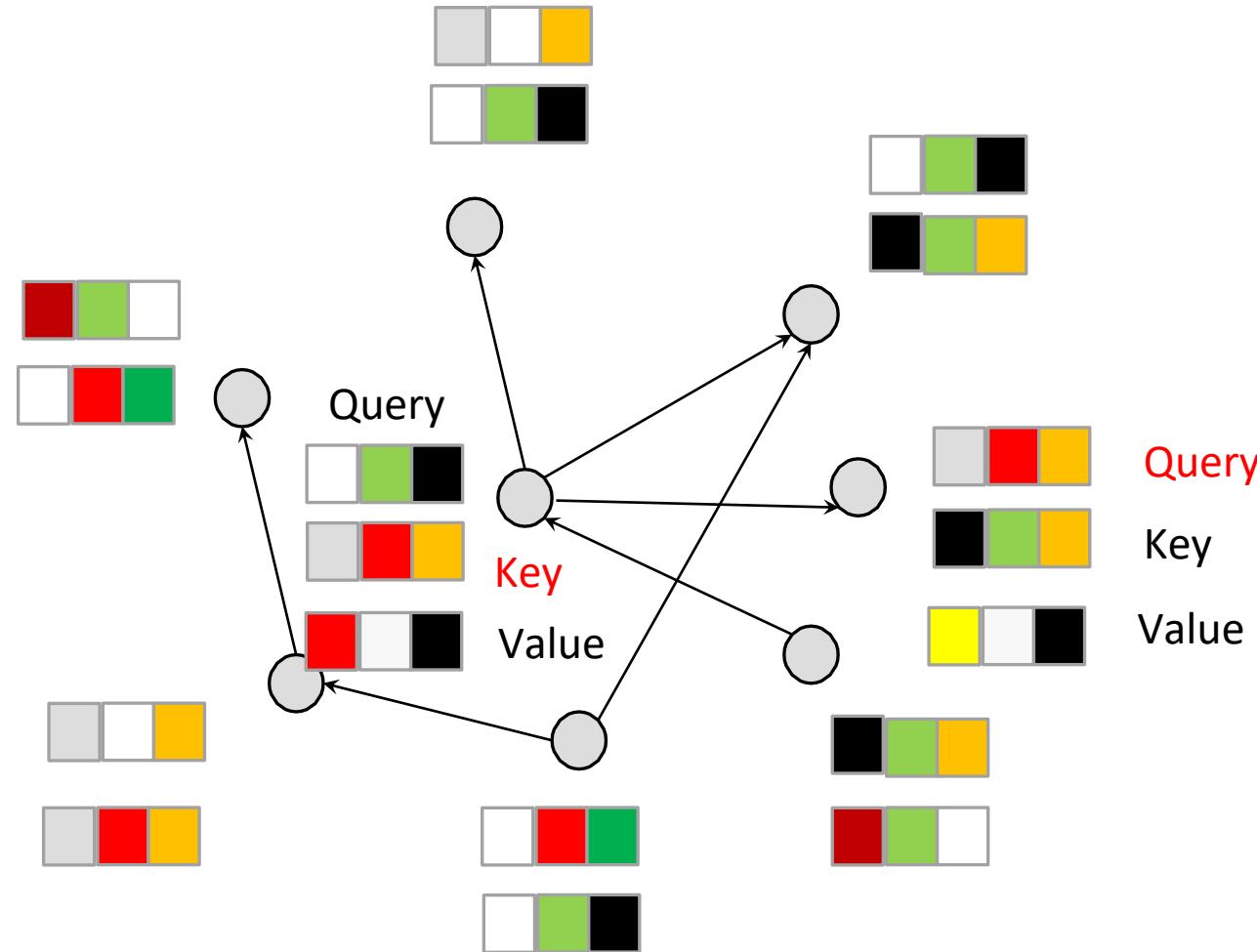
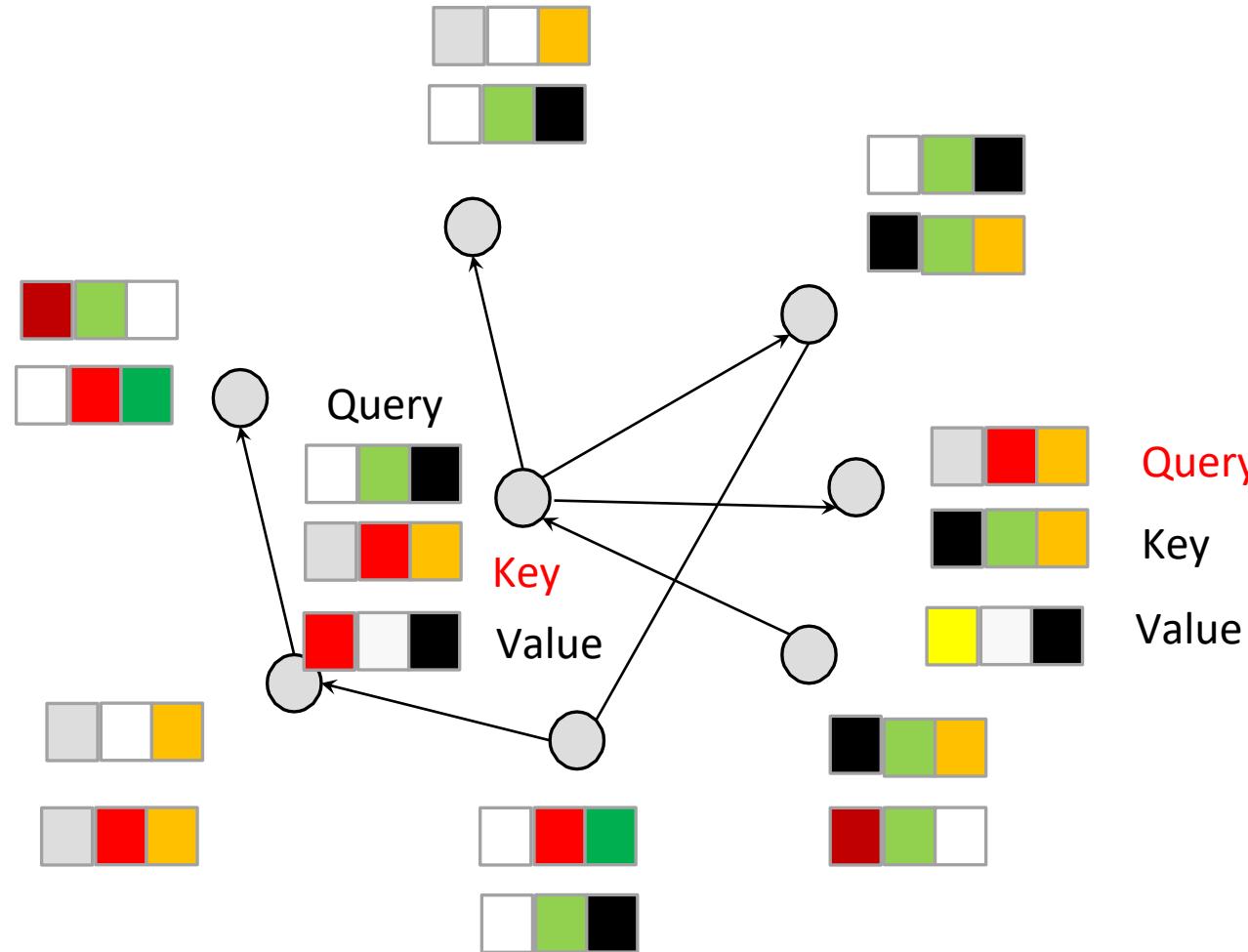


# 怎样构造一个有向网络?

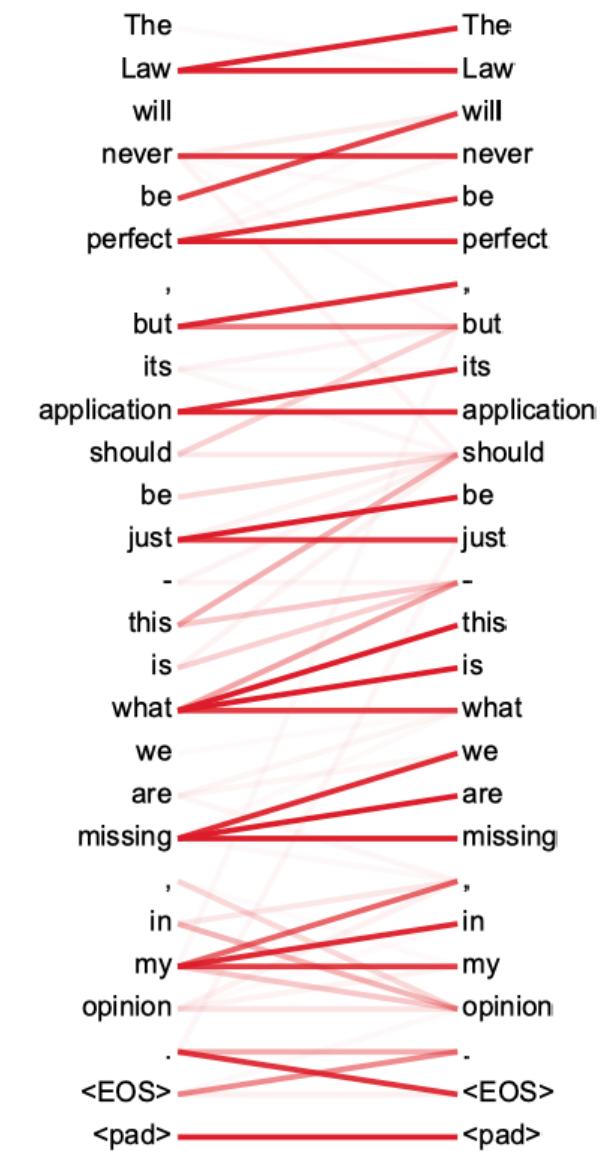
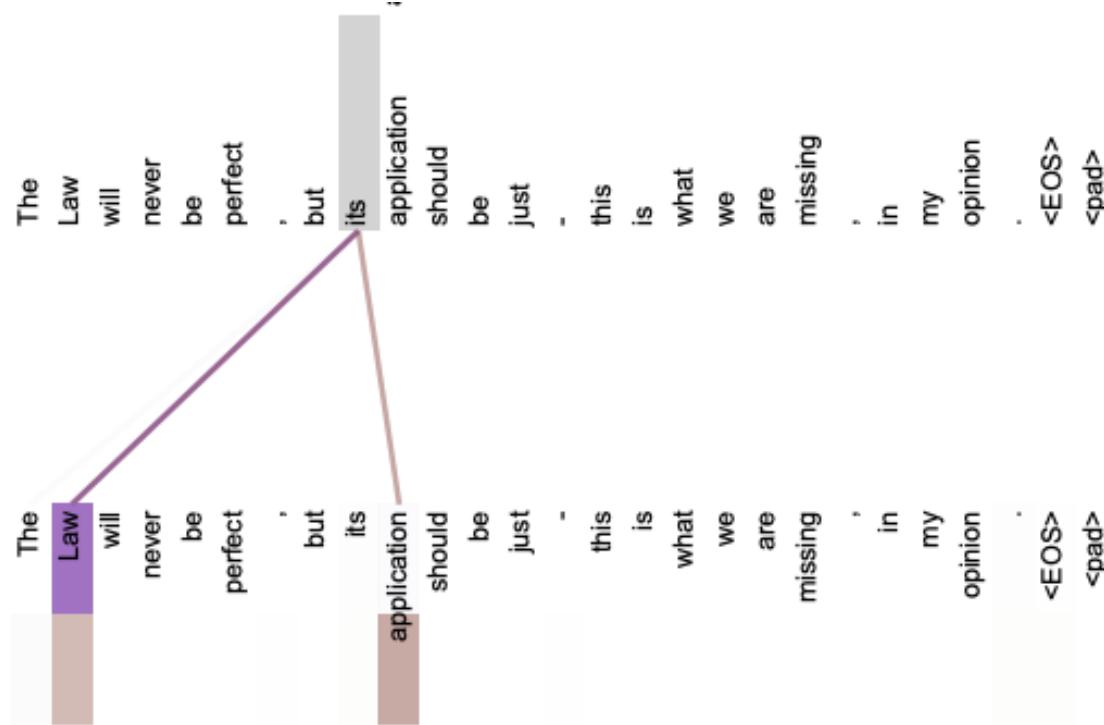


$$x_{j \rightarrow i} = a_{i \rightarrow j} v_j$$

# 如何提取信息?

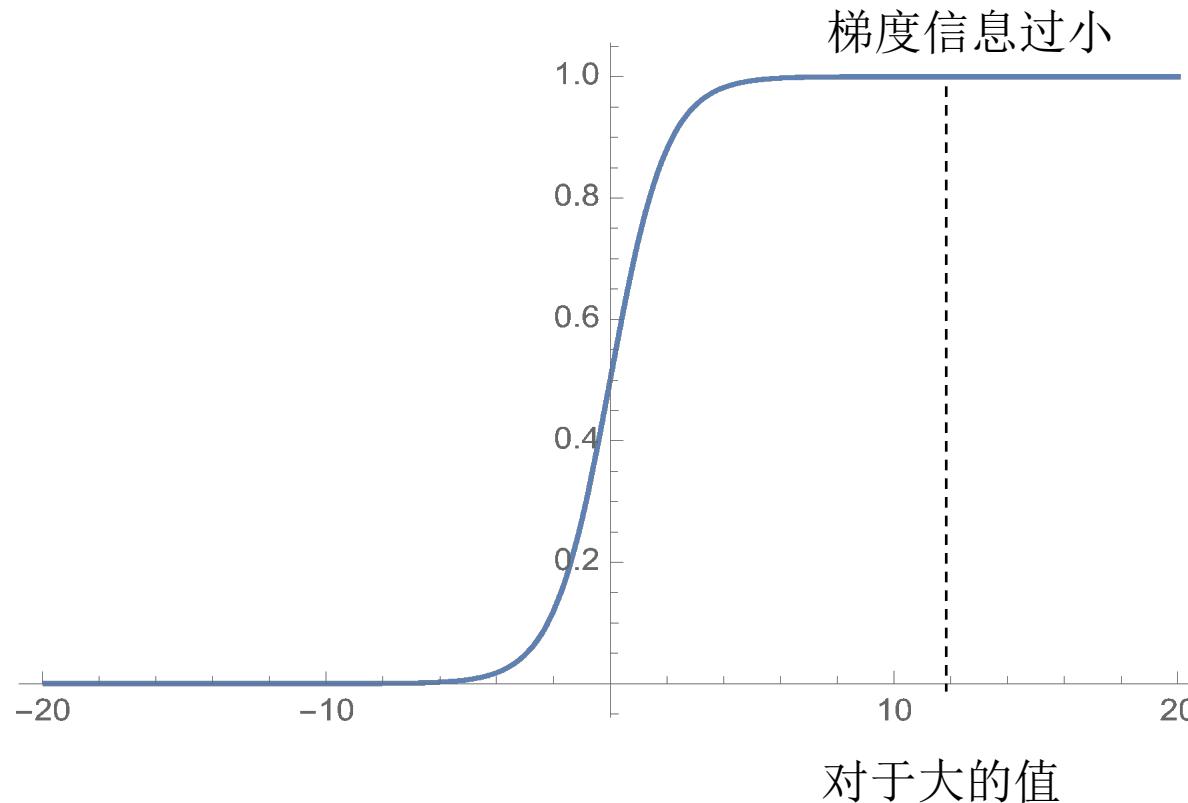


# Self attention effect



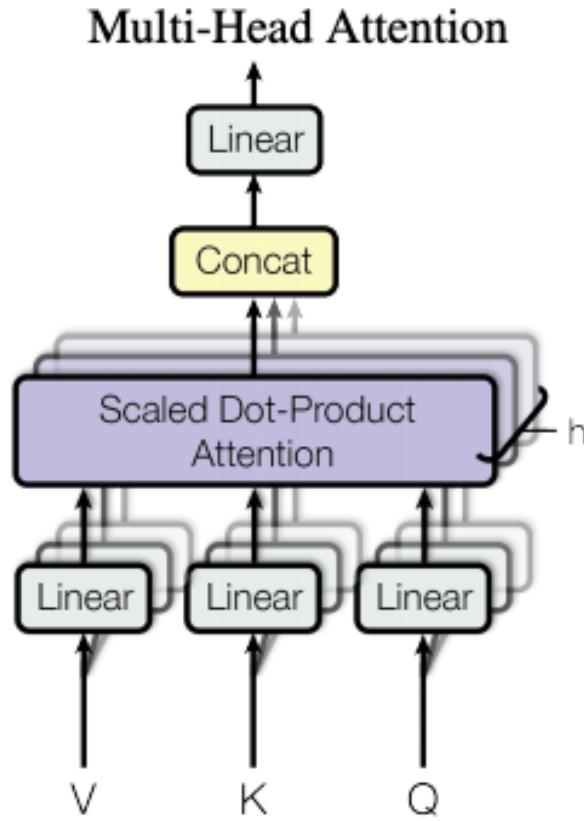


$\sqrt{d_k}$ ?



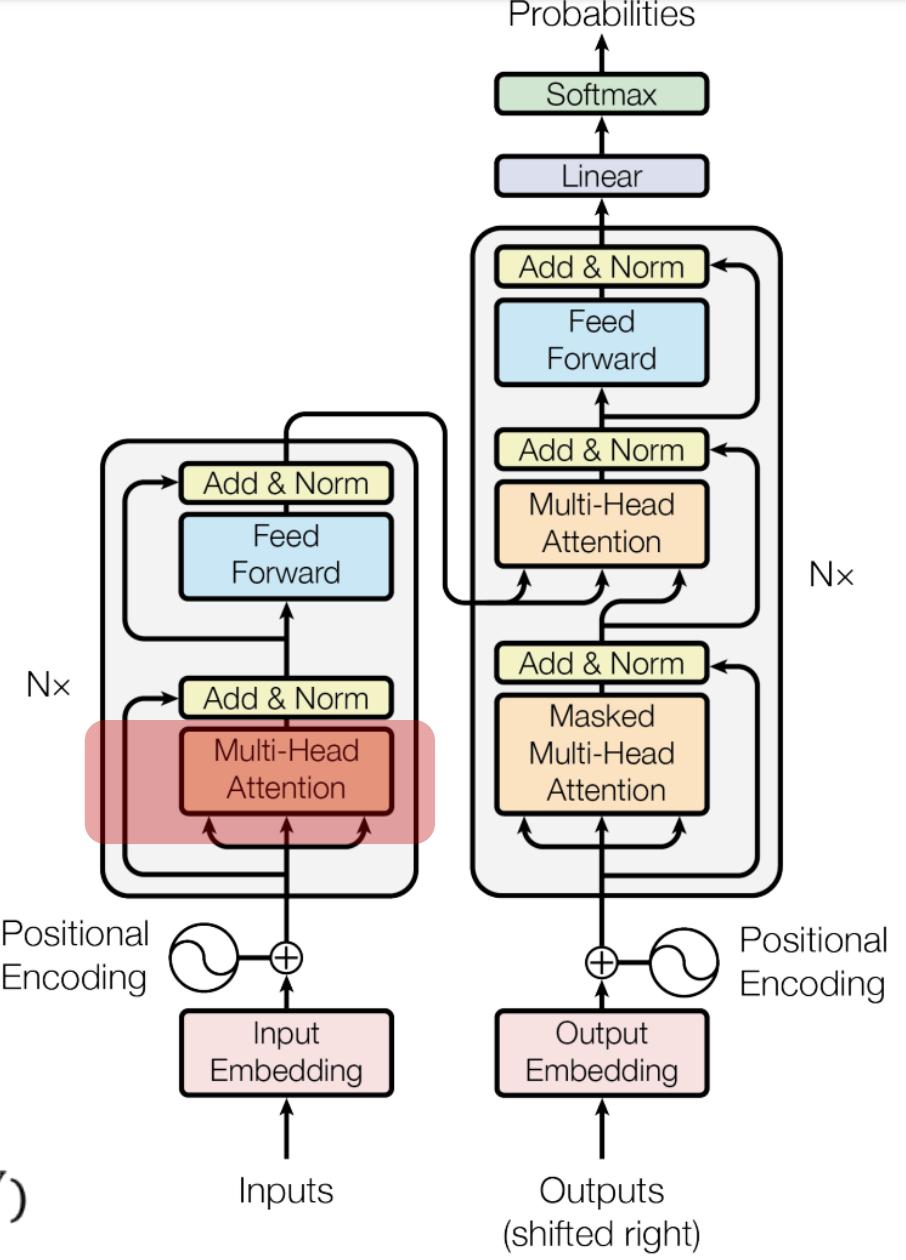
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

# Multi-head

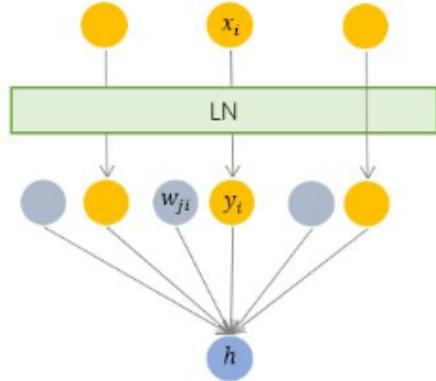


$$\text{MultiHead}(Q, K, V) = (H_1 || H_2 || \cdots || H_h)W^O$$

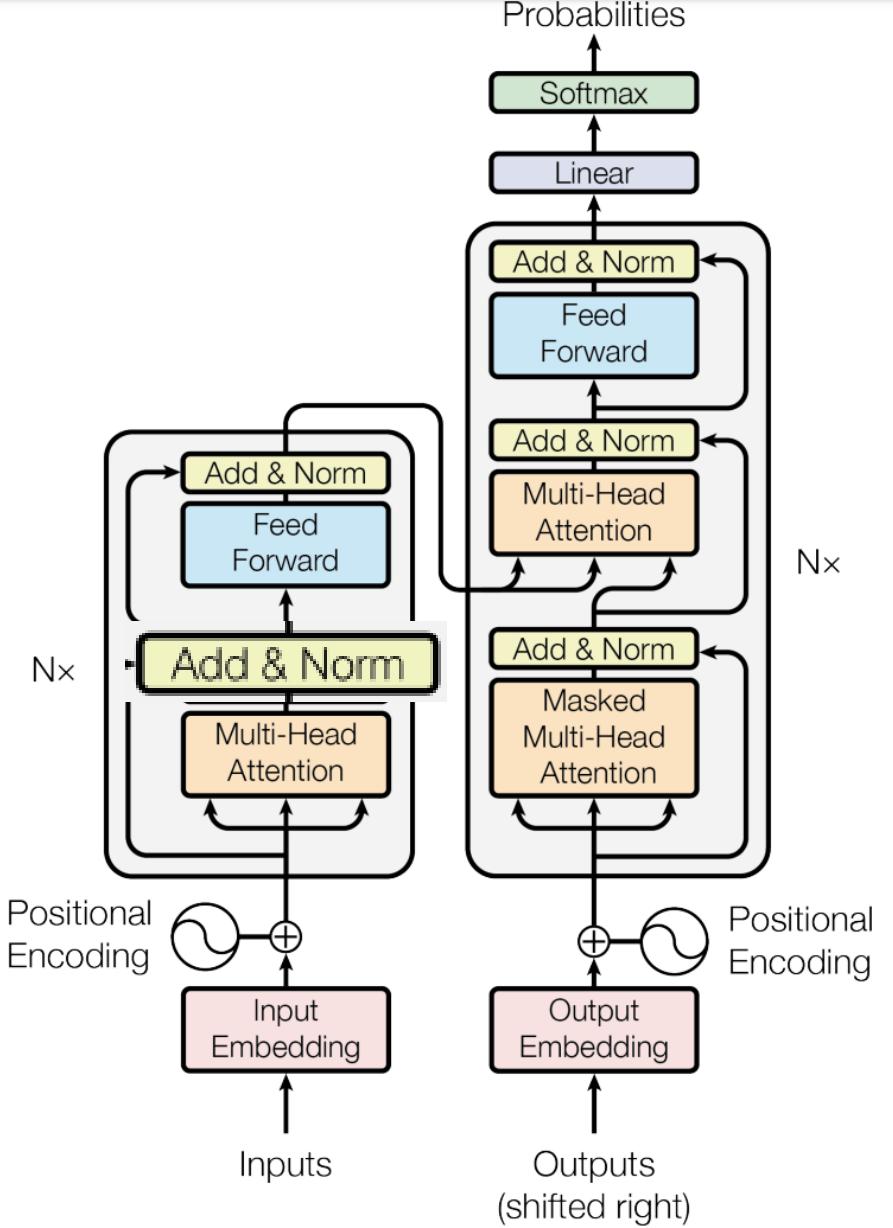
$$H_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



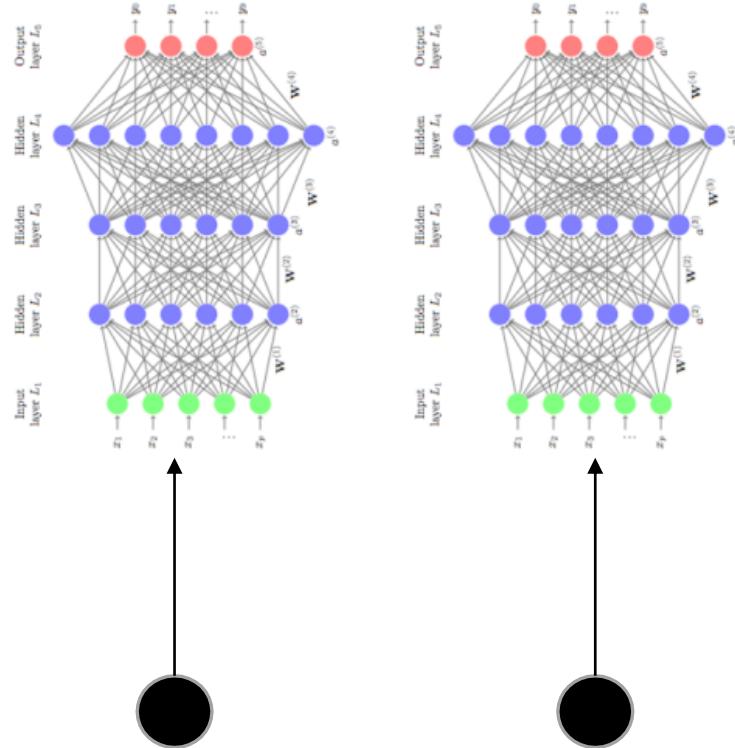
# Layer Norm



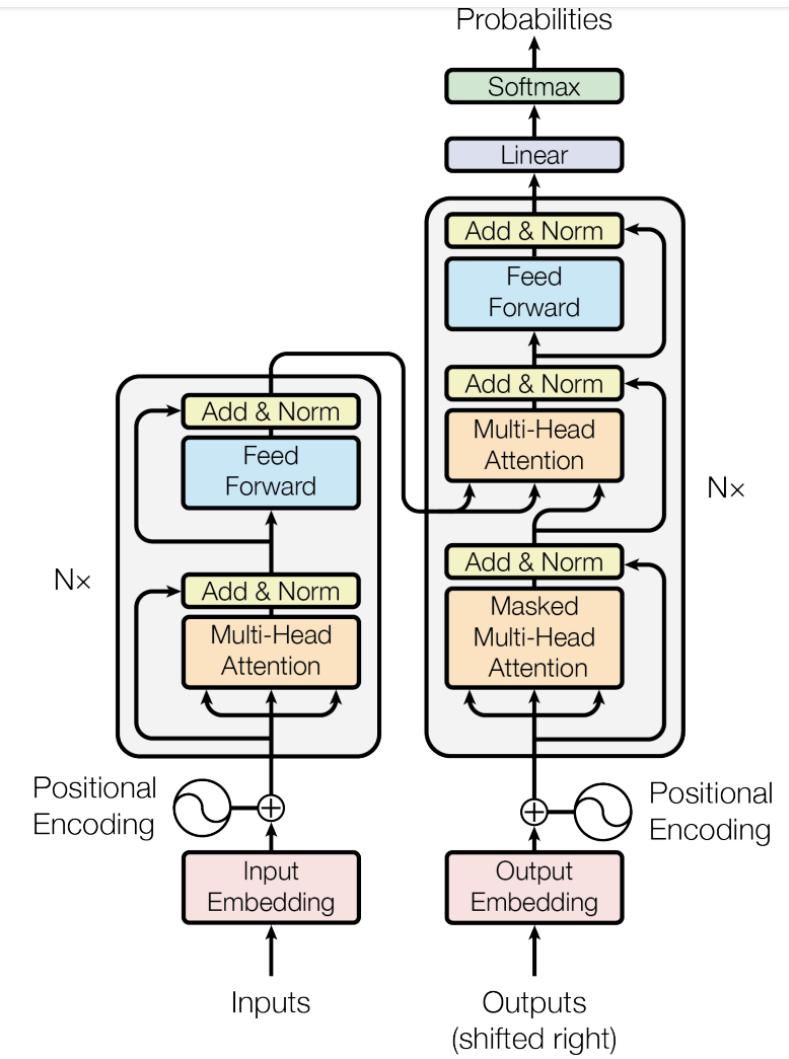
$$h = f \left( \mathbf{g} \cdot \frac{\mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} + \mathbf{b} \right)$$



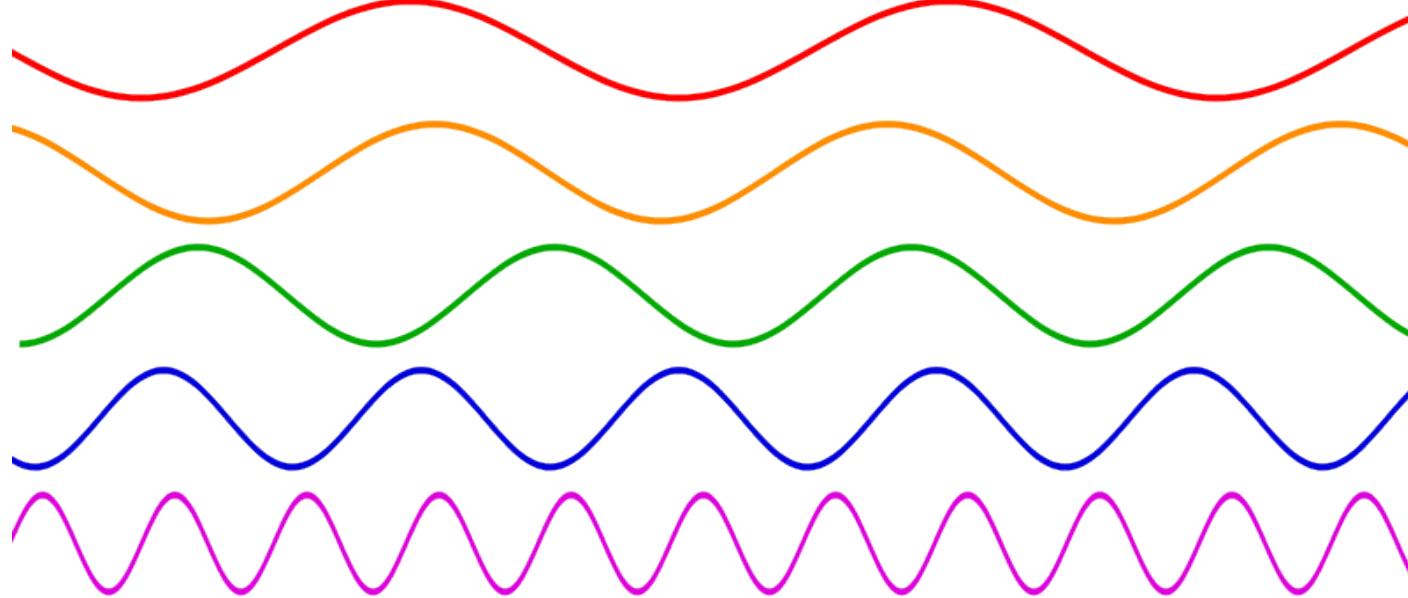
# Position-wise feedforward



Share parameters for different position



# Positional Encoding

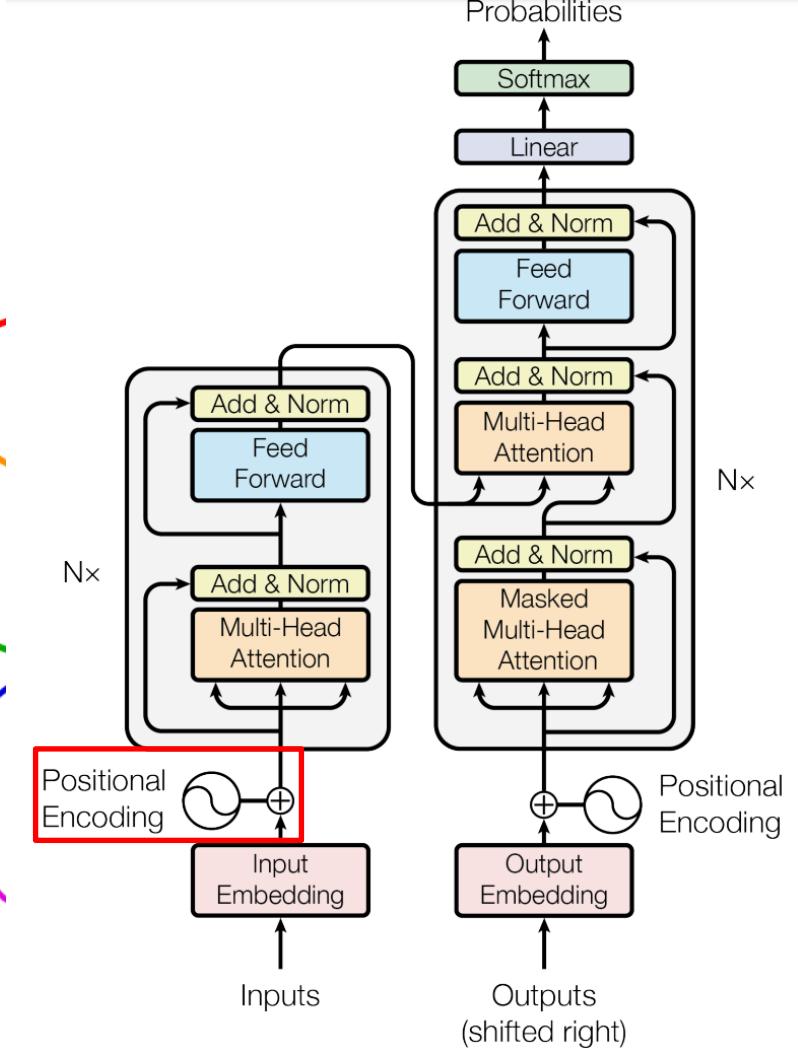


$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

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

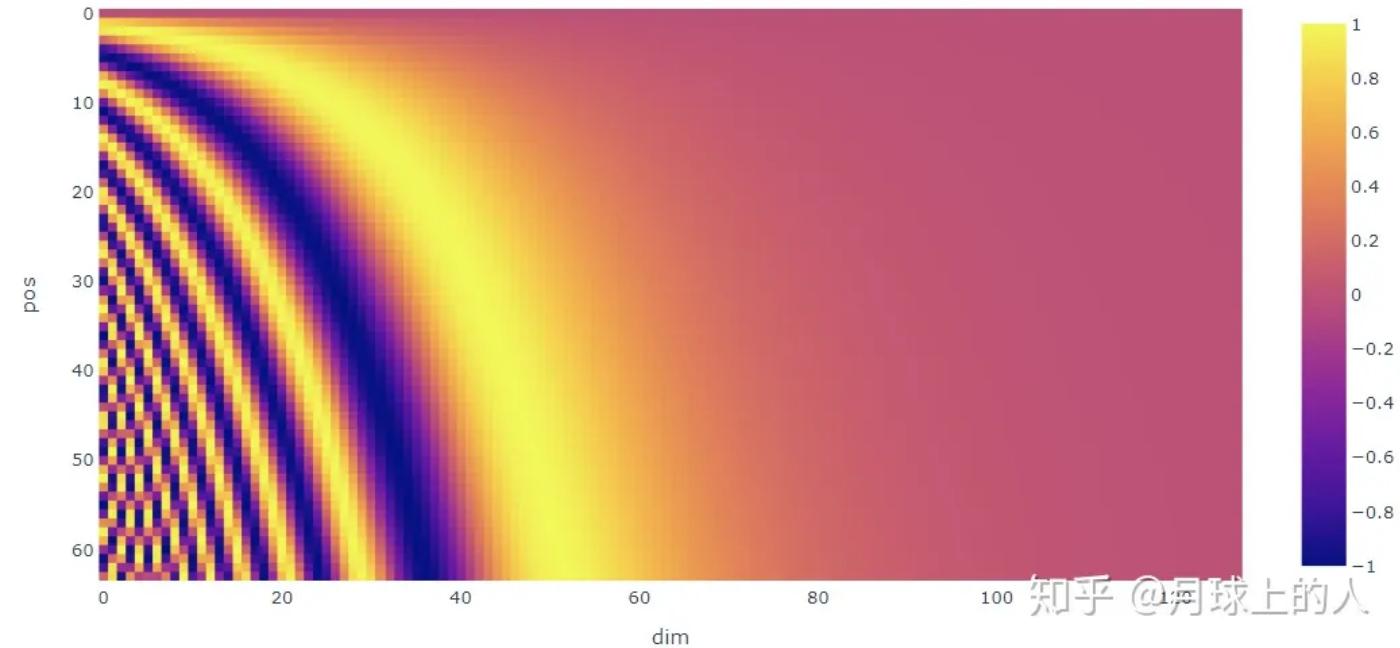
pos is the position and i is the dimension

we hypothesized it would allow the model to easily learn to attend by relative positions



# Positional Encoding

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1



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

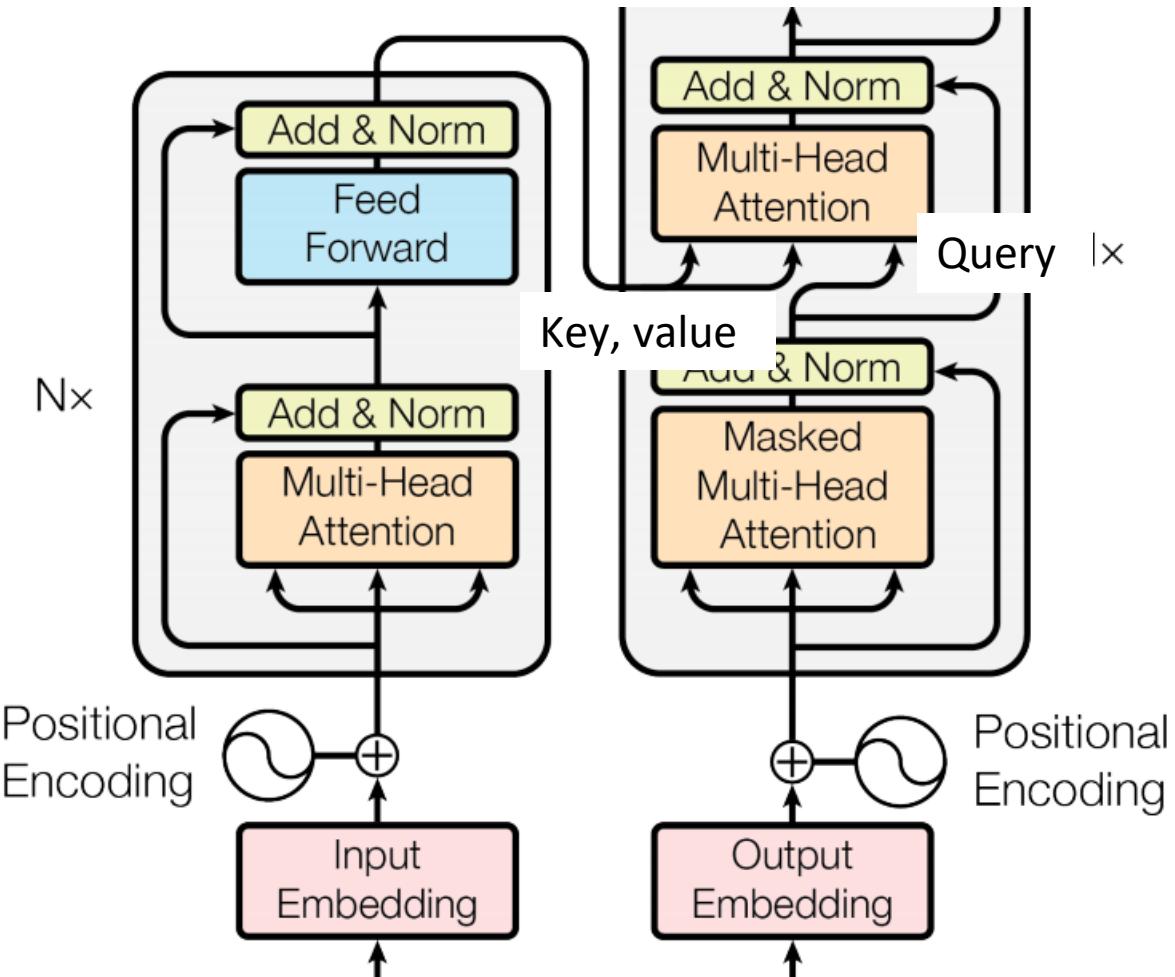
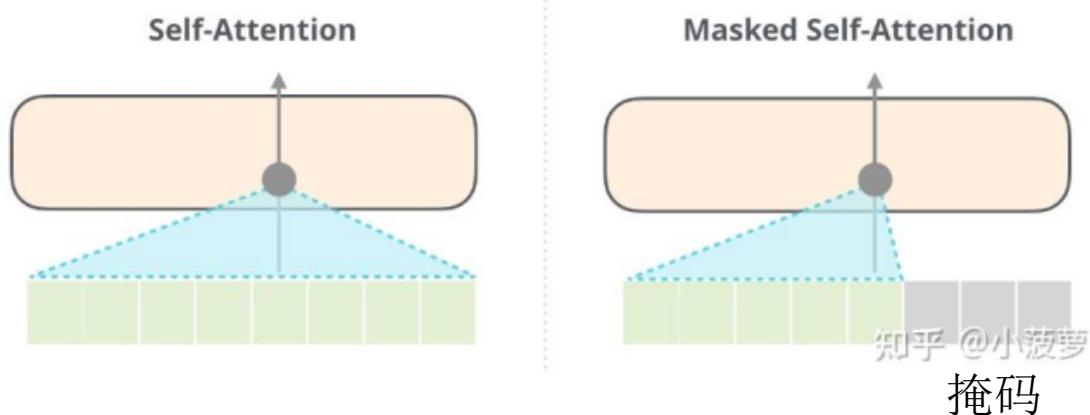
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

pos is the position and i is the dimension

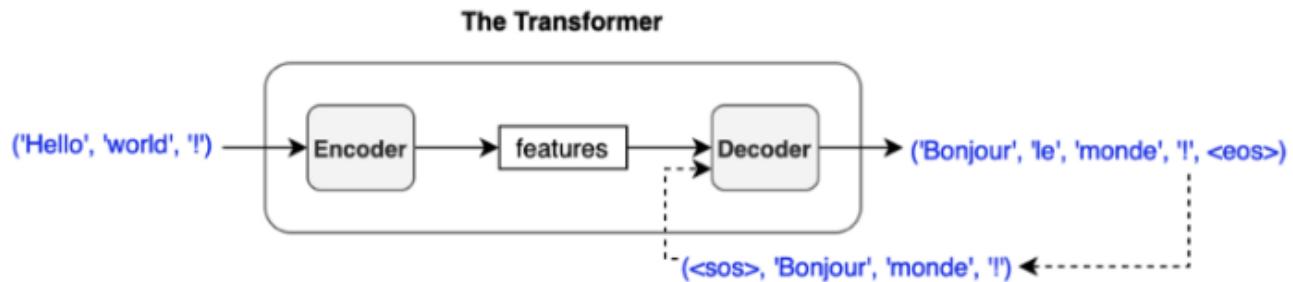
we hypothesized it would allow the model to easily learn to attend by relative positions

# Transformer Decoder

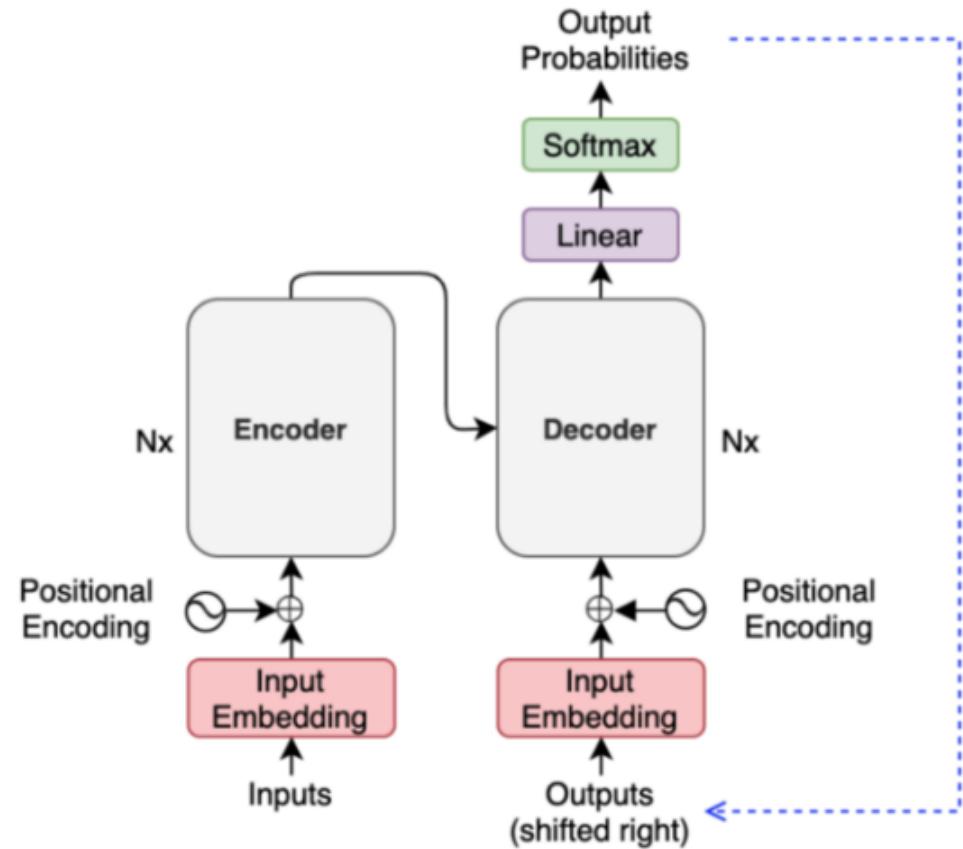
- 解码器的工作类似RNN
- 一个一个词读入，并输出预测的下一个词
- 输入来源：上一时刻的解码器输出



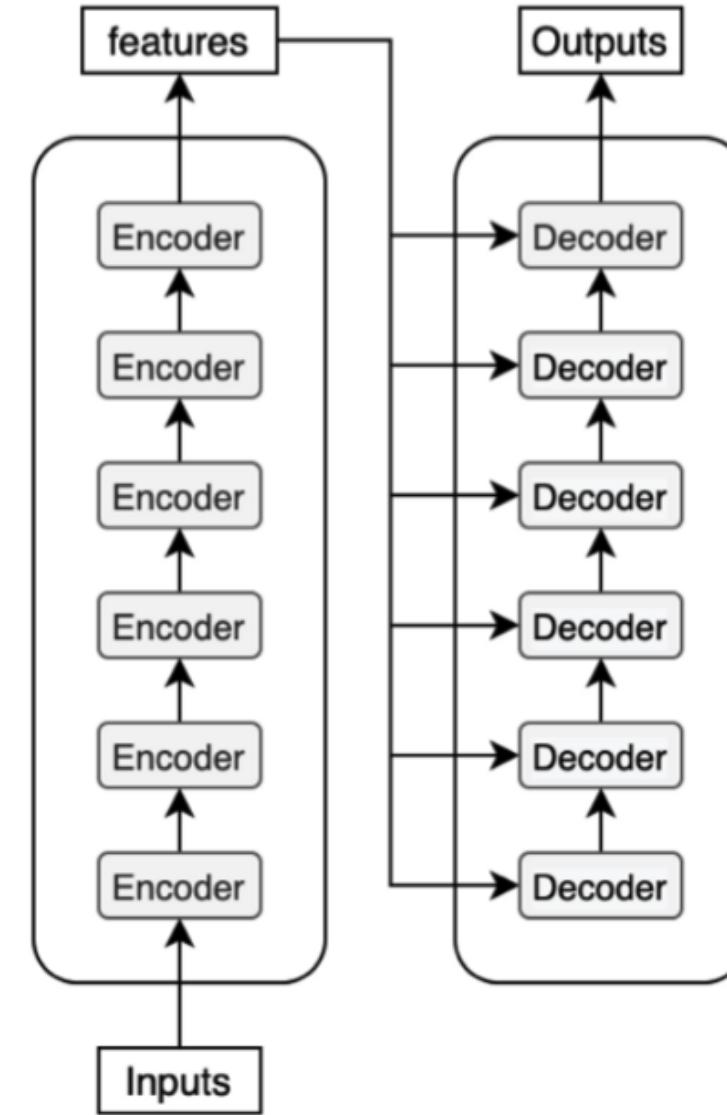
# Transformer Decoder



```
(1, 0, 0, 0, 0, ..., 0) => (<SOS>)
(1, 1, 0, 0, 0, ..., 0) => (<SOS>, 'Bonjour')
(1, 1, 1, 0, 0, ..., 0) => (<SOS>, 'Bonjour', 'le')
(1, 1, 1, 1, 0, ..., 0) => (<SOS>, 'Bonjour', 'le', 'monde')
(1, 1, 1, 1, 1, ..., 0) => (<SOS>, 'Bonjour', 'le', 'monde',
'!')
```



# Whole Picture



# Source Code

```
# 实现Transformer模型的PyTorch类
class Transformer(nn.Module):
    def __init__(self, encoder, decoder, src_embed, tgt_embed, generator):
        super(Transformer, self).__init__()
        self.encoder = encoder # 编码器
        self.decoder = decoder # 解码器
        self.src_embed = src_embed # 源语言的embedding层
        self.tgt_embed = tgt_embed # 目标语言的embedding层
        self.generator = generator # 生成最终输出结果的线性层

    def encode(self, src, src_mask):
        return self.encoder(self.src_embed(src), src_mask) # 对源语言序列进行编码

    def decode(self, memory, src_mask, tgt, tgt_mask):
        return self.decoder(self.tgt_embed(tgt), memory, src_mask, tgt_mask) # 对目标语言序列进行解码

    def forward(self, src, tgt, src_mask, tgt_mask):
        return self.decode(self.encode(src, src_mask), src_mask, tgt, tgt_mask) # 模型的前向传播

class Generator(nn.Module):
    def __init__(self, d_model, vocab):
        super(Generator, self).__init__()
        # 定义一个全连接层将d_model维的向量映射到vocab维
        self.proj = nn.Linear(d_model, vocab)

    def forward(self, x):
        # 经过全连接层后，使用log_softmax将结果转换为概率分布
        return F.log_softmax(self.proj(x), dim=-1)
```

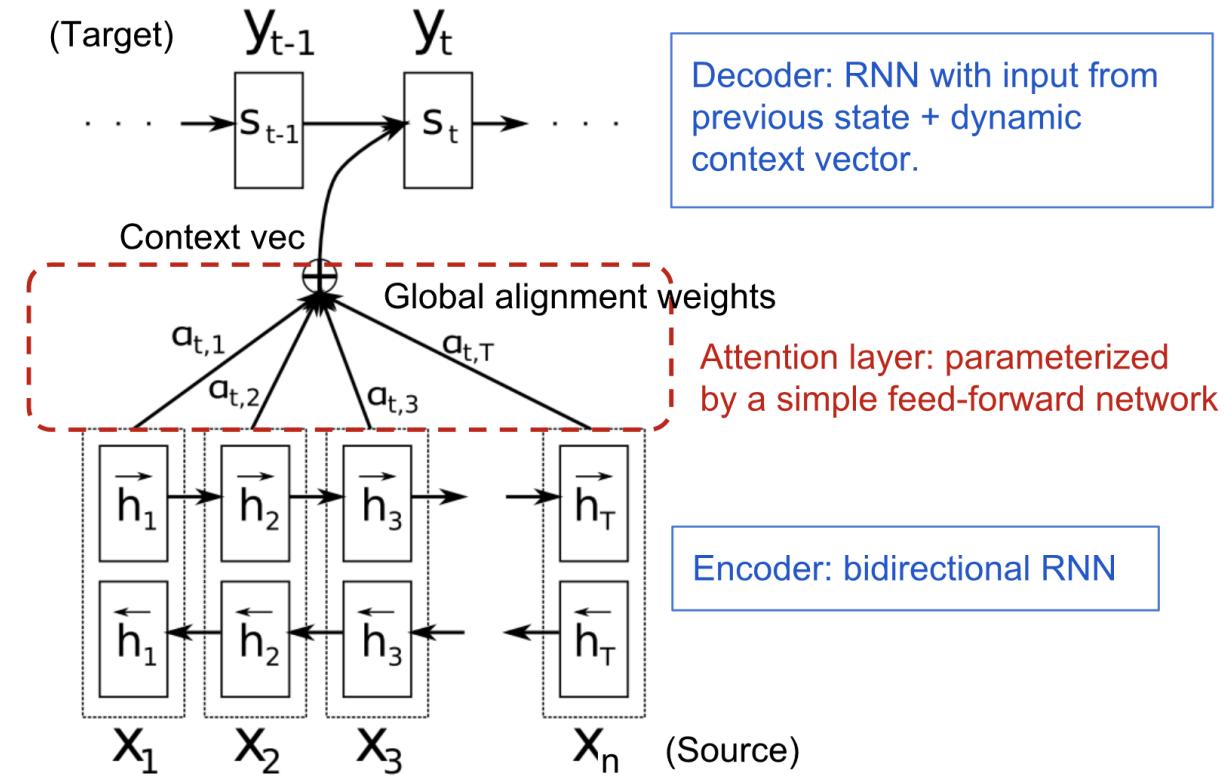
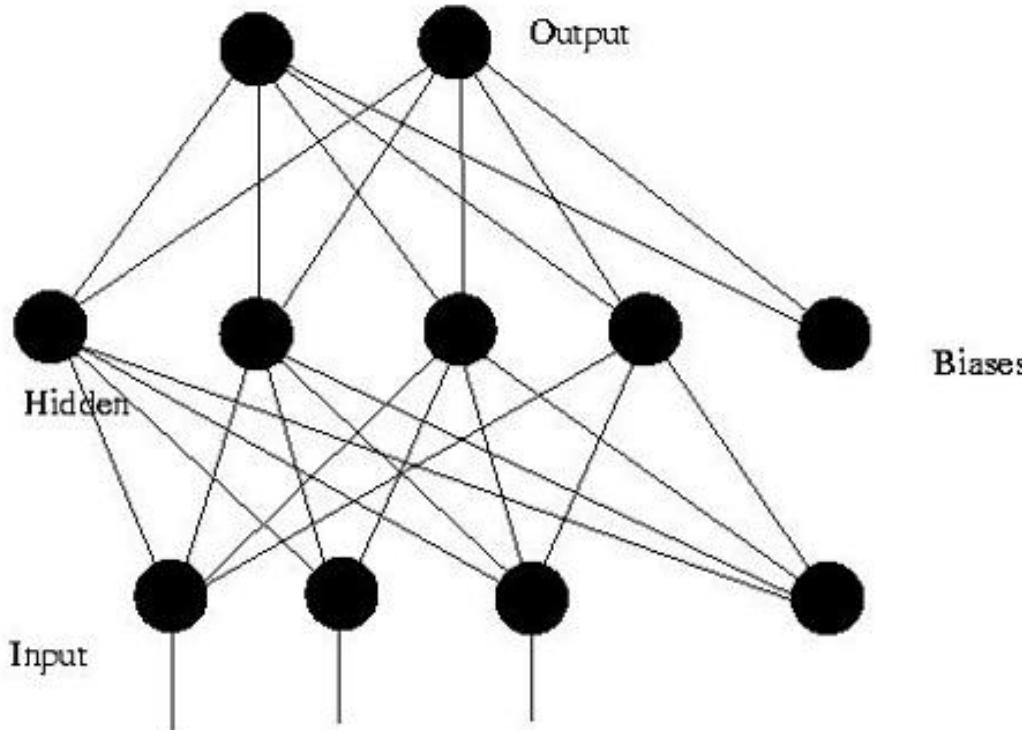


# Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

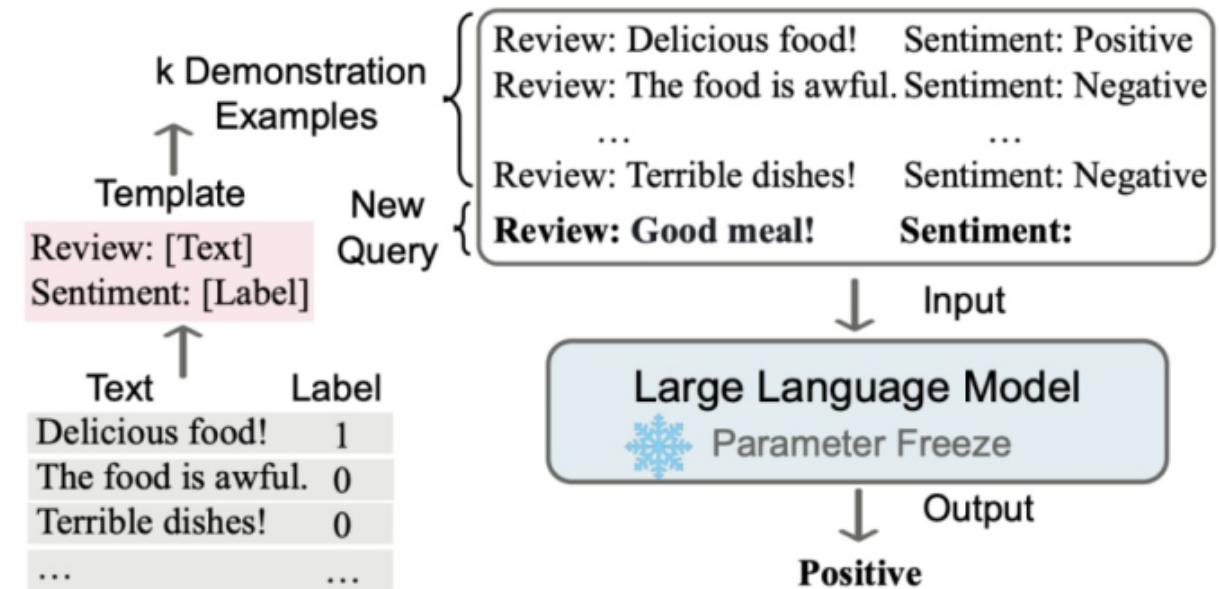
Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	<b><math>3.3 \cdot 10^{18}</math></b>	
Transformer (big)	<b>28.4</b>	<b>41.0</b>	$2.3 \cdot 10^{19}$	

# 注意力是什么？ - 前馈过程能动态调整的权重



# 揭秘In-context learning之谜

In Context Learning (ICL) 的关键思想是从示例中学习



# 揭秘In-context learning之谜

$$\begin{aligned}
 \mathcal{F}_{\text{ICL}}(\mathbf{q}) &= \text{Attn}(V, K, \mathbf{q}) \\
 &\approx W_V [X'; X] (W_K [X'; X])^T \mathbf{q} \\
 &= W_V X (W_K X)^T \mathbf{q} + W_V X' (W_K X')^T \mathbf{q} \\
 &= \tilde{\mathcal{F}}_{\text{ICL}}(\mathbf{q}).
 \end{aligned}$$

$$\begin{aligned}
 \tilde{\mathcal{F}}_{\text{ICL}}(\mathbf{q}) &= W_{\text{ZSL}} \mathbf{q} + W_V X' (W_K X')^T \mathbf{q} \\
 &= W_{\text{ZSL}} \mathbf{q} + \text{LinearAttn}(W_V X', W_K X', \mathbf{q}) \\
 &= W_{\text{ZSL}} \mathbf{q} + \sum_i W_V \mathbf{x}'_i ((W_K \mathbf{x}'_i)^T \mathbf{q}) \\
 &= W_{\text{ZSL}} \mathbf{q} + \sum_i (W_V \mathbf{x}'_i \otimes (W_K \mathbf{x}'_i)^T) \mathbf{q} \\
 &= W_{\text{ZSL}} \mathbf{q} + \Delta W_{\text{ICL}} \mathbf{q} \\
 &= (W_{\text{ZSL}} + \Delta W_{\text{ICL}}) \mathbf{q}.
 \end{aligned}$$

Input      label

## Why Can GPT Learn In-Context? Language Models Secretly Perform Gradient Descent as Meta-Optimizers

Damai Dai<sup>†\*</sup>, Yutao Sun<sup>||\*</sup>, Li Dong<sup>‡</sup>, Yaru Hao<sup>‡</sup>, Zhifang Sui<sup>†</sup>, Furu Wei<sup>‡</sup>

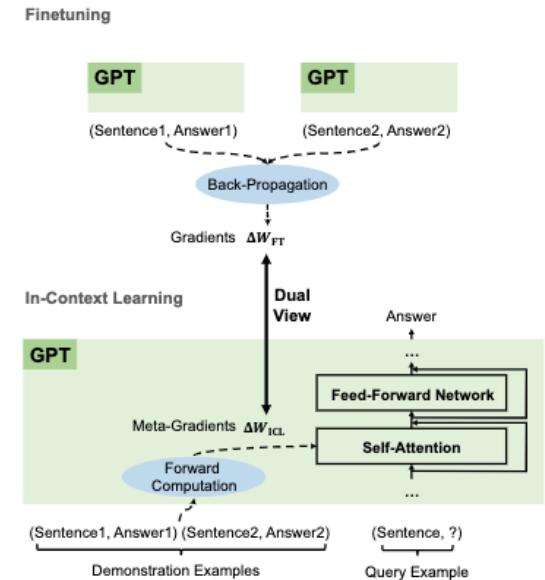
<sup>†</sup> Peking University    <sup>||</sup> Tsinghua University

<sup>‡</sup> Microsoft Research

<https://github.com/microsoft/LMOpns>

### Abstract

Large pretrained language models have shown surprising In-Context Learning (ICL) ability. With a few demonstration input-label pairs, they can predict the label for an unseen input without additional parameter updates. Despite the great success in performance, the working mechanism of ICL still remains an open problem. In order to better understand how ICL works, this paper explains language models as meta-optimizers and understands ICL as a kind of implicit finetuning. Theoretically, we figure out that the Transformer attention has a dual form of gradient descent based optimization. On top of it, we understand ICL as follows: GPT first produces meta-gradients according to the demonstration examples, and then these meta-gradients are applied to the original GPT to build an ICL

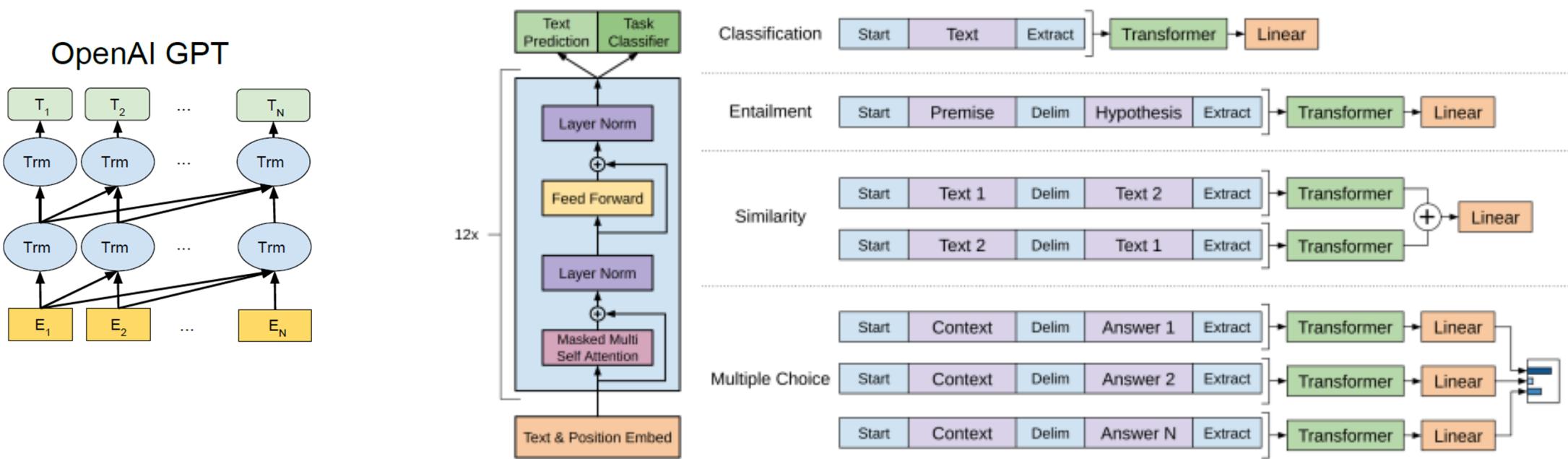


# Outline

- Knowing LLM
- Transformer
- GPT

# What is GPT

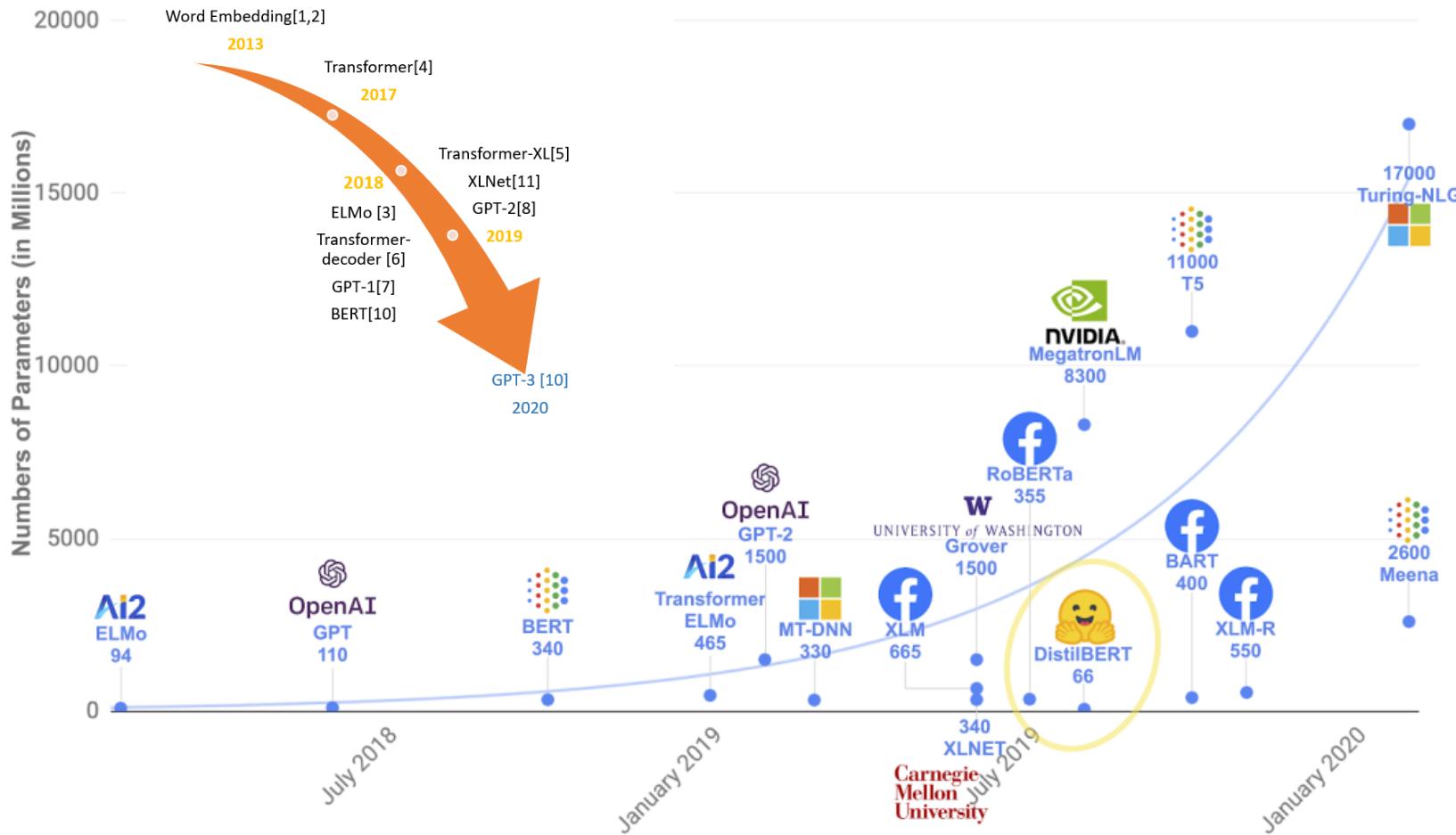
## Generative Pre-trained Transformer



[https://cdn.openai.com/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf)



# History of Pretraining Language Models



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

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

Google AI Language

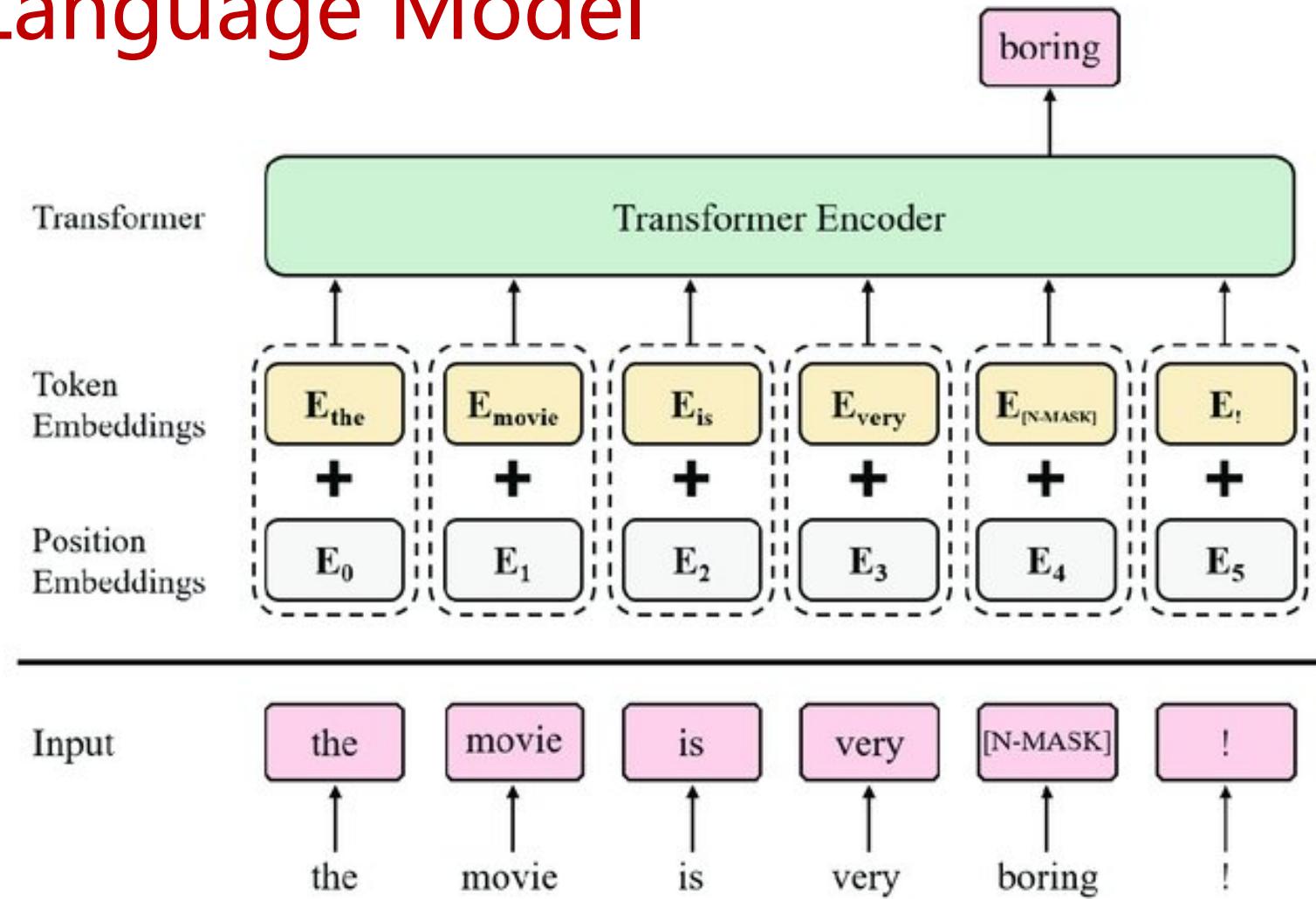
{jacobdevlin, mingweichang, kentonl, kristout}@google.com

### Abstract

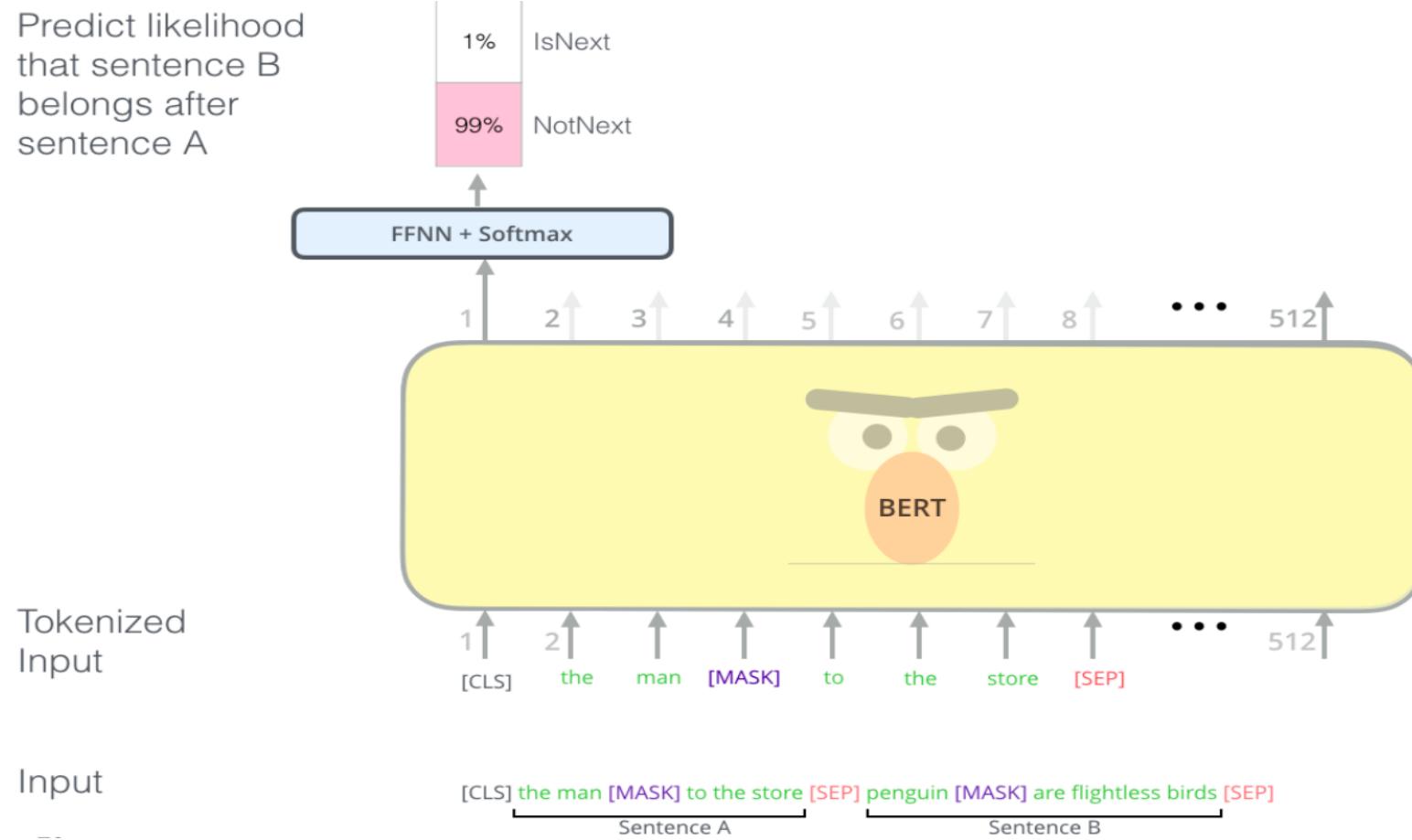
We introduce a new language representation model called **BERT**, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), takes a minimal

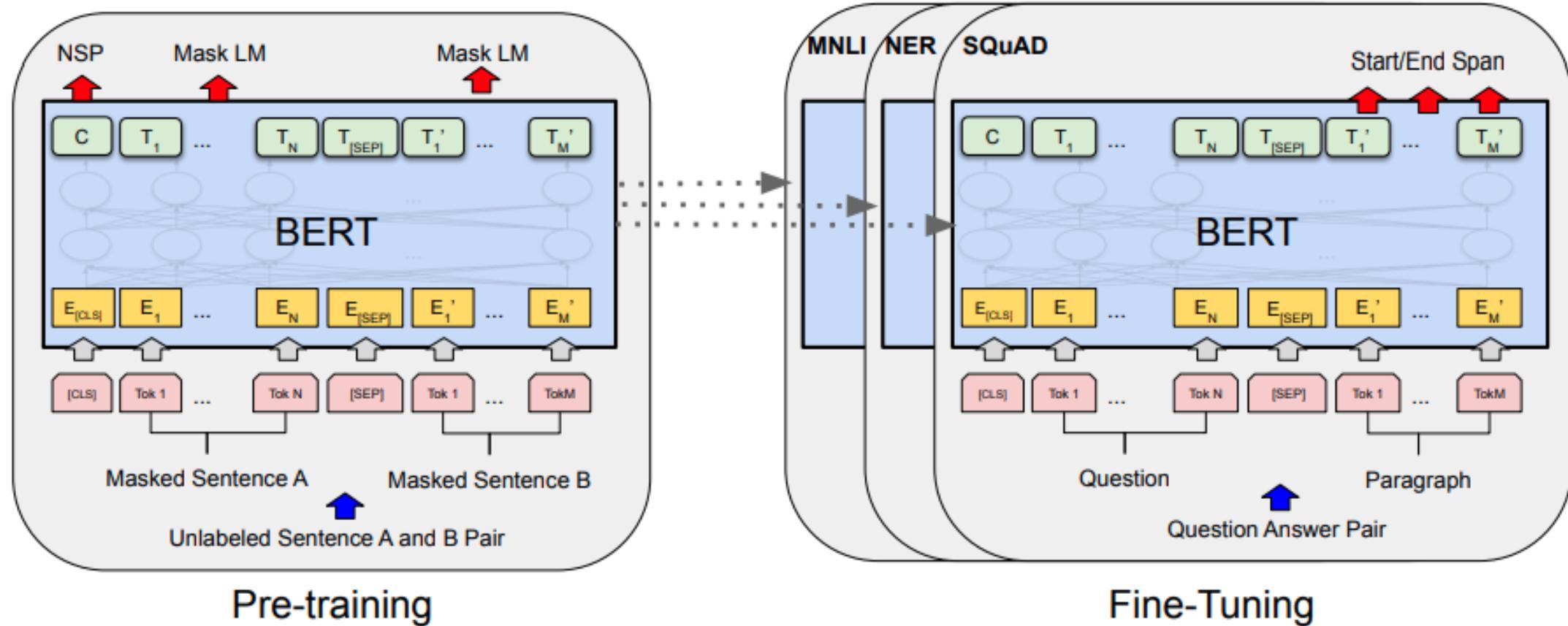
# Masked Language Model



# Predict Next Sentence



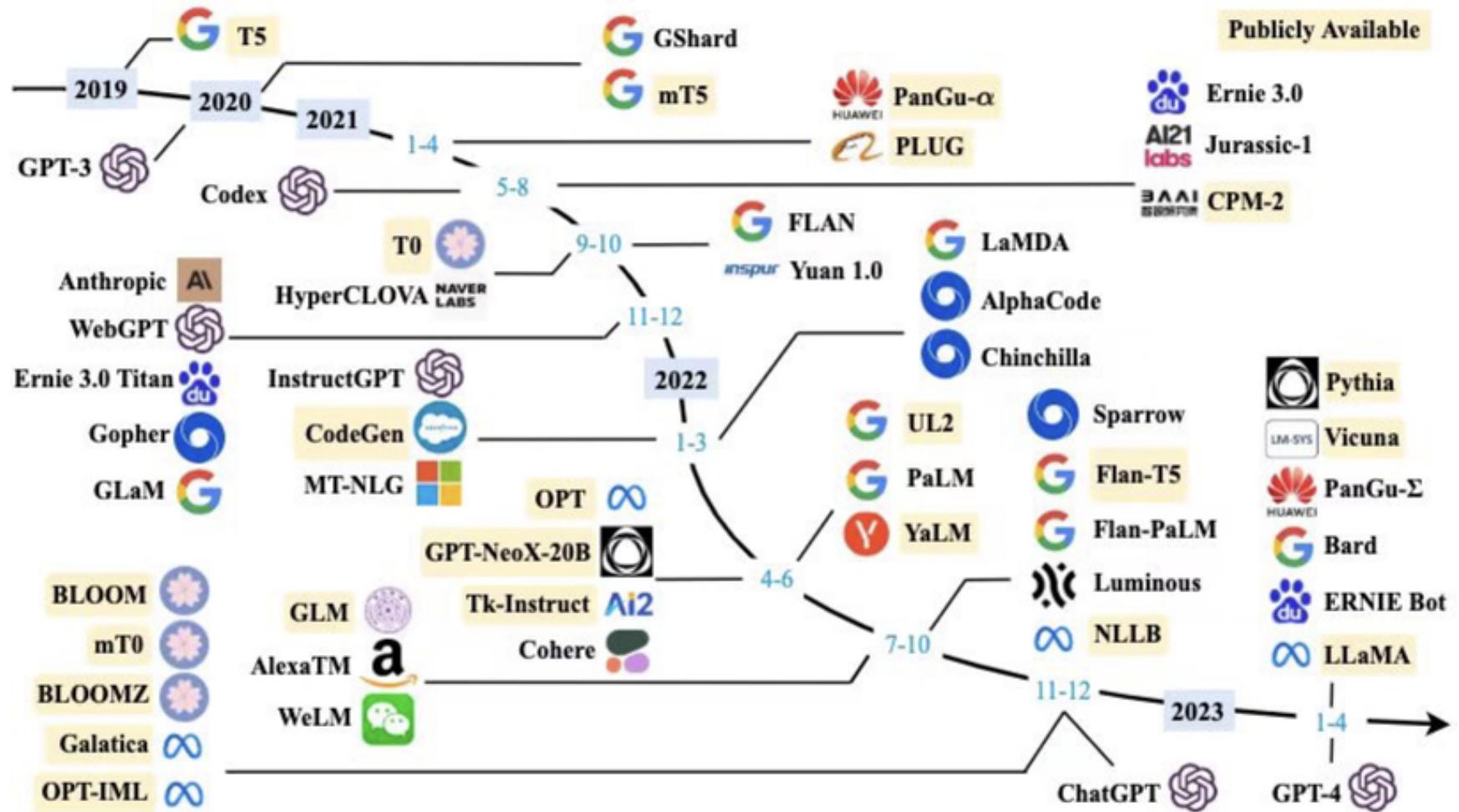
# Two Stages



# Results of BERT

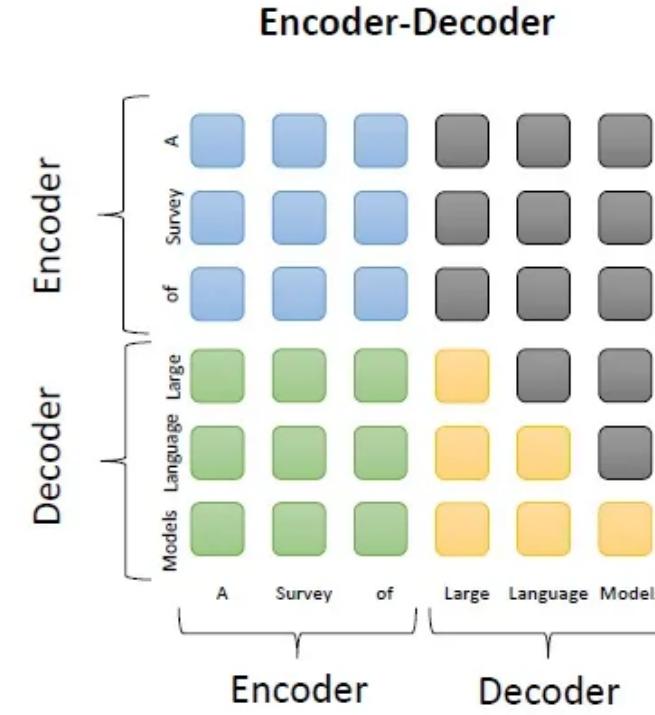
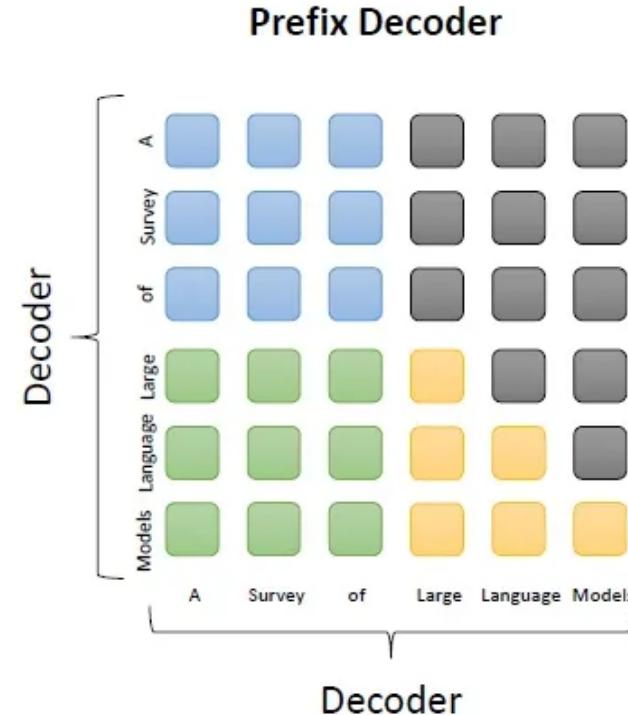
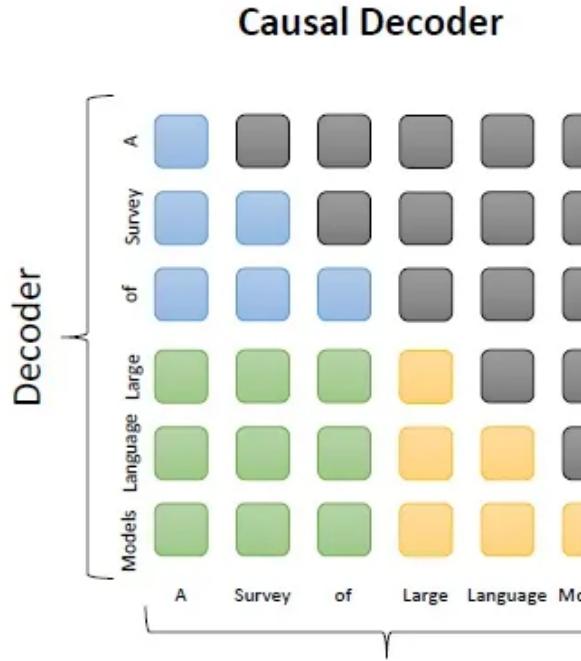
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT <sub>BASE</sub>	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT <sub>LARGE</sub>	<b>86.7/85.9</b>	<b>72.1</b>	<b>92.7</b>	<b>94.9</b>	<b>60.5</b>	<b>86.5</b>	<b>89.3</b>	<b>70.1</b>	<b>82.1</b>

# History of Large Models



# Attention patterns in mainstream architectures

输入双向注意力，输出单向注意力，  
输入输出共享参数



输入双向注意力，  
输出单向注意力  
输入输出不共享参  
数

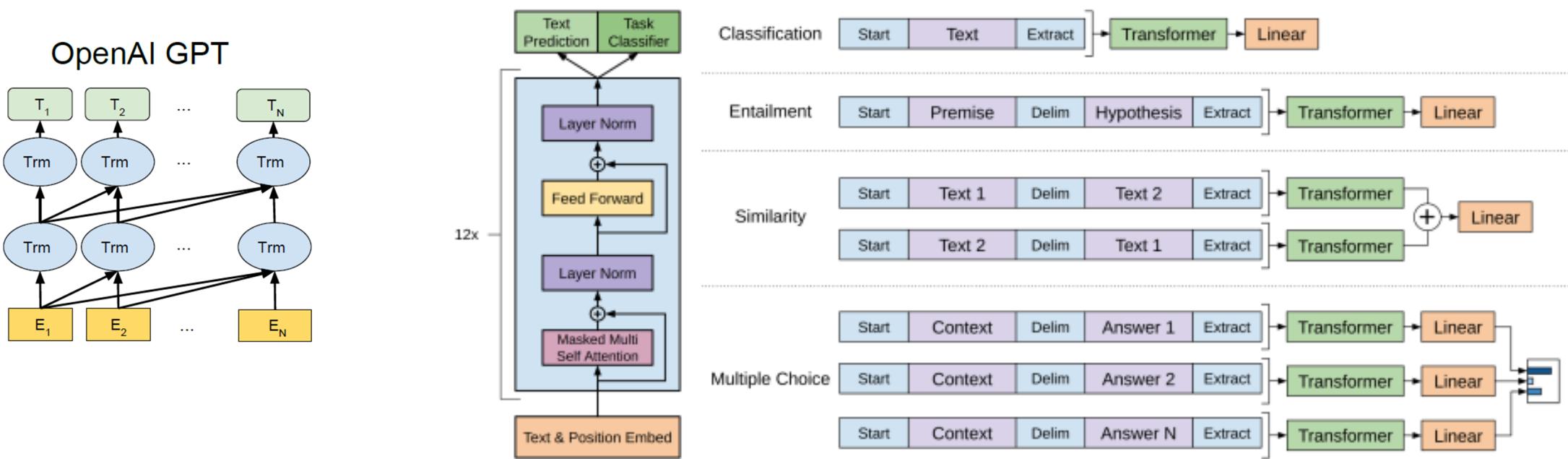
**Causal Decoder:** 从左到右的单向注意力。

Fig. 7: A comparison of the attention patterns in three mainstream architectures. Colored rounded rectangles indicate the attention between prefix tokens, attention between target tokens, and masked attention respectively.

模型架构	代表LLM	注意力机制	是否属于Decoder-Only
Causal Decoder	GPT3/ChatGPT	纯单向	YES
Encoder-Decoder	Flan-T5	输入双向，输出单向	NO
Prefix Decoder	GLM-130B/ChatGLM-6B	输入双向，输出单向	YES

# What is GPT

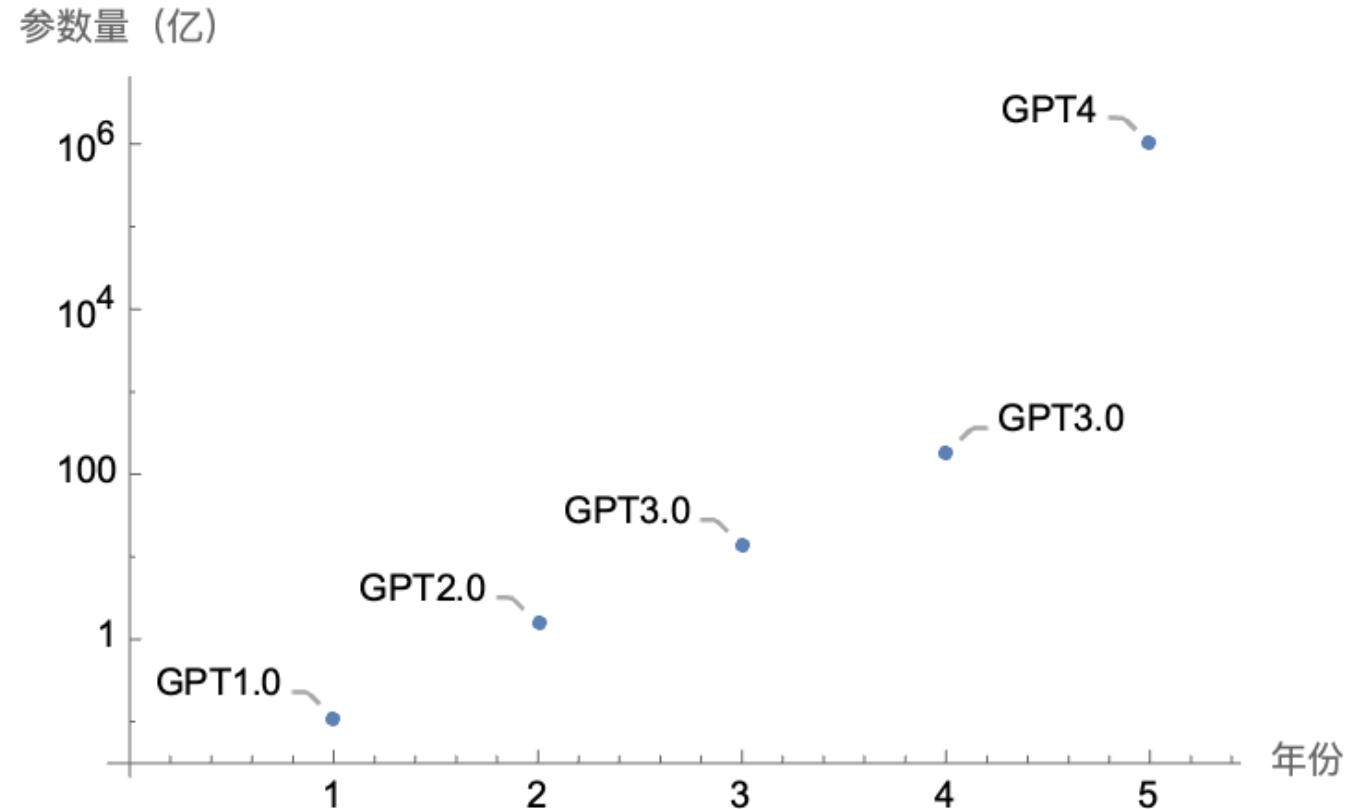
## Generative Pre-trained Transformer



[https://cdn.openai.com/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf)



# # of Parameters



# Evolution of GPT

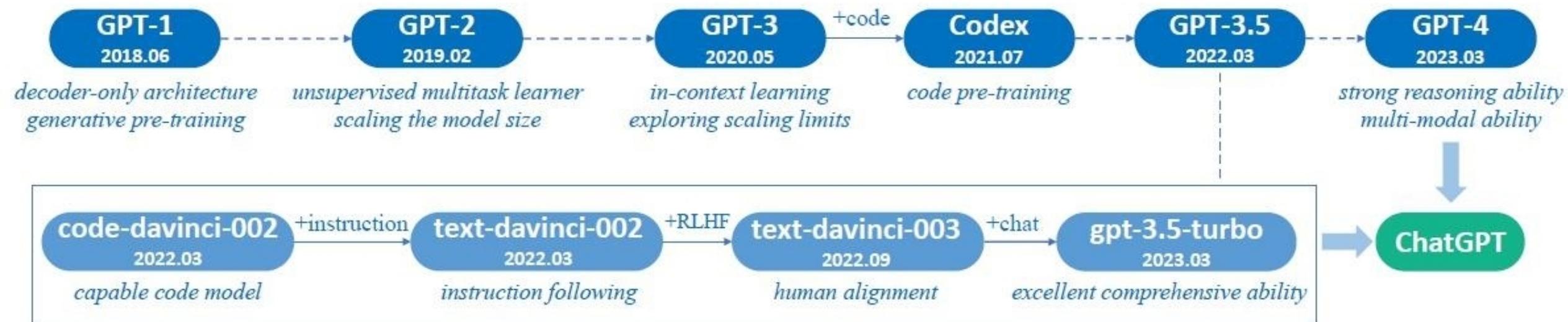
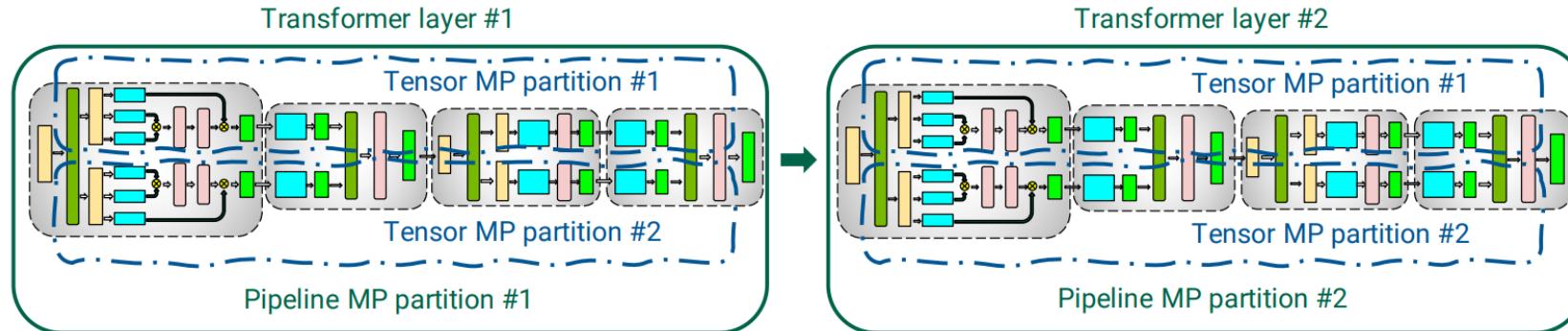


Fig. 3: A brief illustration for the technical evolution of GPT-series models. We plot this figure mainly based on the papers, blog articles and official APIs from OpenAI. Here, *solid lines* denote that there exists an explicit evidence (e.g. the official statement that a new model is developed based on a base model) on the evolution path between two models, while *dashed lines* denote a relatively weaker evolution relation.

# What is GPT



[Submitted on 16 Nov 2018 (v1), last revised 25 Jul 2019 (this version, v5)]

## GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism

[Submitted on 4 Oct 2019 (v1), last revised 13 May 2020 (this version, v3)]

## ZeRO: Memory Optimizations Toward Training Trillion Parameter Models

[Submitted on 17 Sep 2019 (v1), last revised 13 Mar 2020 (this version, v4)]

## Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism

[Submitted on 28 May 2020 (v1), last revised 22 Jul 2020 (this version, v4)]

## Language Models are Few-Shot Learners

[Submitted on 9 Apr 2021 (v1), last revised 23 Aug 2021 (this version, v5)]

## Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM

[Submitted on 5 Oct 2022]

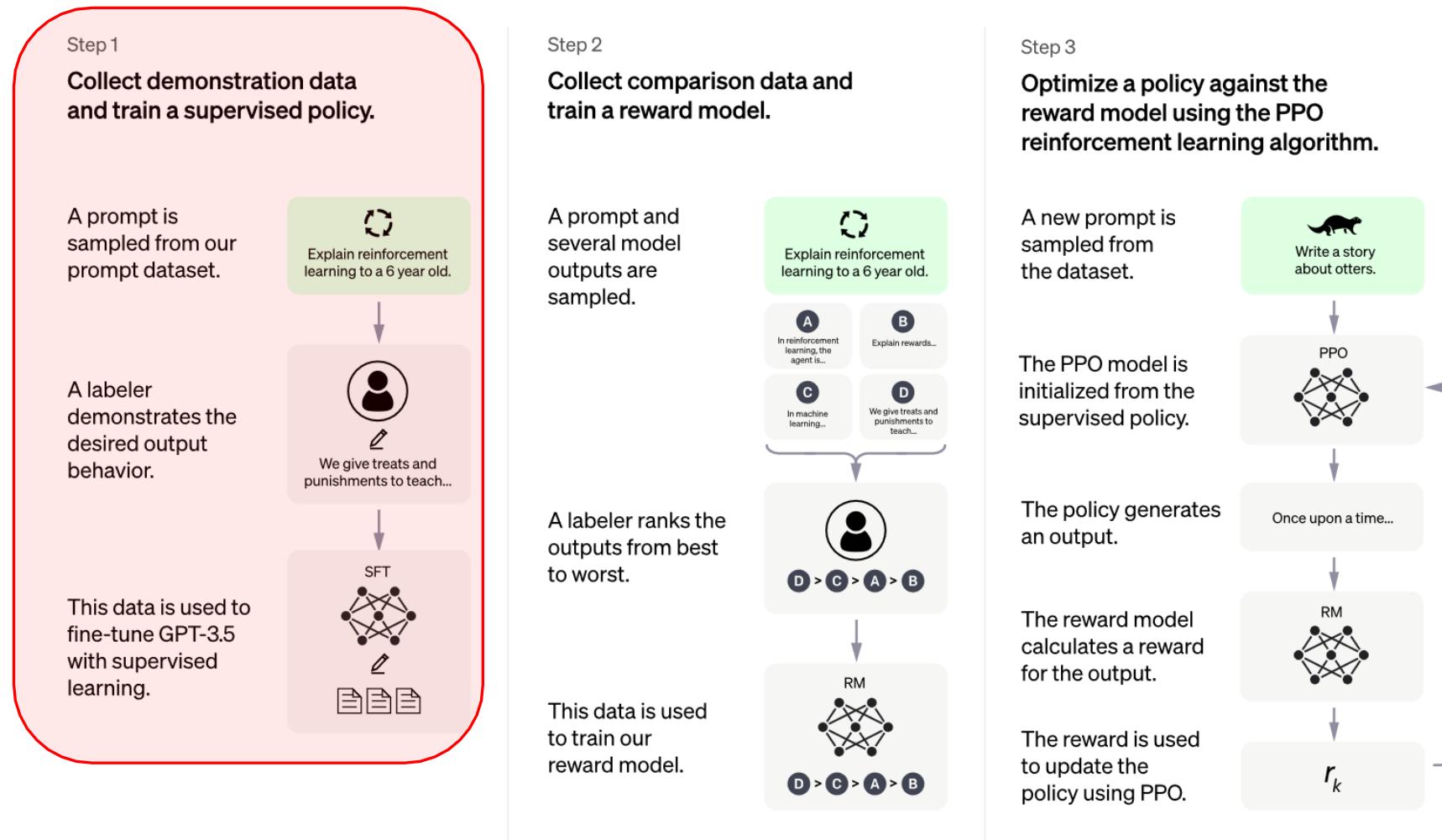
## GLM-130B: An Open Bilingual Pre-trained Model

To train the larger models without running out of memory, we use a mixture of model parallelism within each matrix multiply and model parallelism across the layers of the network.

# GPT Assistant training pipeline



# What is GPT

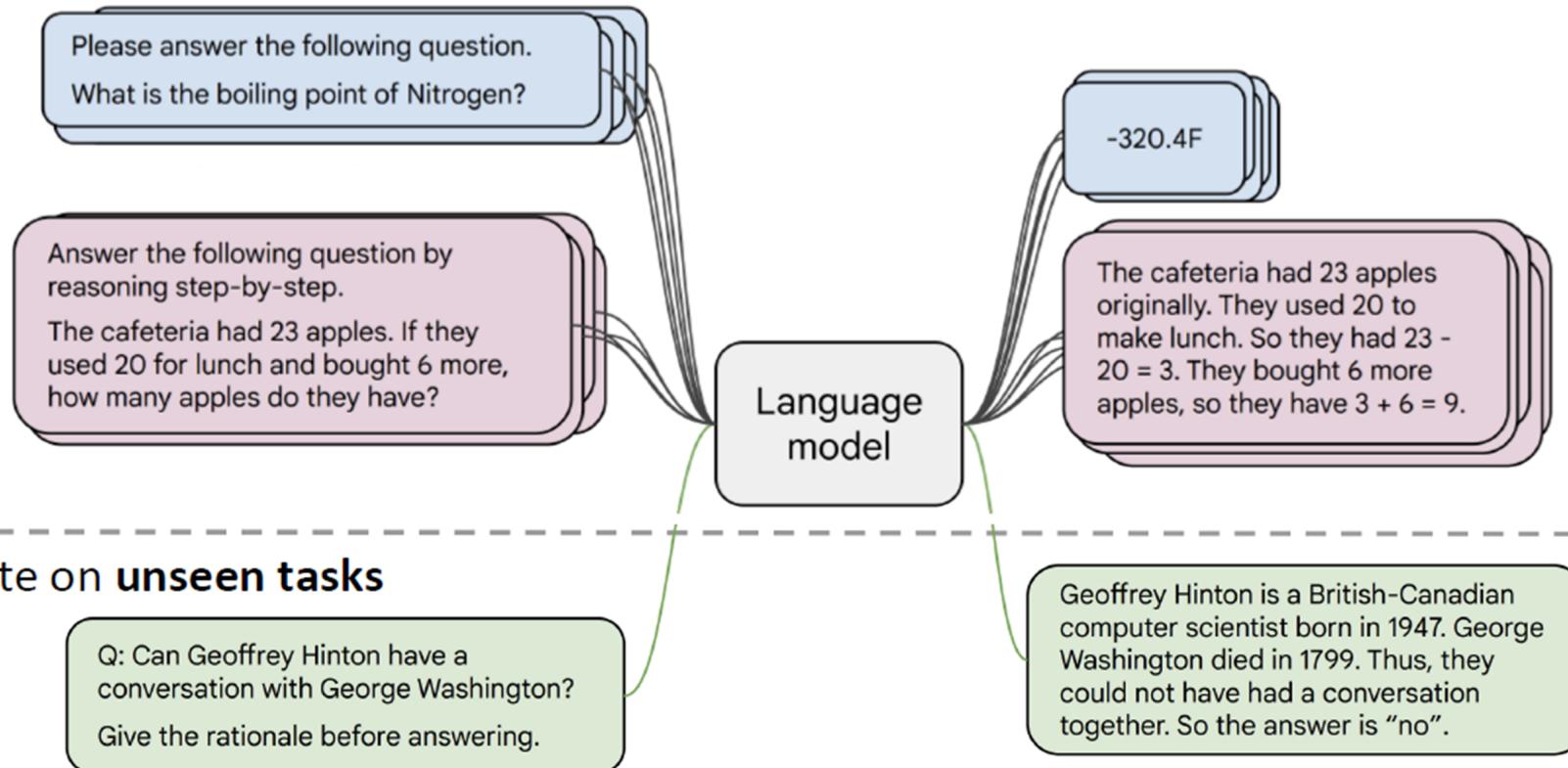


[https://cdn.openai.com/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf)

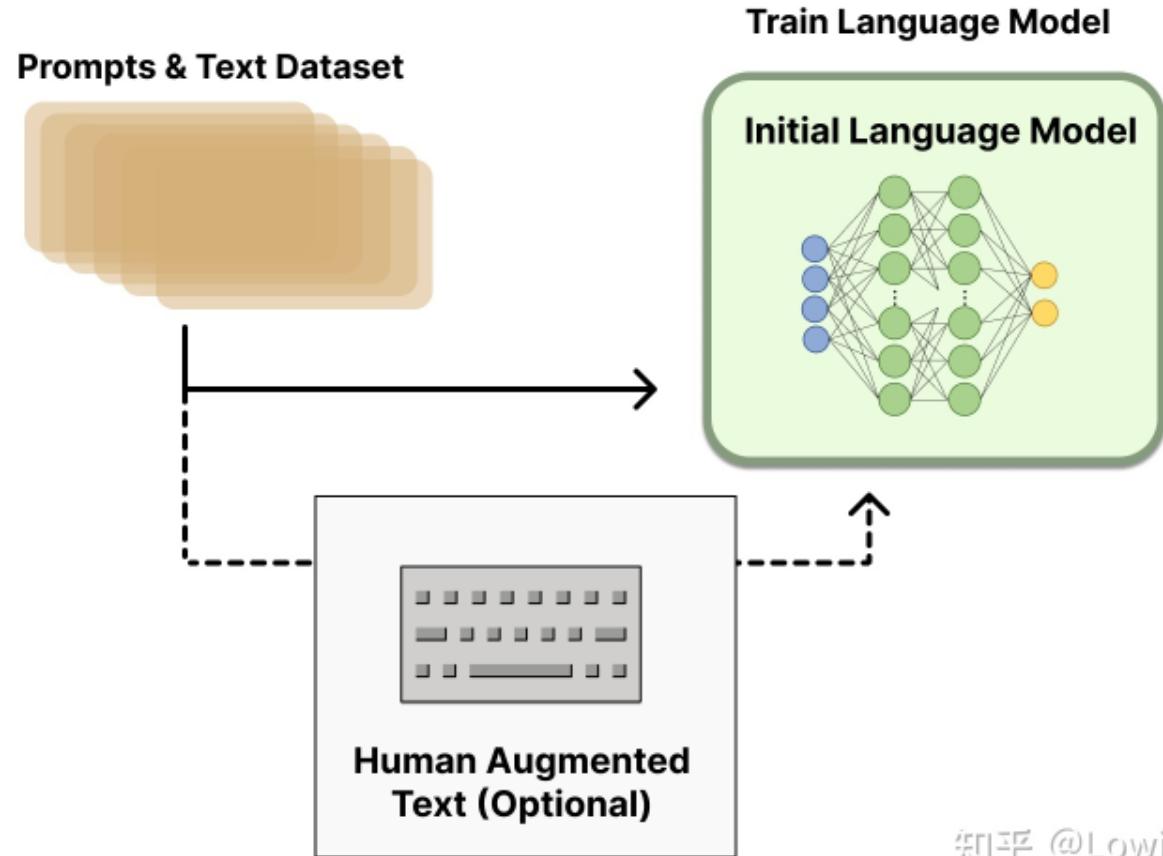


# 指令微调 (Instruct tuning)

Collect examples of (instruction, output) pairs across many tasks and finetune an LM



# 指令微调 (Instruct tuning)



知乎 @LowinLi

# What is GPT

Step 1

**Collect demonstration data and train a supervised policy.**

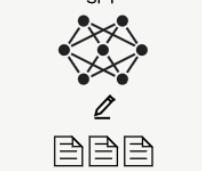
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



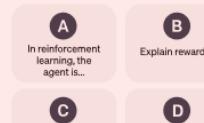
This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

**Collect comparison data and train a reward model.**

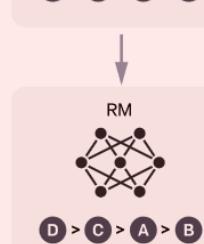
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



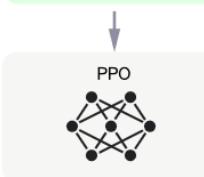
Step 3

**Optimize a policy against the reward model using the PPO reinforcement learning algorithm.**

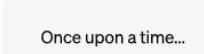
A new prompt is sampled from the dataset.



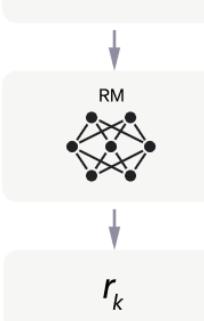
The PPO model is initialized from the supervised policy.



The policy generates an output.

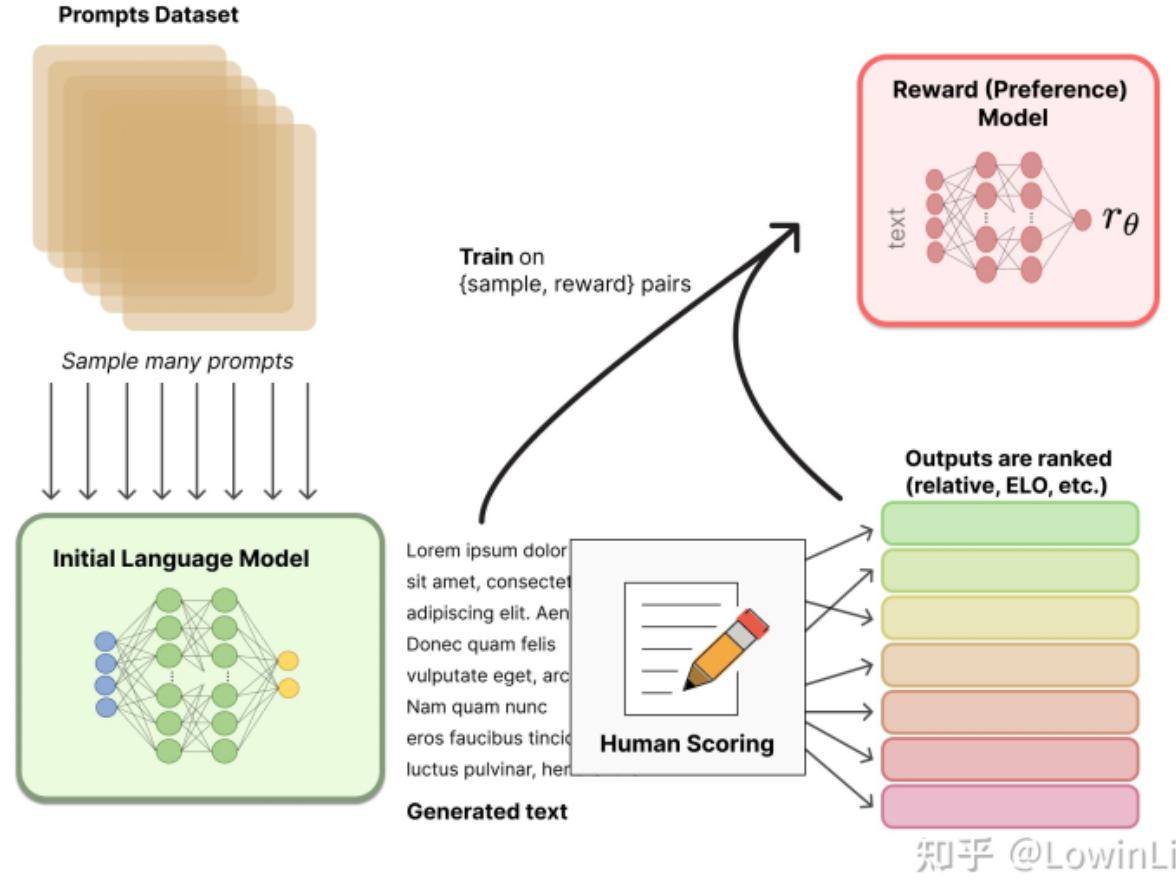


The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

# Reward Model



<https://zhuanlan.zhihu.com/p/596059221>



# What is GPT

Step 1

**Collect demonstration data and train a supervised policy.**

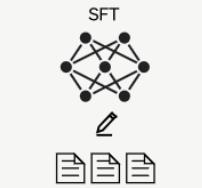
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

**Collect comparison data and train a reward model.**

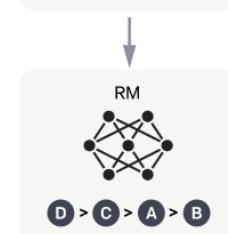
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



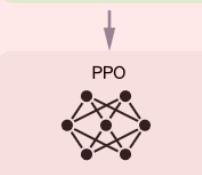
Step 3

**Optimize a policy against the reward model using the PPO reinforcement learning algorithm.**

A new prompt is sampled from the dataset.



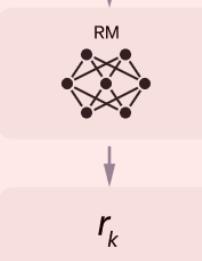
The PPO model is initialized from the supervised policy.



The policy generates an output.



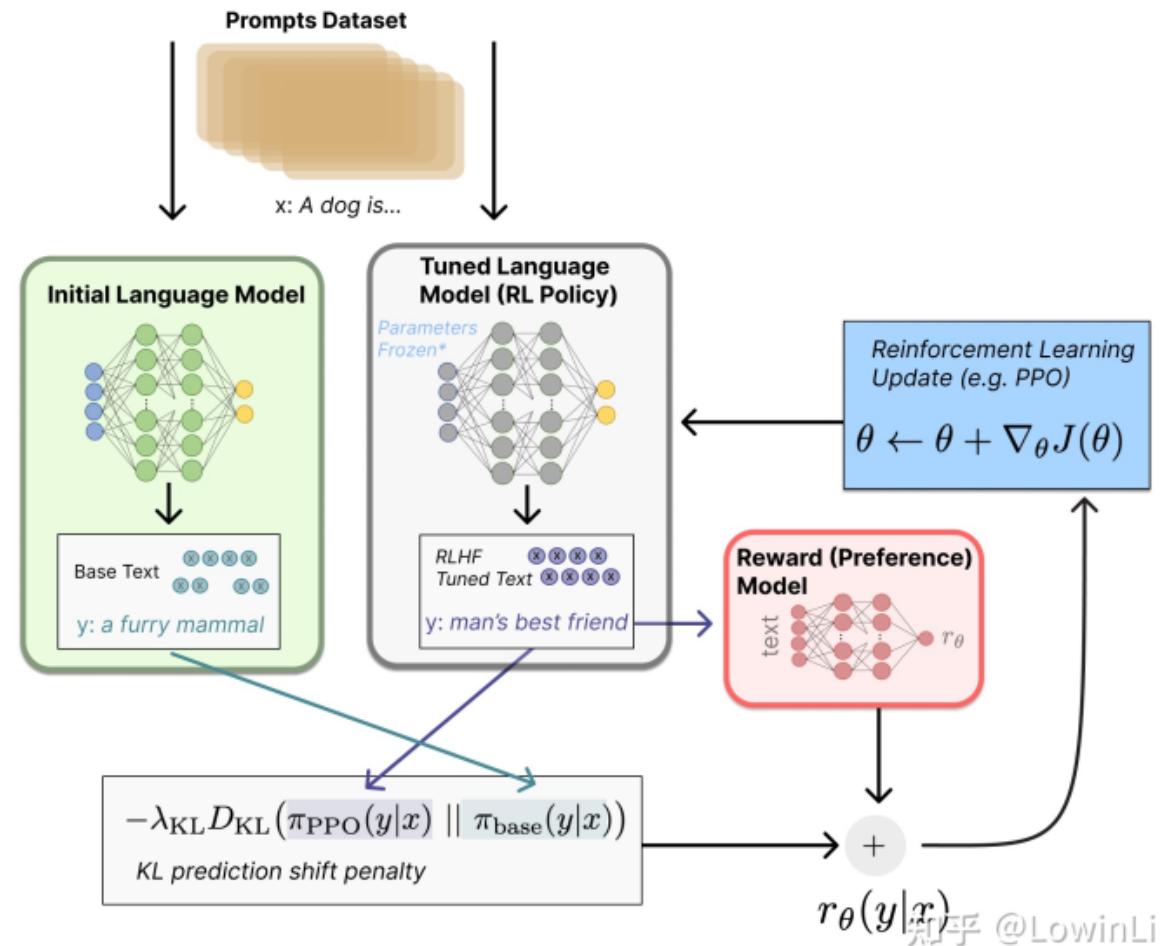
The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

$r_k$

# 人类反馈强化学习 (RLHF)



PPO:近端策略优化

# 大语言模型的涌现能力

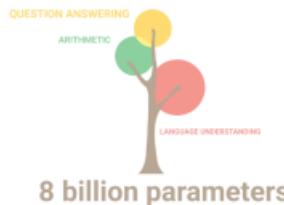
Published in Transactions on Machine Learning Research (08/2022)

## Emergent Abilities of Large Language Models

Jason Wei<sup>1</sup>  
Yi Tay<sup>1</sup>  
Rishi Bommasani<sup>2</sup>  
Colin Raffel<sup>3</sup>  
Barret Zoph<sup>1</sup>  
Sebastian Borgeaud<sup>4</sup>  
Dani Yogatama<sup>4</sup>  
Maarten Bosma<sup>1</sup>  
Denny Zhou<sup>1</sup>  
Donald Metzler<sup>1</sup>  
Ed H. Chi<sup>1</sup>  
Tatsunori Hashimoto<sup>2</sup>  
Oriol Vinyals<sup>4</sup>  
Percy Liang<sup>2</sup>  
Jeff Dean<sup>1</sup>  
William Fedus<sup>1</sup>

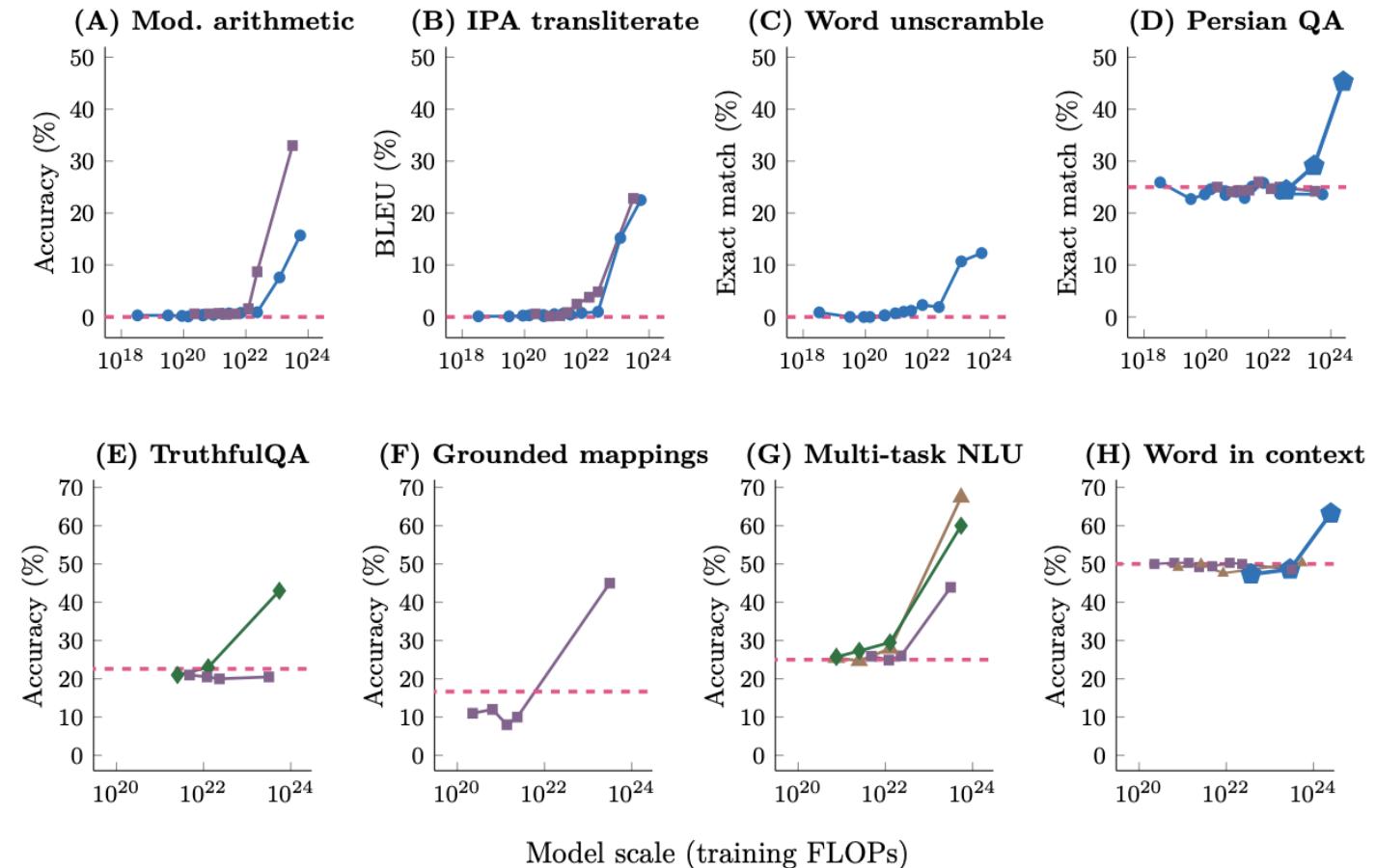
<sup>1</sup> Google Research <sup>2</sup>Stanford University <sup>3</sup> UNC Chapel Hill <sup>4</sup> DeepMind

Reviewed on OpenReview: <https://openreview.net/forum?id=yzkSU5zdwD>



jasonwei@google.com  
yitay@google.com  
nlprishi@stanford.edu  
craffel@gmail.com  
barretzoph@google.com  
sborgeaud@deepmind.com  
dyogatama@deepmind.com  
bosma@google.com  
dennyzhou@google.com  
metzler@google.com  
edchi@google.com  
thashim@stanford.edu  
vinyals@deepmind.com  
pliang@stanford.edu  
jeff@google.com  
liamfedus@google.com

LaMDA GPT-3 Gopher Chinchilla PaLM Random



# 大语言模型的Scaling Laws

## Scaling Laws for Neural Language Models

Jared Kaplan \*

Johns Hopkins University, OpenAI  
jaredk@jhu.edu

Sam McCandlish\*

OpenAI  
sam@openai.com

Tom Henighan

OpenAI  
henighan@openai.com

Tom B. Brown

OpenAI  
tom@openai.com

Benjamin Chess

OpenAI  
bchess@openai.com

Rewon Child

OpenAI  
rewon@openai.com

Scott Gray

OpenAI  
scott@openai.com

Alec Radford

OpenAI  
alec@openai.com

Jeffrey Wu

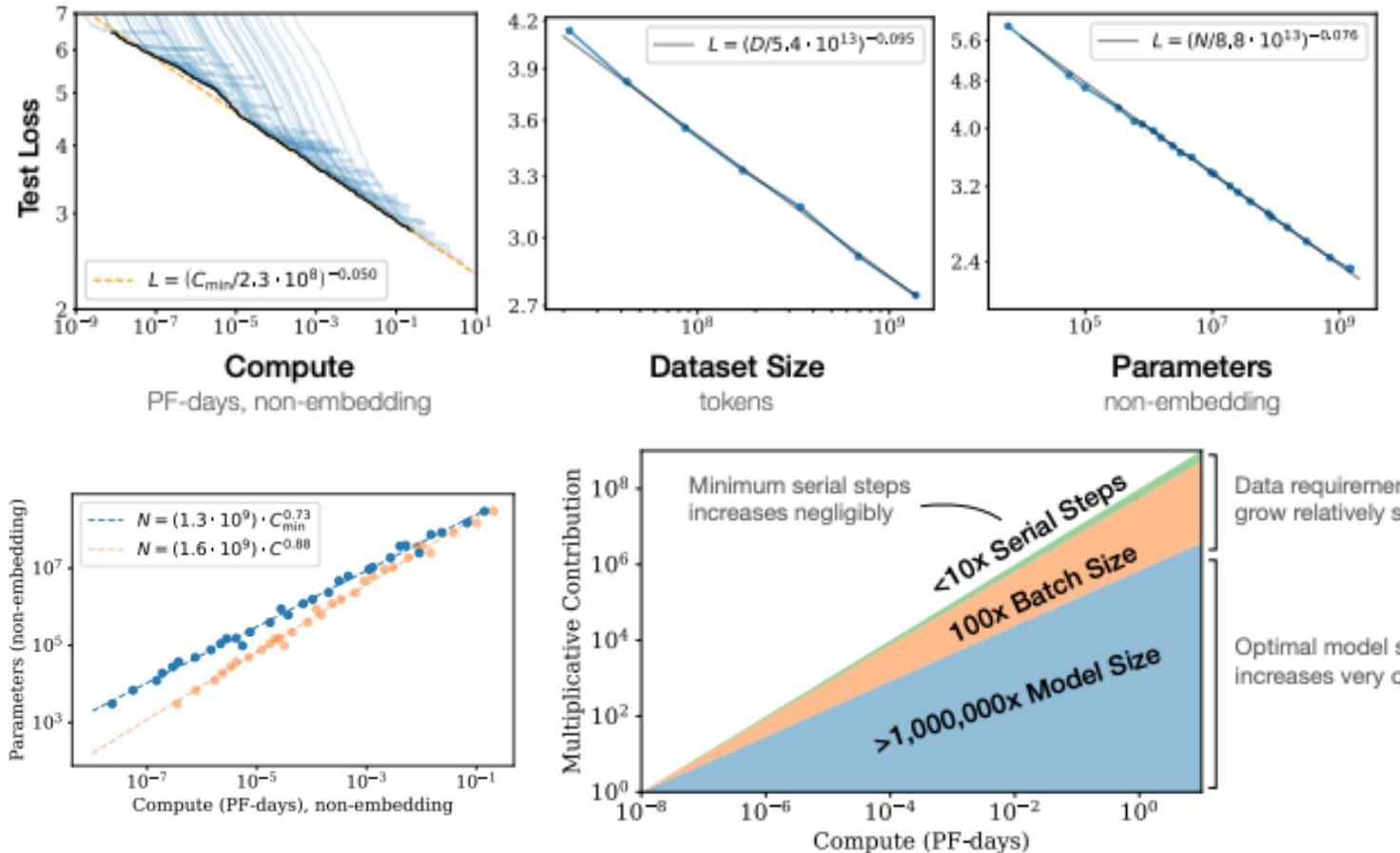
OpenAI  
jeffwu@openai.com

Dario Amodei

OpenAI  
damodei@openai.com

### Abstract

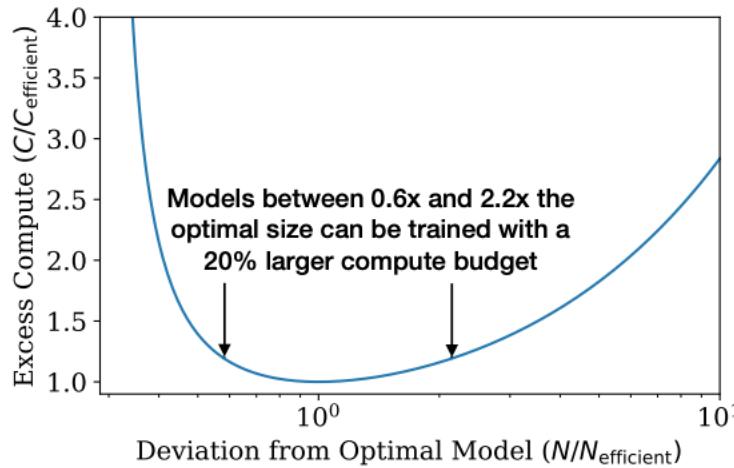
We study empirical scaling laws for language model performance on the cross-entropy loss. The loss scales as a power-law with model size, dataset size, and the amount of compute used for training, with some trends spanning more than seven orders of magnitude. Other architectural details such as network width or depth have minimal effects within a wide range. Simple equations govern the dependence of overfitting on model/dataset size and the dependence of training speed on model size. These relationships allow us to determine the optimal allocation of a fixed compute budget. Larger models are significantly more sample-efficient, such that optimally compute-efficient training involves training very large models on a relatively modest amount of data and stopping significantly before convergence.



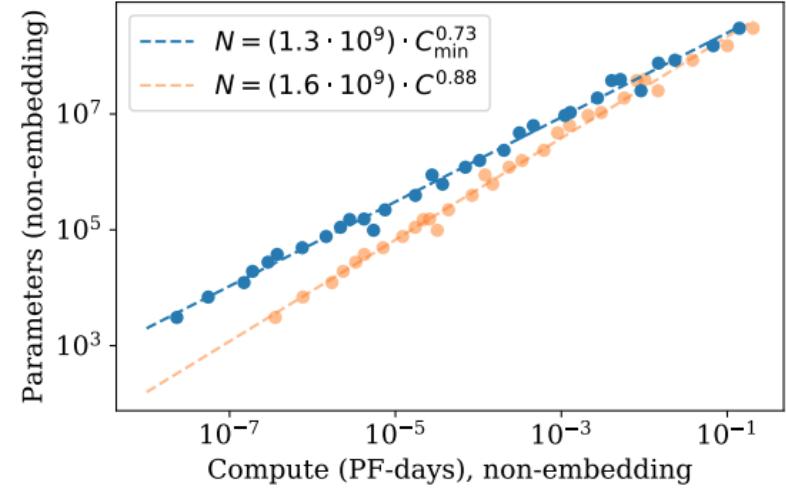
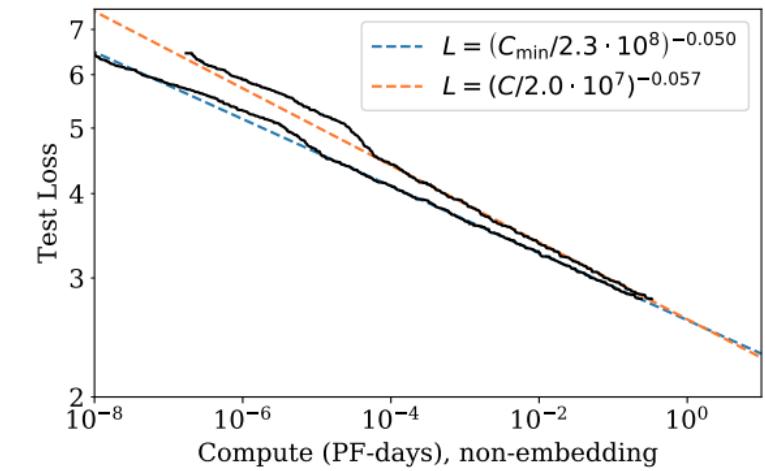
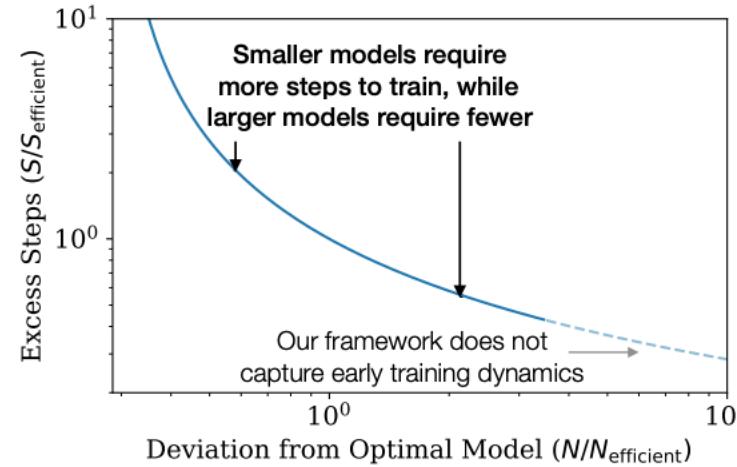
# 大语言模型的Scaling Laws

$$B_{\text{crit}}(L) \approx \frac{B_*}{L^{1/\alpha_B}}$$

$$S_{\min}(S) \equiv \frac{S}{1 + B_{\text{crit}}(L)/B}$$



$$C_{\min}(C) \equiv \frac{C}{1 + B/B_{\text{crit}}(L)}$$



# GPT: from LLM to AGI

## Compose music

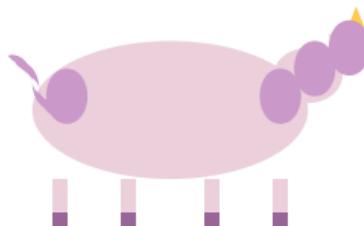
**Prompt:** Can you write a proof that there are infinitely many primes, with every line that rhymes?

**GPT-4:**

```
Q: Can you write a proof that there are infinitely many primes, with every line that rhymes?
A: Yes, I think I can,
Though it might take a clever plan.
I'll start by noting Euclid's proof,
Which shows that primes aren't just aloof.
Assume we have a finite list,
Of primes, and that none have been missed.
Multiply them all together,
And add one, just to be clever.
The result will be a number,
That has no prime factors, oh wonder!
But every number has a prime,
That divides it, it's just a matter of time.
So we've found a contradiction,
And our finite list needs eviction.
There must be infinitely many primes,
And that's the end of my rhyming lines.
```

**Prompt:** Draw a unicorn in TiKZ.

**GPT-4:** [Produces L<sup>A</sup>T<sub>E</sub>X compiling to following picture.]



## Explain jokes

What is funny about this image? Describe it panel by panel.



The image shows a package for a "Lightning Cable" adapter with three panels.

Panel 1: A smartphone with a VGA connector (a large, blue, 15-pin connector typically used for computer monitors) plugged into its charging port.

Panel 2: The package for the "Lightning Cable" adapter with a picture of a VGA connector on it.

Panel 3: A close-up of the VGA connector with a small Lightning connector (used for charging iPhones and other Apple devices) at the end.

This humor in this image comes from the absurdity of plugging a large,

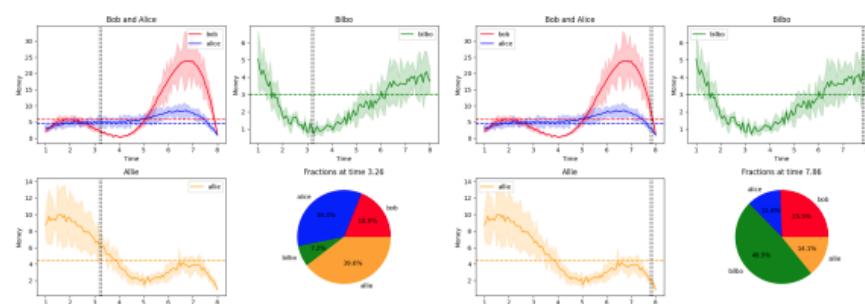
outdated VGA connector into a small, modern smartphone charging port.

Example prompt demonstrating GPT-4's visual input capability. The prompt consists of a short image with multiple panels which GPT-4 is able to answer.

**Prompt:** Can you generate a pyplot for the following data:  $x = [1, 3, 5, 6, 8]$ ,  $y1 = [2, 3, 5, 18, 1]$ ,  $y2 = [3, 5, 6, 8, 1]$ ,  $y3 = [5, 1, 2, 3, 4]$ ,  $y4 = [9, 7, 2, 3, 1]$ . I want  $y1$ ,  $y2$  to be in the same plot, but  $y3$  is in another plot next to that plot,  $y4$  is in below. I want the legend of  $y1$  to be "bob",  $y2$  to be "alice",  $y3$  to be "bilbo",  $y4$  to be "allie". I want the x-axis to be labeled with "time" and y axis to be labeled with "money". I want to add a 10 %-40% random error bar to each curve, through all times (including non-integers). I want smoothed curves to show the plot, and smoothed error bar. Do not use linear interpolation, use smooth interpolation! I want to also add some small zig-zag to the smoothed curve to make it look more real. I want to put a baseline as the mean of each line. I want to put a pie chart below indicating the fraction of the four people in each time step. I also want an animation to show how the fractions are changing in the pie chart in \*\*continuous time\*\*. Interpolate the missing fractions! I also want vertical line animation in other three plots to match the pie chart. I want the fanciest plot. Please add as many fancy things as possible.

Here is a possible code to generate a pyplot for the following data, with some comments to explain the steps:

**GPT-4:** [We give two snapshots from the animation resulting from the produced code]



## Sparks of Artificial General Intelligence: Early experiments with GPT-4

Sébastien Bubeck Varun Chandrasekaran Ronen Eldan Johannes Gehrke  
Eric Horvitz Ece Kamar Peter Lee Yin Tat Lee Yuanzhi Li Scott Lundberg  
Harsha Nori Hamid Palangi Marco Tulio Ribeiro Yi Zhang

Microsoft Research

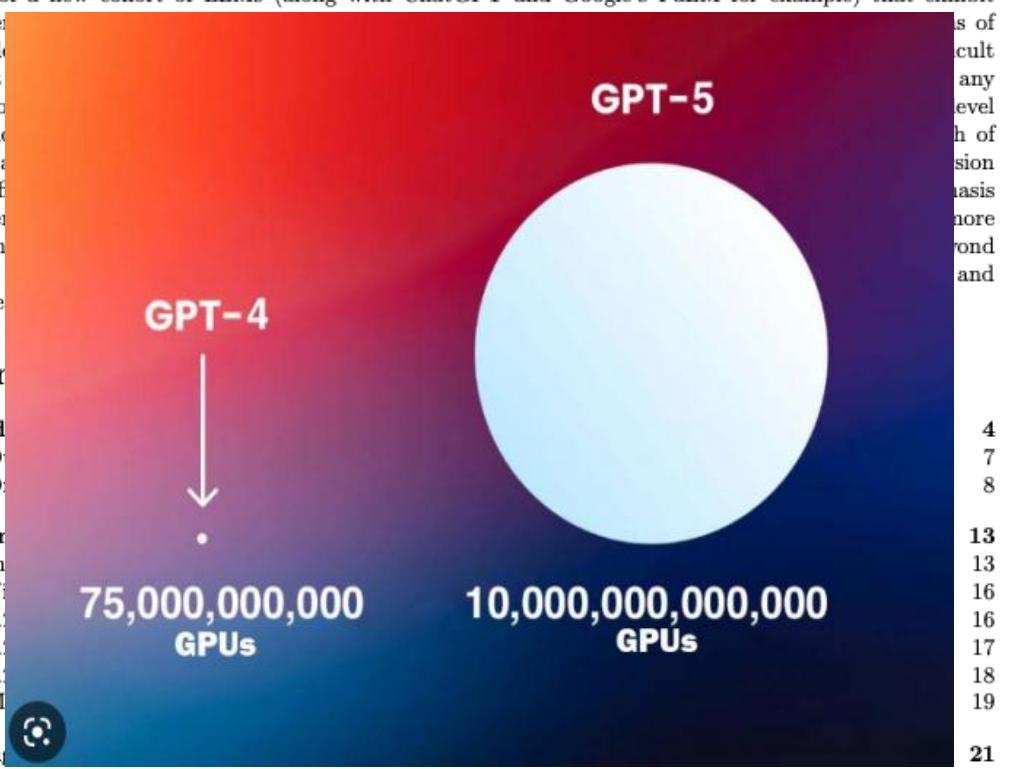
## Abstract

Artificial intelligence (AI) researchers have been developing and refining large language models (LLMs) that exhibit remarkable capabilities across a variety of domains and tasks, challenging our understanding of learning and cognition. The latest model developed by OpenAI, GPT-4 [Ope23], was trained using an unprecedented scale of compute and data. In this paper, we report on our investigation of an early version of GPT-4, when it was still in active development by OpenAI. We contend that (this early version of) GPT-4 is part of a new cohort of LLMs (along with ChatGPT and Google's PaLM for example) that exhibit more general and flexible abilities than previous models. These models are able to perform tasks that require special problem-solving skills, such as those involving complex reasoning or common sense knowledge. We find that GPT-4's capabilities are similar to those of an artificial general intelligence (AGI) system, such as an AI agent that can reason, learn, and act in complex environments. We also find that GPT-4's performance on various benchmarks is comparable to that of AGI systems, such as the AGI-1000 system developed by DeepMind [Dri23].

arXiv:2303.12712v3 [cs.CL] 27 Mar 2023

## Content

- 1 Introduction
  - 1.1 Overview
  - 1.2 Our approach
- 2 Multimodal processing
  - 2.1 Input modalities
  - 2.2 Visual processing
  - 2.3 Language processing
  - 2.4 Multimodal integration
- 3 Coding challenges
  - 3.1 From instructions to code
    - 3.1.1 Coding challenges
    - 3.1.2 Real world scenarios



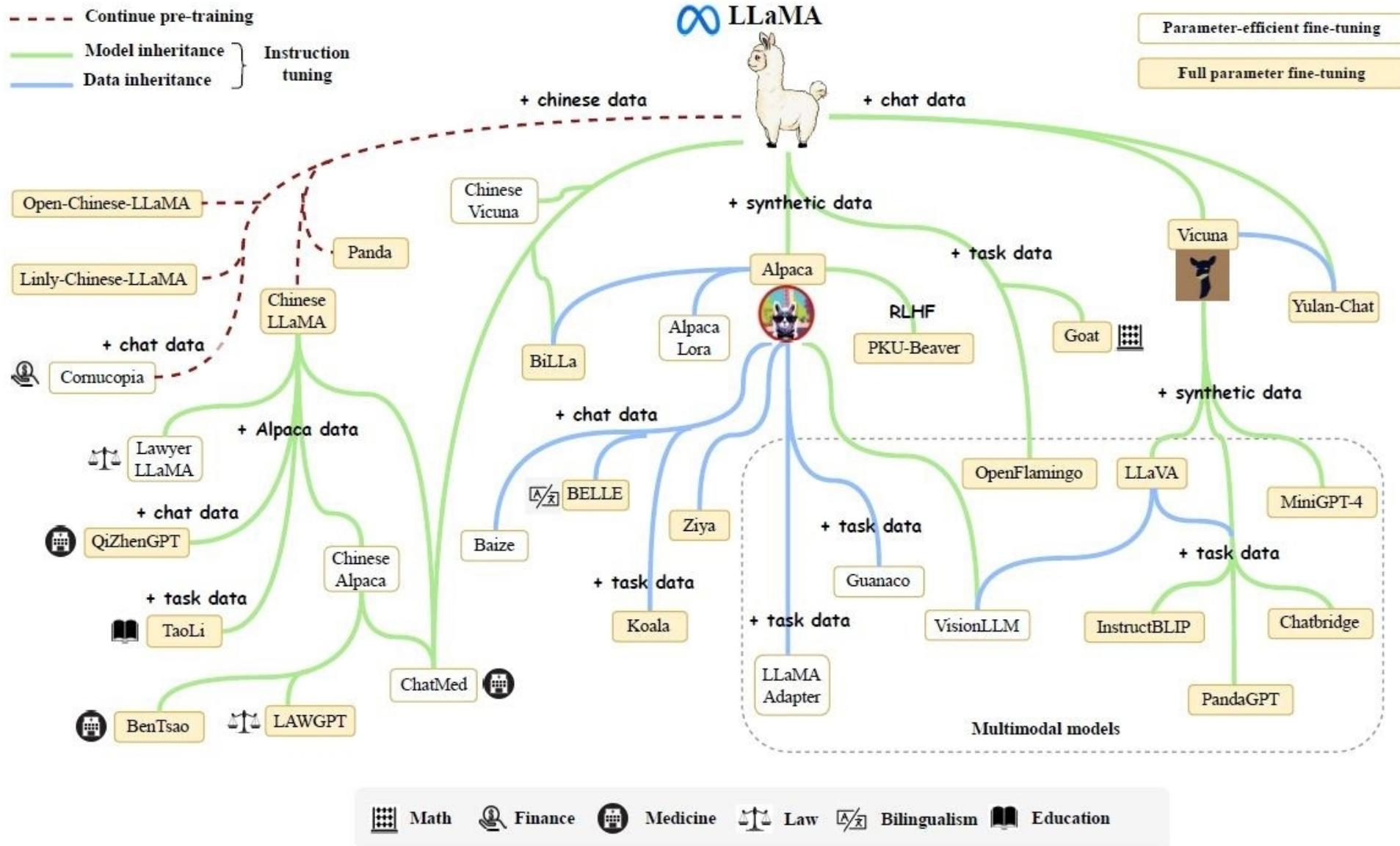


Fig. 4: An evolutionary graph of the research work conducted on LLaMA. Due to the huge number, we cannot include all the LLaMA variants in this figure, even much excellent work. To support incremental update, we share the source file of this figure, and welcome the readers to include the desired models by submitting the pull requests on our GitHub page.

# 结论

- **Prompt**工程是使用大模型的基本手段
  - **Prompt**已成为一种新的计算机语言
- **Transformer**是重要的大模型基本构建
- **GPT**的工作原理
- 未来大模型对人类的影响