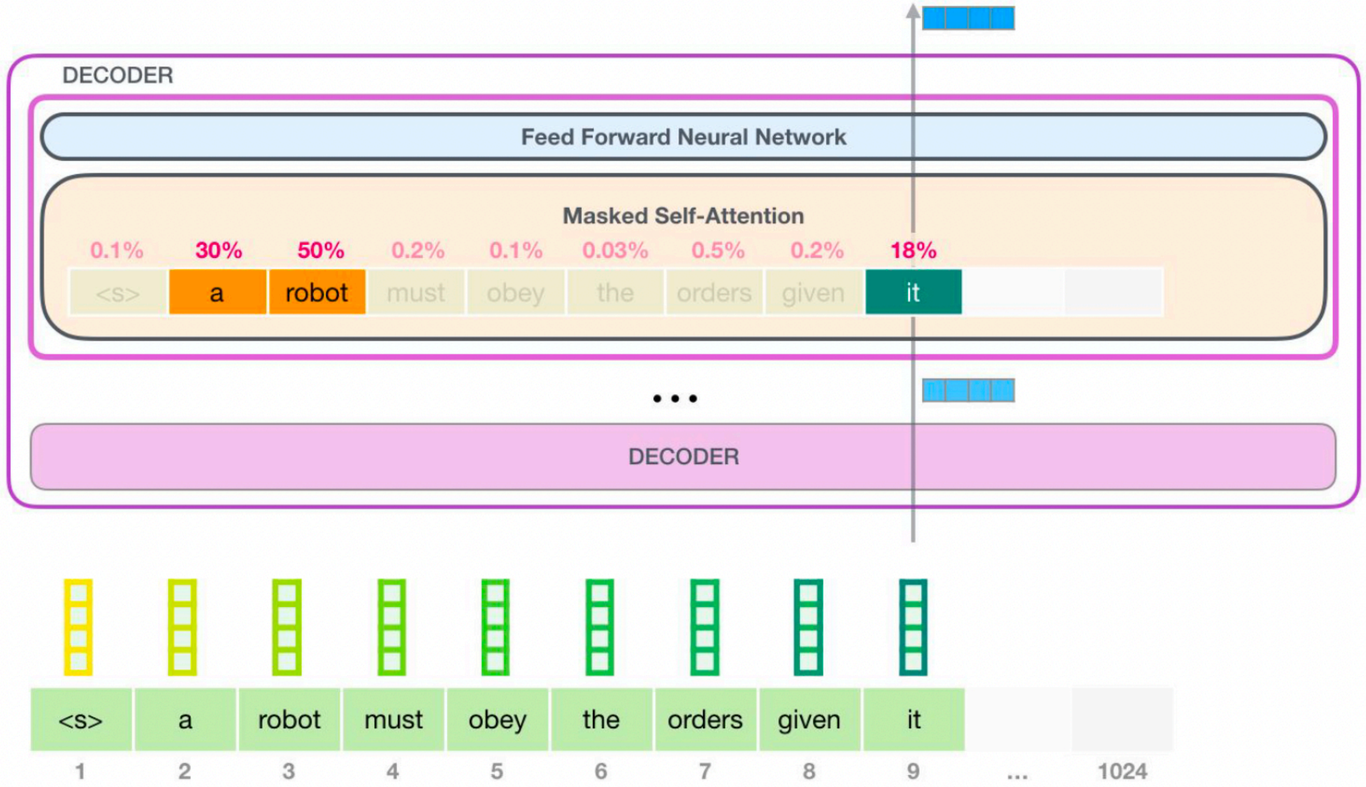



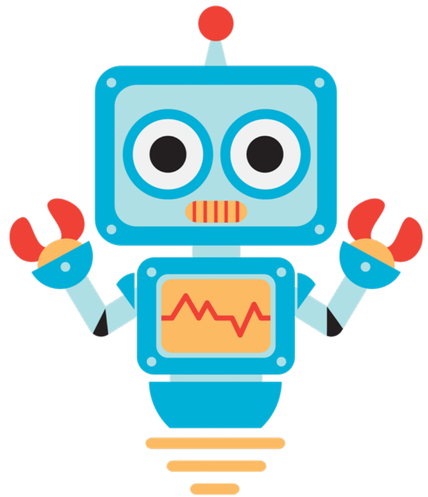



https://jalammar.github.io/illustrated-gpt2/

# Next Token Prediction

Future timesteps are masked to prevent decoder from “peaking”



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

# LLMs

## History

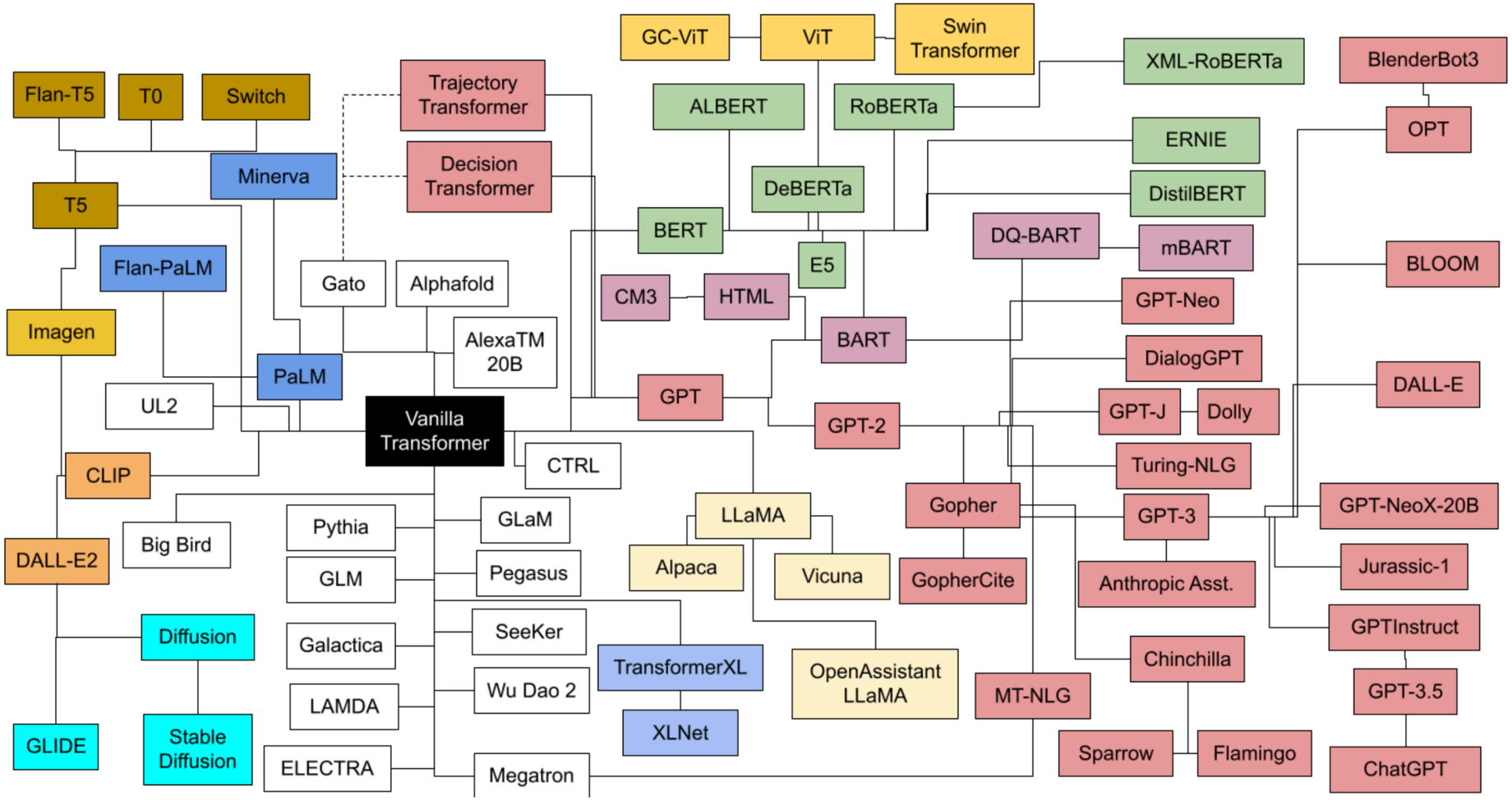



Figure 5: Transformers Family Tree

# LLMs

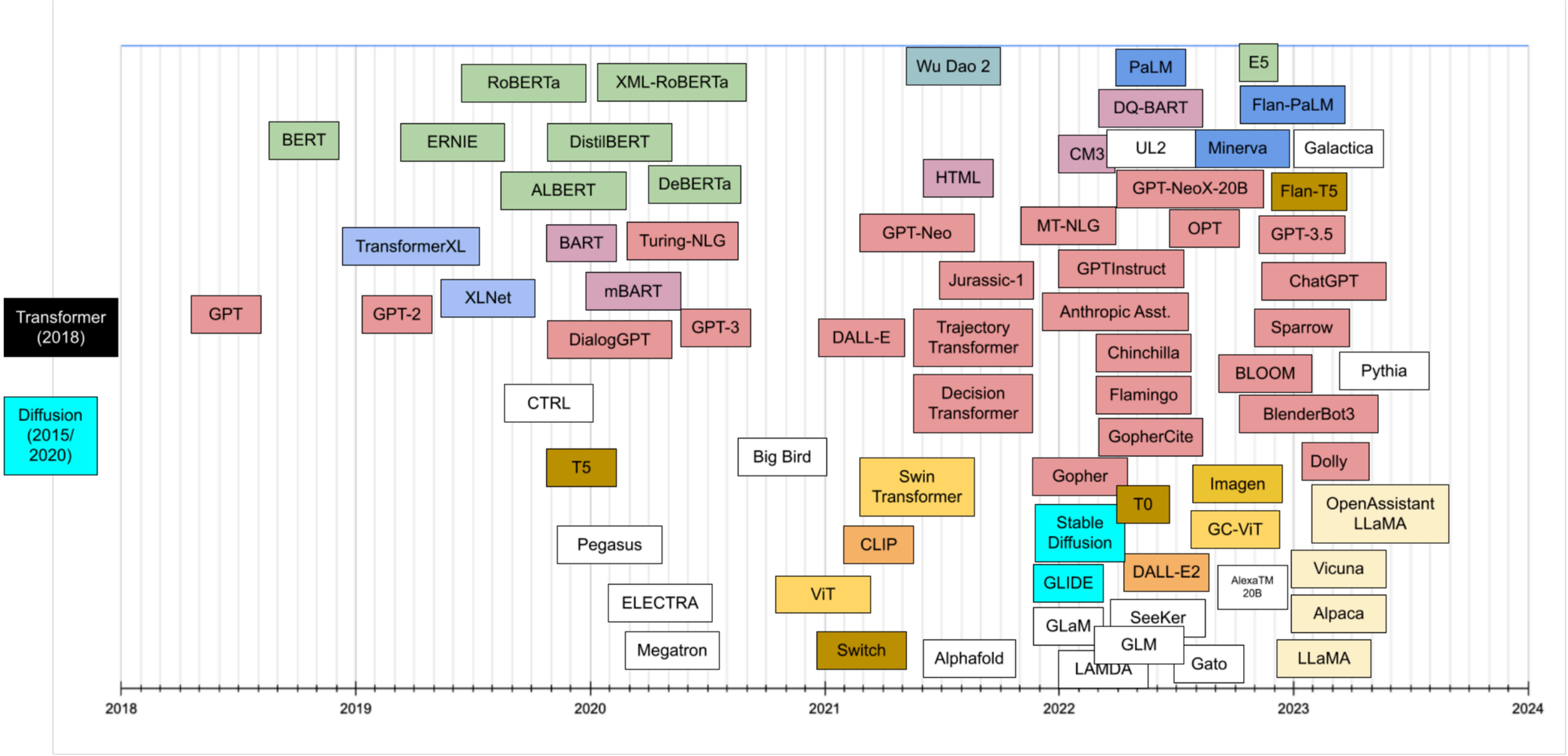
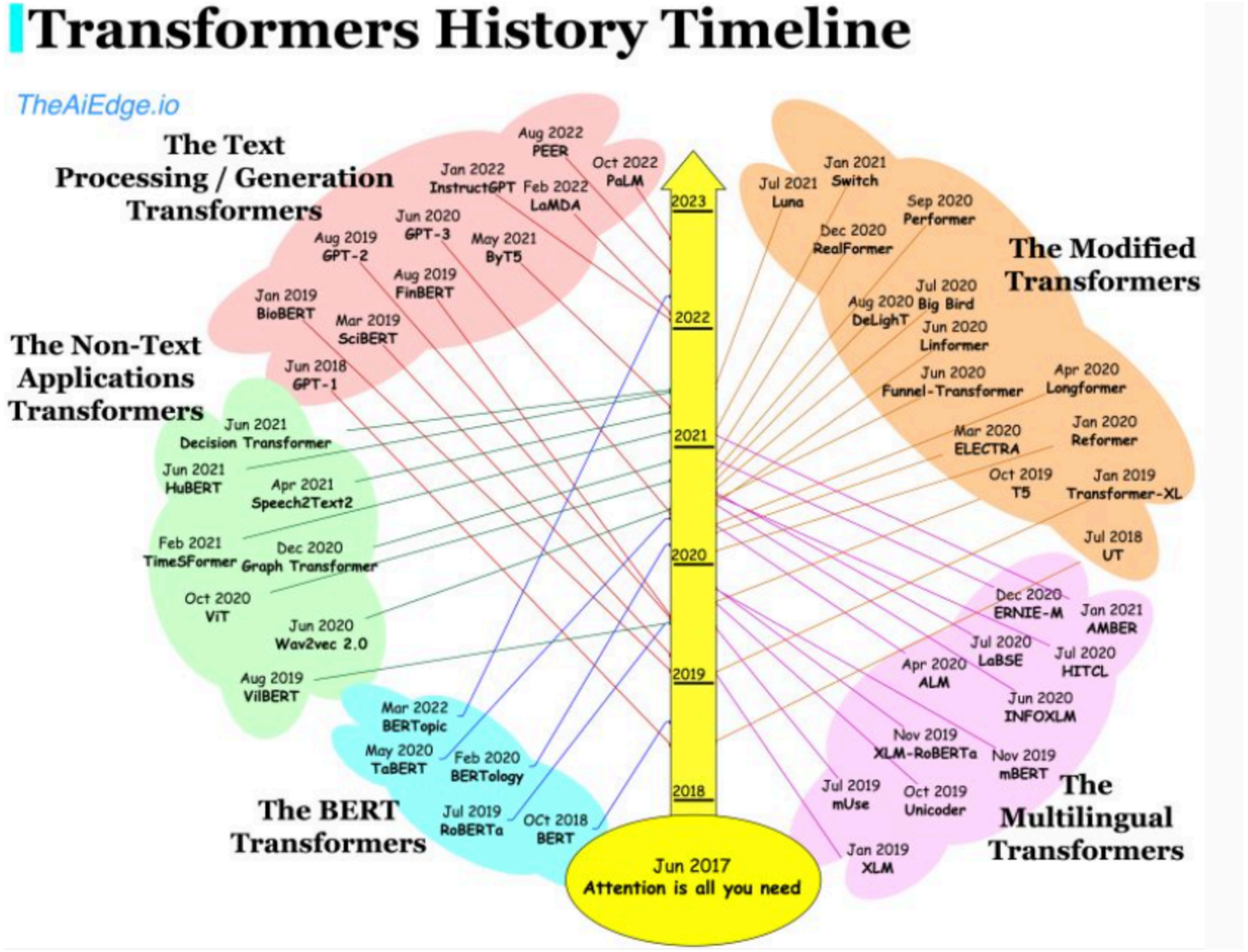



Figure 6: Transformer timeline. Colors describe Transformer family.

# LLMs

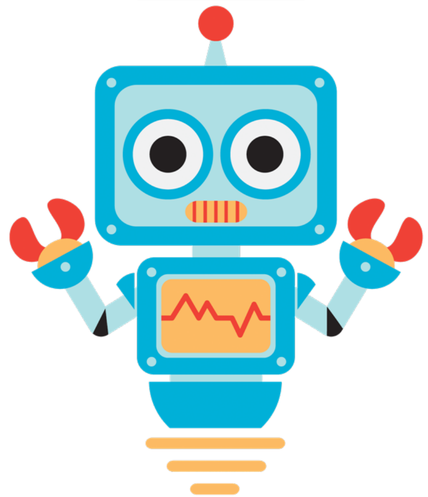

Transformers History Timeline
TheAiEdge.io
The Text Processing / Generation Transformers
The Non-Text Applications Transformers
The BERT Transformers
The Modified Transformers
The Multilingual Transformers
2023
2022
2021
2020
2019
2018
Jun 2017
Attention is all you need
Aug 2022 PEER
Oct 2022 PaLM
Jan 2022 InstructGPT
Feb 2022 LaMDA
Jun 2020 GPT-3
Aug 2019 GPT-2
May 2021 ByT5
Aug 2019 FinBERT
Jan 2019 BioBERT
Mar 2019 SciBERT
Jun 2018 GPT-1
Jun 2021 Decision Transformer
Jun 2021 HuBERT
Apr 2021 Speech2Text2
Feb 2021 TimeSFormer
Dec 2020 Graph Transformer
Oct 2020 ViT
Jun 2020 Wav2vec 2.0
Aug 2019 VilBERT
Mar 2022 BERTopic
May 2020 TaBERT
Feb 2020 BERTology
Jul 2019 RoBERTa
Oct 2018 BERT
Jan 2021 Switch
Jul 2021 Luna
Sep 2020 Performer
Dec 2020 RealFormer
Jul 2020 Big Bird
Aug 2020 DeLighT
Jun 2020 Linformer
Jun 2020 Funnel-Transformer
Apr 2020 Longformer
Mar 2020 ELECTRA
Jan 2020 Reformer
Oct 2019 T5
Jan 2019 Transformer-XL
Jul 2018 UT
Dec 2020 ERNIE-M
Jan 2021 AMBER
Jul 2020 LaBSE
Jul 2020 HITCL
Apr 2020 ALM
Jun 2020 INFOXLM
Nov 2019 XLM-RoBERTa
Nov 2019 mBERT
Jul 2019 mUse
Oct 2019 Unicoder
Jan 2019 XLM

# LLMs

| model | date | developer | # parameters | open source |
|---|---|---|---|---|
| GPT-3 | May 2020 | OpenAI | 175,000,000,000 | No |
| GPT-Neo | Mar 2021 | EleutherAI | 2,700,000,000 | Yes |
| GPT-J | Jun 2021 | EleutherAI | 6,000,000,000 | Yes |
| Megatron-Turing NLG | Oct 2021 | Microsoft & Nvidia | 530,000,000,000 | No |
| Gopher | Dec 2021 | DeepMind | 280,000,000,000 | No |
| LaMDA | Jan 2022 | Google | 137,000,000,000 | No |
| GPT-NeoX | Feb 2022 | EleutherAI | 20,000,000,000 | Yes |
| Chinchilla | Mar 2022 | DeepMind | 70,000,000,000 | No |
| PaLM | Apr 2022 | Google | 540,000,000,000 | No |
| Luminous | Apr 2022 | Aleph Alpha | 70,000,000,000 | No |
| OPT | May 2022 | Meta | 175,000,000,000 | Yes (66bn) |
| BLOOM | Jul 2022 | Hugging Face collaboration | 175,000,000,000 | Yes |
| GPT-3.5 | Nov 2022 | OpenAI | Unknown | No |
| LLaMA | Feb 2023 | Meta | 65,000,000,000 | No |
| GPT-4 | Mar 2023 | OpenAI | Unknown | No |

Poe

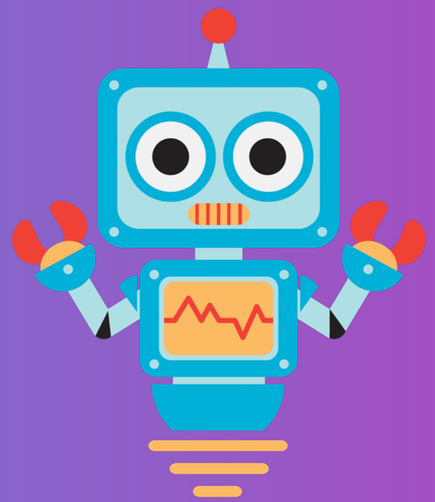
**Mistral 7B**



# Thank you for your attention

## Day 2

January 16th 2024



GAME Khoot