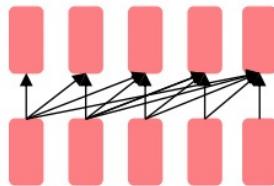


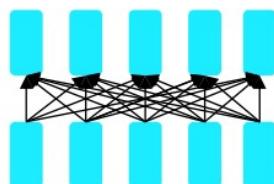
# Pretraining for three types of architectures

- The neural architecture influences the type of pretraining, and natural use cases.



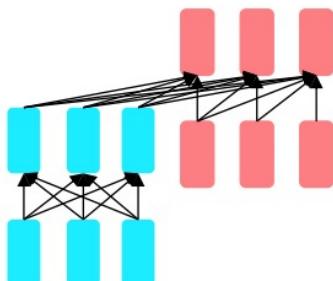
**Decoders**

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- Examples: GPT-2, GPT-3, LaMDA, LLaMa



**Encoders**

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- Examples: BERT and its many variants, e.g. RoBERTa

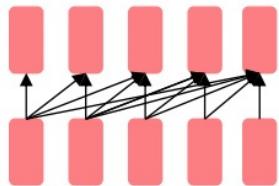


**Encoder-Decoders**

- Good parts of decoders and encoders?
- What's the best way to pretrain them?
- Examples: Transformer, T5, Meena

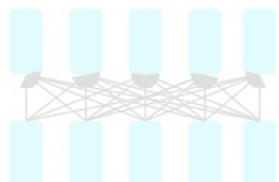
# Pretraining for three types of architectures

- The neural architecture influences the type of pretraining, and natural use cases.



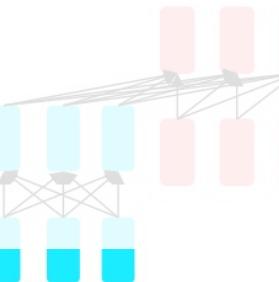
**Decoders**

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- Examples: GPT-2, GPT-3, LaMDA



**Encoders**

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- Examples: BERT and its many variants, e.g. RoBERTa



**Encoder-Decoders**

- Good parts of decoders and encoders?
- What's the best way to pretrain them?
- Examples: Transformer, T5, Meena

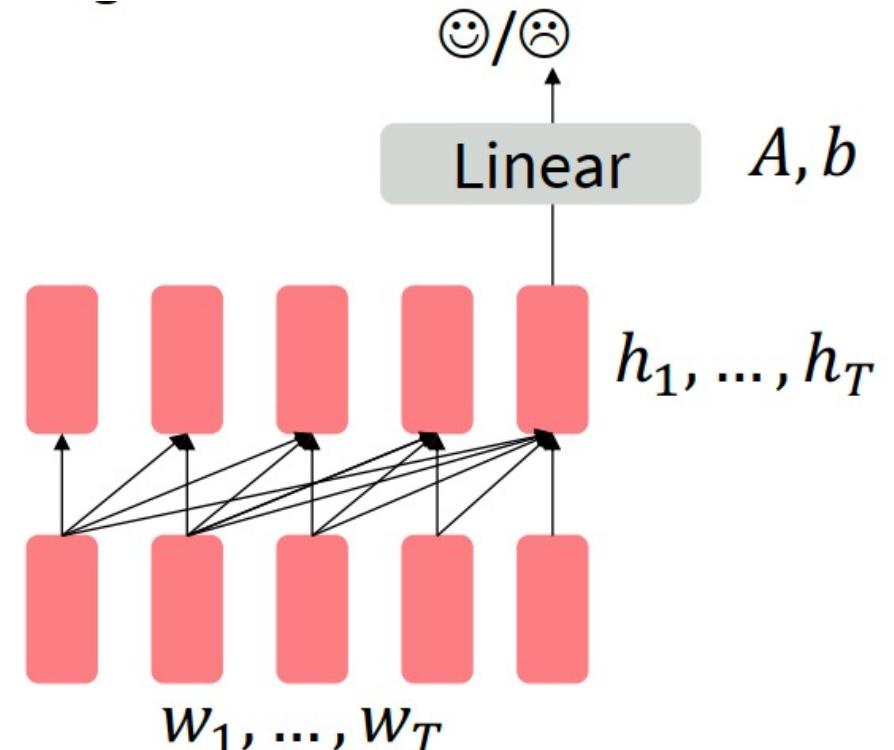
# Pretraining decoders

- 当我们使用利用LM预训练的decoder时，暂时忘记LM是为了建模  $p(w_t|w_{1:t-1})$

- 我们利用最后一个词的隐状态来训练一个分类器，从而微调这个预训练模型

$$\begin{aligned} h_1, \dots, h_T &= \text{Decoder}(w_1, \dots, w_T) \\ y &\sim Ah_T + b \end{aligned}$$

- $A, b$ 是随机初始化的参数，用下游模型来训练
- 梯度回传整个网络



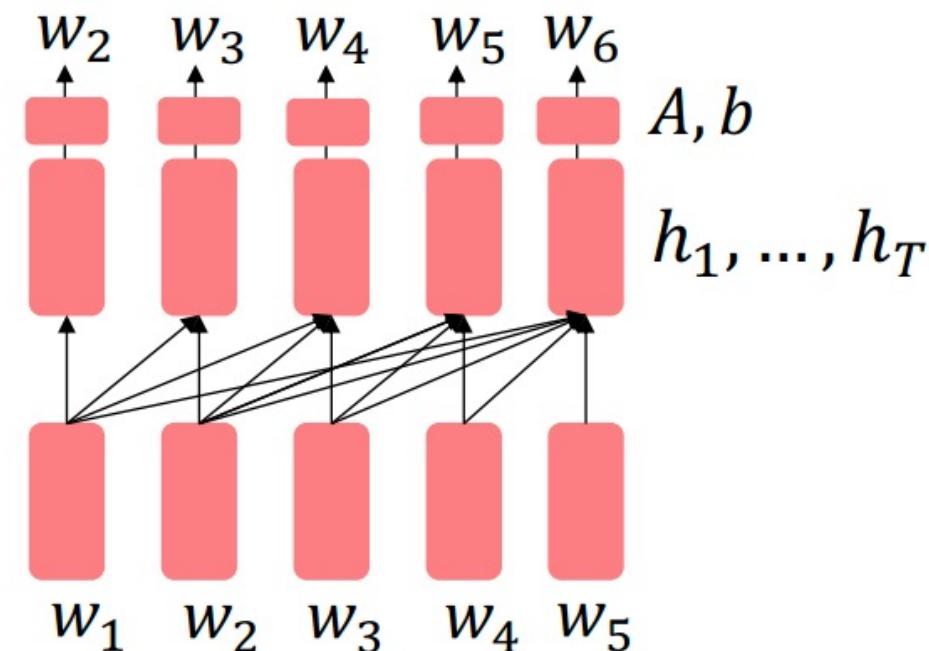
[Note how the linear layer hasn't been pretrained and must be learned from scratch.]

# Pretraining decoders

- 下游任务如果是生成任务，那么很自然的，我们要微调  $p(w_t|w_{1:t-1})$
- 如果任务的输出是个序列，而且词表与预训练数据高度重合，那么将会非常有效
  - 对话（上下文=对话历史）
  - 摘要（上下文=文档）

$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$
$$w_t \sim Ah_{t-1} + b$$

- $A, b$ 是随机初始化的参数，用下游模型来训练



[Note how the linear layer has been pretrained.]

# Generative Pretrained Transformer (GPT) [Radford et al., 2018]

2018's GPT was a big success in pretraining a decoder!

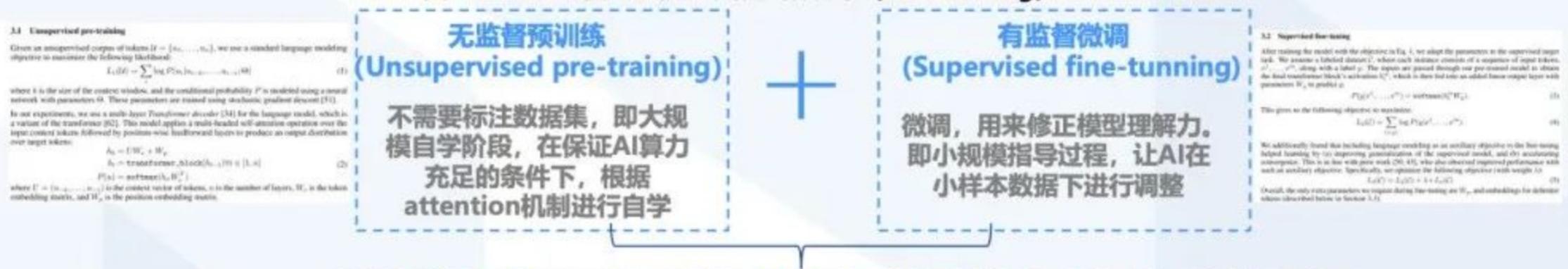
- Transformer decoder with 12 layers.
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Byte-pair encoding with 40,000 merges
- Trained on BooksCorpus: over 7000 unique books.
  - Contains long spans of contiguous text, for learning long-distance dependencies

# GPT-1（预训练+微调）

■ GPT-1模型基于Transformer解除了顺序关联和依赖性的前提，采用生成式模型方式，重点考虑了从原始文本中有效学习的能力，这对于减轻自然语言处理（NLP）中对监督学习的依赖至关重要

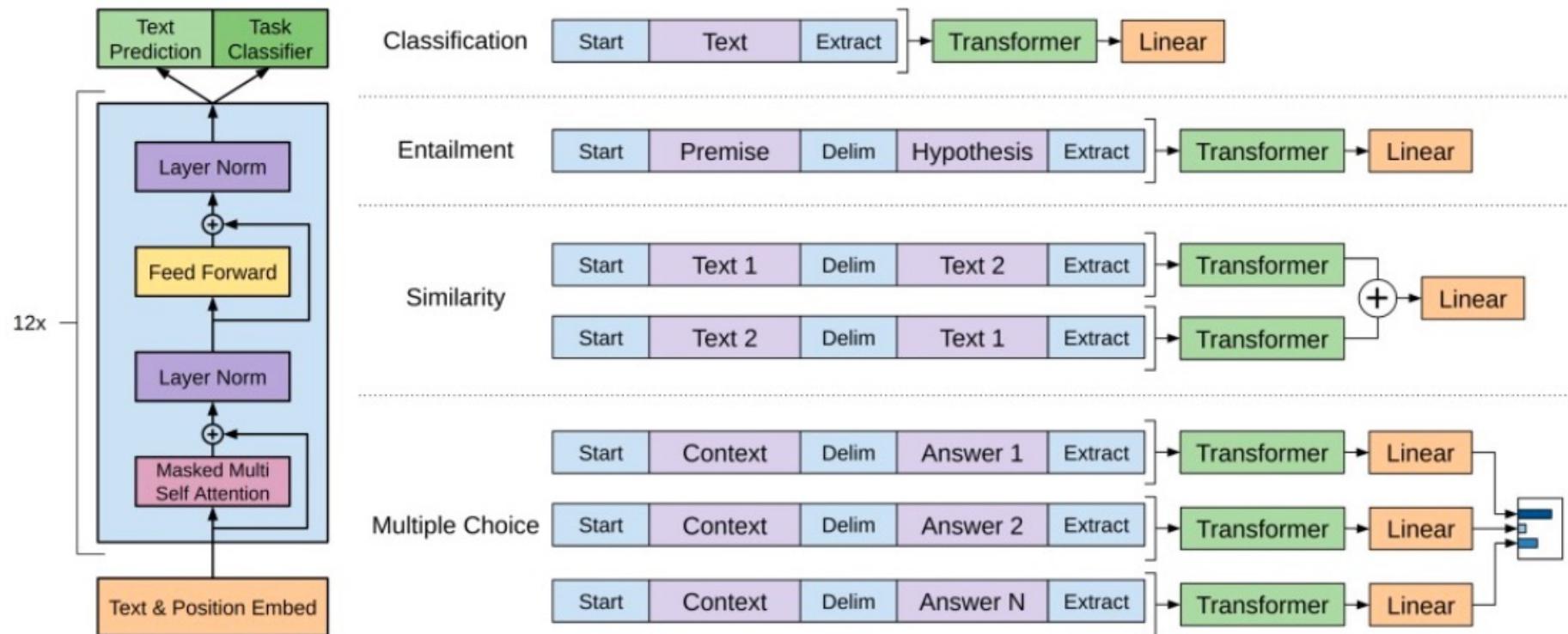
- ✓ GPT (Generative Pre-training Transformer) 于2018年6月由OpenAI首次提出。GPT模型考虑到在自然语言理解中有大量不同的任务，尽管大量的未标记文本语料库非常丰富，但用于学习这些特定任务的标记数据却很少，这使得经过区分训练的模型很难充分执行。同时，大多数深度学习方法需要大量手动标记的数据，这限制了它们在许多缺少注释资源的领域的适用性。
- ✓ 在考虑以上局限性的前提下，GPT论文中证明，通过对未标记文本的不同语料库进行语言模型的生成性预训练，然后对每个特定任务进行区分性微调，可以实现这些任务上的巨大收益。和之前方法不同，GPT在微调期间使用任务感知输入转换，以实现有效的传输，同时对模型架构的更改最小。

图29：GPT-1模型的核心手段是预训练（Pre-training）



# Generative Pretrained Transformer (GPT) [Radford et al., 2018]

- How do we format inputs to our decoder for finetuning tasks?



- The linear classifier is applied to the representation of the [EXTRACT] token.

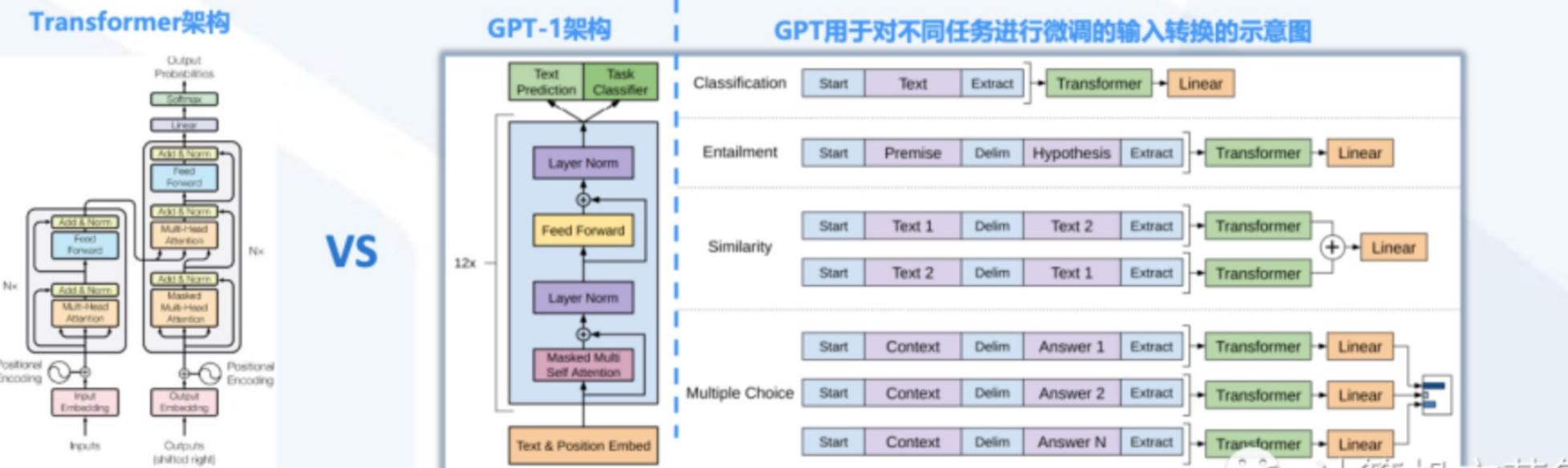
# GPT架构对于Transformer的改进

## ■ GPT相比于Transformer等模型进行了显著简化

- ✓ 相比于Transformer，GPT训练了一个12层仅decoder的解码器（原Transformer模型中包含Encoder和Decoder两部分）。
- ✓ 相比于Google的BERT(Bidirectional Encoder Representations from Transformers,双向编码生成Transformer)，**GPT仅采用上文预测单词** (BERT采用了基于上下文双向的预测手段)。

注：ChatGPT的表现更贴近人类意图，部分因为一开始GPT是基于上文的预测，这更贴近人类的话语模式，因为人类言语无法基于将来的话来做分析。

图30：GPT-1模型相比于Transformer模型有了显著简化

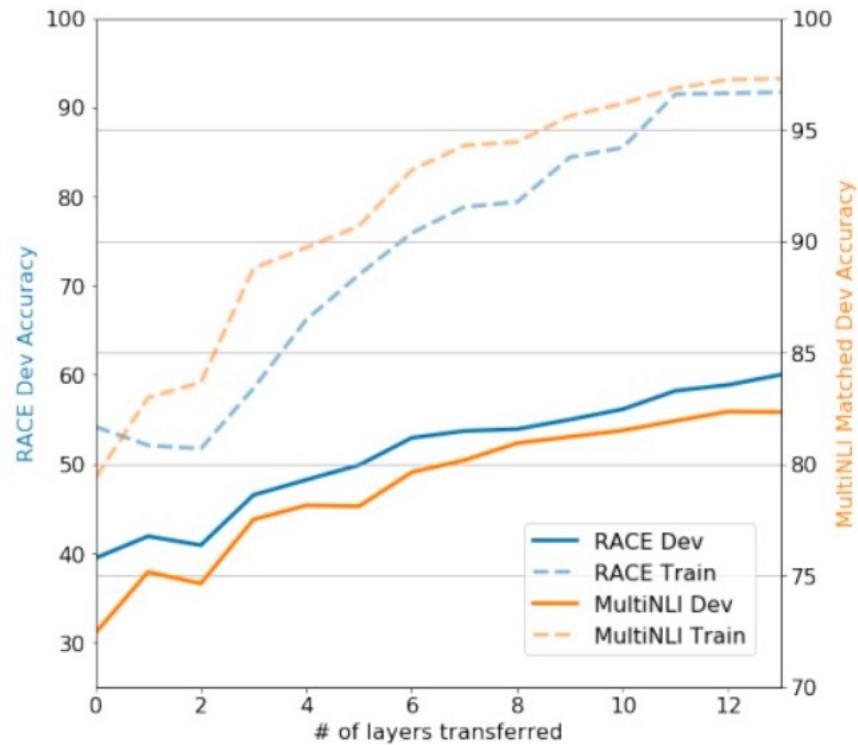


# Generative Pretrained Transformer (GPT) [Radford et al., 2018]

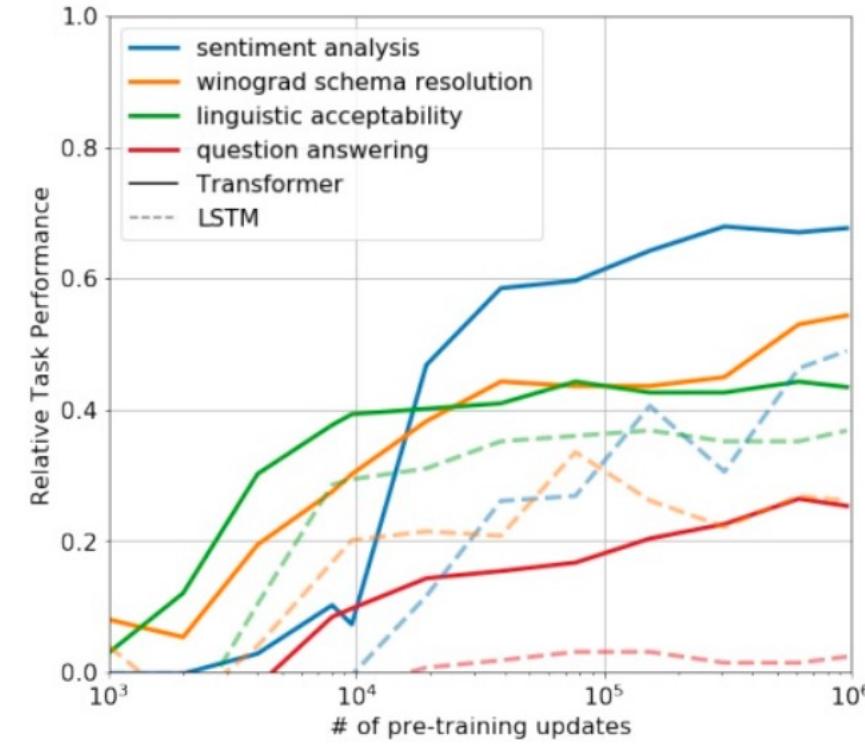
- GPT results on various natural language inference datasets

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	<u>89.3</u>	-	-	-
CAFE [58] (5x)	80.2	79.0	<u>89.3</u>	-	-	-
Stochastic Answer Network [35] (3x)	<u>80.6</u>	<u>80.1</u>	-	-	-	-
CAFE [58]	78.7	77.9	88.5	<u>83.3</u>		
GenSen [64]	71.4	71.3	-	-	<u>82.3</u>	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	<b>61.7</b>
Finetuned Transformer LM (ours)	<b>82.1</b>	<b>81.4</b>	<b>89.9</b>	<b>88.3</b>	<b>88.1</b>	56.0

# Examining the Effect of Pretraining in GPT [Radford et al., 2018]



Transformer层数迁移的越多，  
RACE和MultiNLI的效果越好



Transformer和LSTM的zero-shot效果  
随着预训练更新数的关系

# Increasingly convincing generations (GPT2) [Radford et al., 2018]

- GPT2, 更大的GPT，用更多数据预训练
- 可以生成更加可信的自然语言文本

**Context (human-written):** In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

**GPT-2:** The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

# GPT2 解决 zero-shot问题

- 目的：成为更通用的模型，可以解决多任务
- 去掉有监督微调，只用无监督预训练
- 更大，1.5B参数，48层Transformer，40GB文本预训练
- 在无监督NLP任务中具有决定性优势，依然无法打败有监督模型

图32：GPT-2尚未解决诸多瓶颈问题

## 存在的问题01：

从实用的角度来看，每一项新任务都需要一个标记示例的大数据集，这限制了语言模型的适用性；

对于其中的许多任务（从纠正语法到生成抽象概念的示例，再到评论一个短篇故事等等），很难收集一个大型的监督训练数据集，特别是当每个新任务都必须重复该过程时。



## 存在的问题02：

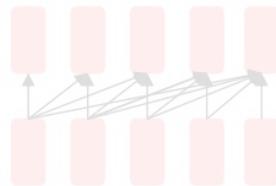
- 预训练加微调范式中，可能在这种范式下实现的泛化可能很差，因为该模型过于特定于训练分布，并且在其之外无法很好地泛化。
- 微调模型在特定基准上的性能，即使名义上是人类水平，也可能夸大基础任务的实际性能。

## 存在的问题03：

因为人类学习大多数语言任务不需要大型受监督的数据集，当前NLP技术在概念上具有一定的局限性。

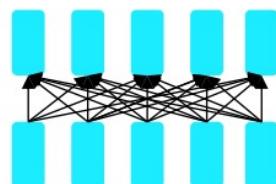
# Pretraining for three types of architectures

- The neural architecture influences the type of pretraining, and natural use cases.



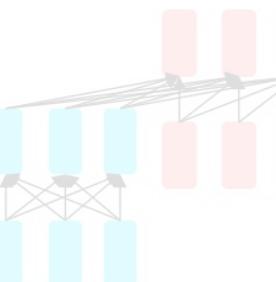
Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- Examples: GPT-2, GPT-3, LaMDA



Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- Examples: BERT and its many variants, e.g. RoBERTa

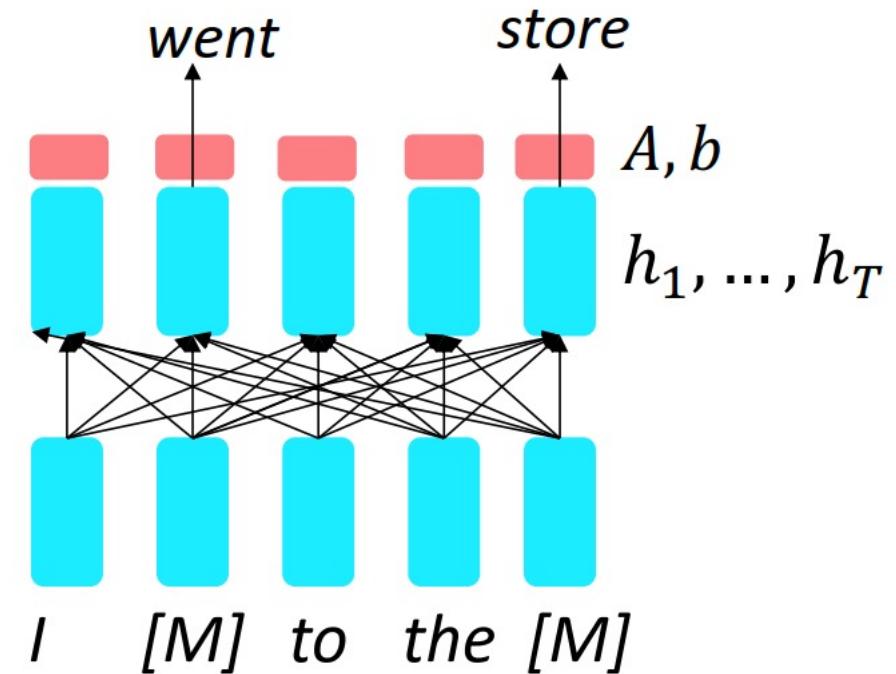


Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?
- Examples: Transformer, T5, Meena

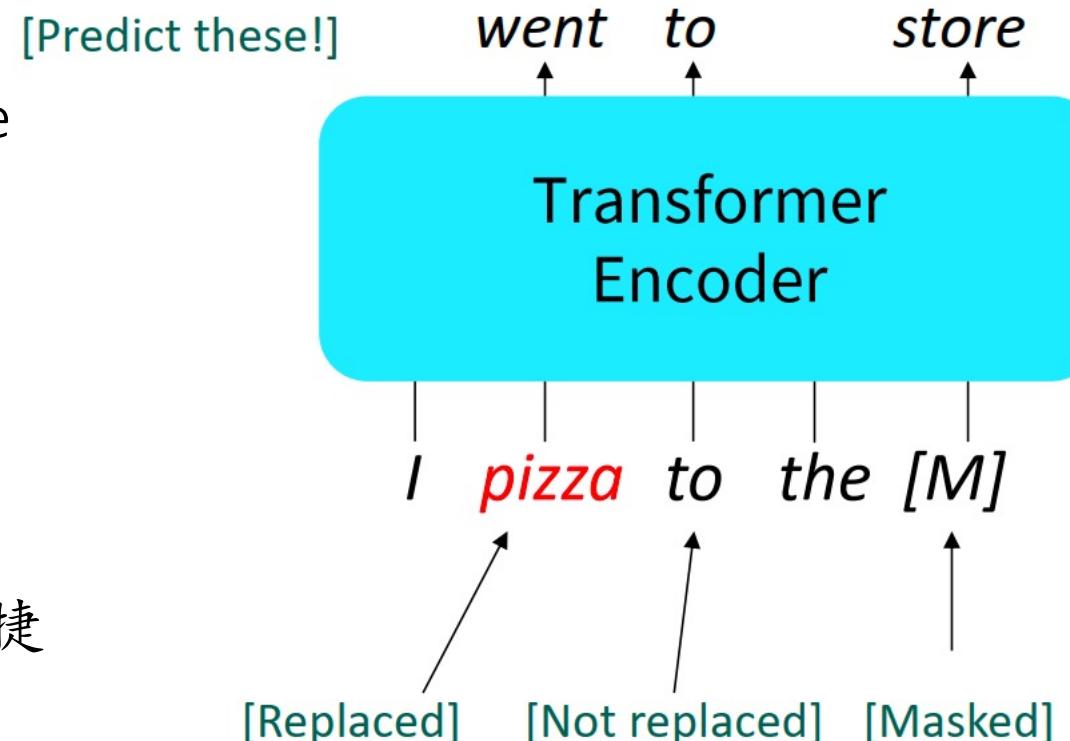
# Pretraining encoders: what pretraining objective to use?

- 我们刚刚学习了语言模型的预训练，但encoder可以接受两边的信息，所以不能用语言模型的方法来预训练
- Idea: 用[MASK]符号替换掉输入文本中的一部分词，让模型来预测这部分词
- 只在被mask out的位置加loss function
- 令  $\tilde{x}$  为被mask掉一些词的句子x，则我们想要学习的是  $p_\theta(x|\tilde{x})$
- 此为**masked LM**



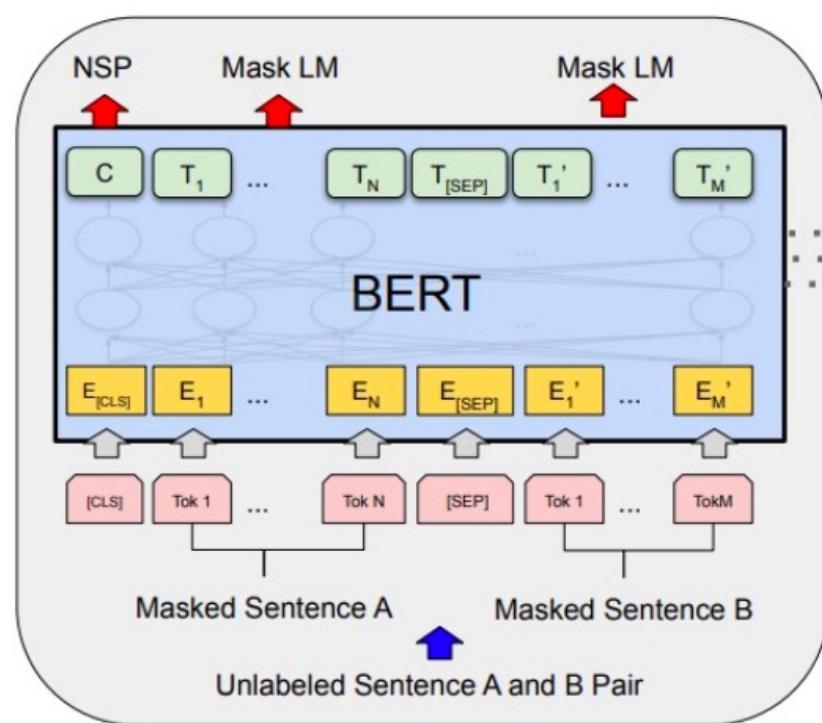
# BERT: Bidirectional Encoder Representations from Transformers

- BERT的mask LM方法
- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
- 为什么这么折腾?
  - 不让模型捕捉到某种固定模式，不让找捷径，逼迫它按照我们的想法去学习

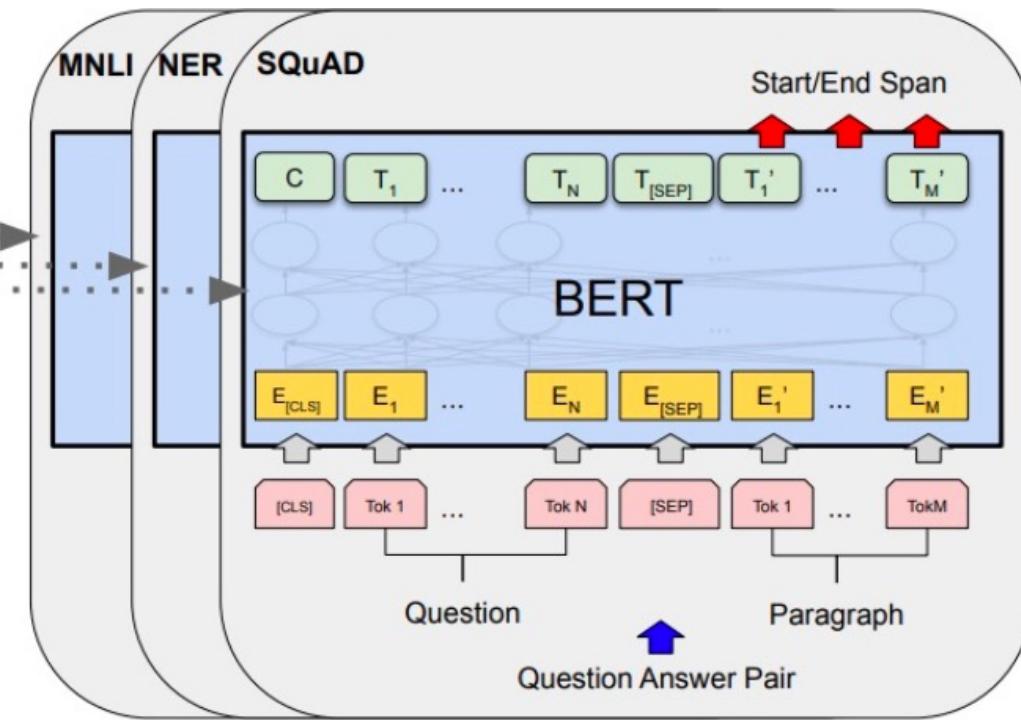


# BERT: Bidirectional Encoder Representations from Transformers

- 预训练结构与微调结构大同小异



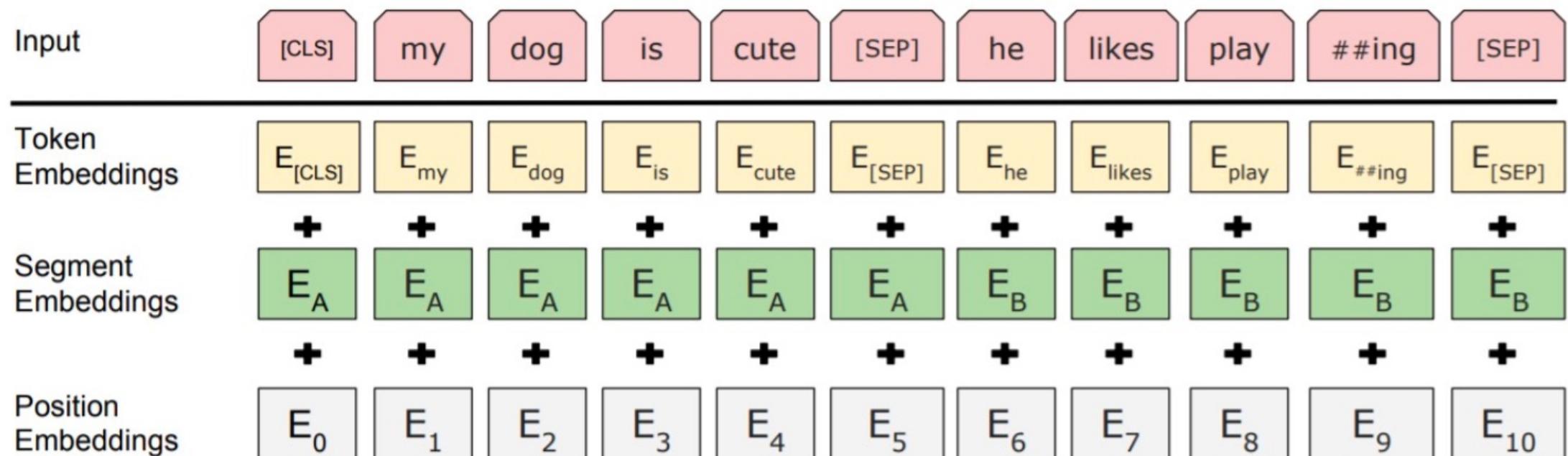
Pre-training



Fine-Tuning

# BERT: Bidirectional Encoder Representations from Transformers

- 第二个预训练任务：输入到BERT的文本包含两部分
- BERT需要预测，这两个句子是接续关系还是随机采样出来的（其实这个是不必要的）



# BERT: Bidirectional Encoder Representations from Transformers

## Details about BERT

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
  - (TPUs are special tensor operation acceleration hardware)
- Finetuning is practical and common on a single GPU
  - “Pretrain once, finetune many times.”

# BERT: Bidirectional Encoder Representations from Transformers

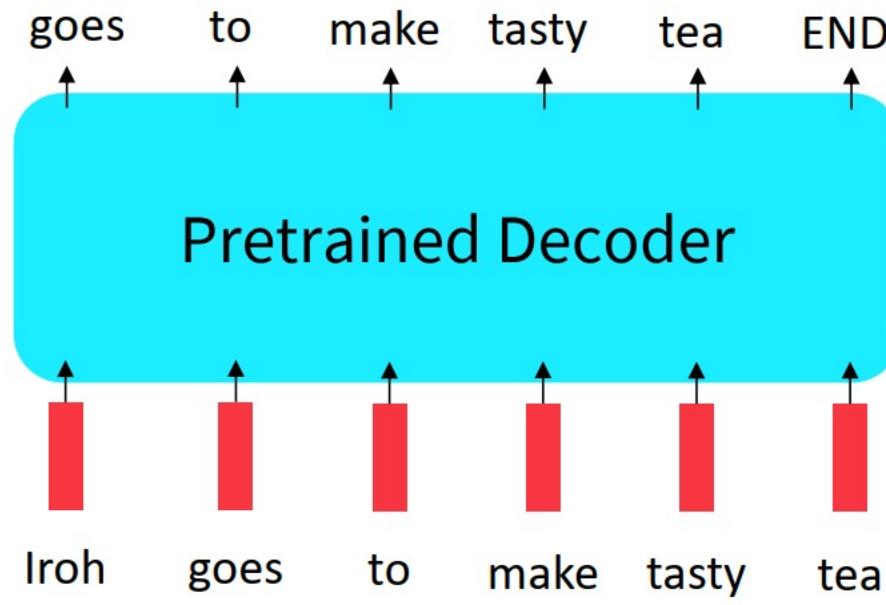
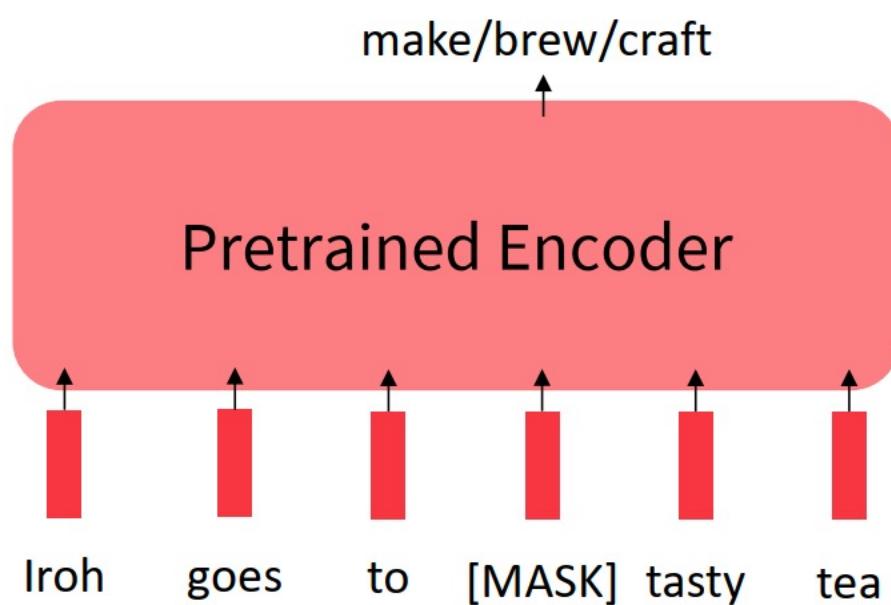
- BERT非常受欢迎，用途广泛;对BERT的微调为广泛的任务带来了新的最高结果。
- QQP: Quora Question Pairs (detect paraphrase questions)
- QNLI: natural language inference over question answering data
- SST-2: sentiment analysis
- CoLA: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- STS-B: semantic textual similarity
- MRPC: microsoft paraphrase corpus
- RTE: a small natural language inference corpus

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>

Note that BERT<sub>BASE</sub> was chosen to have the same number of parameters as OpenAI GPT.

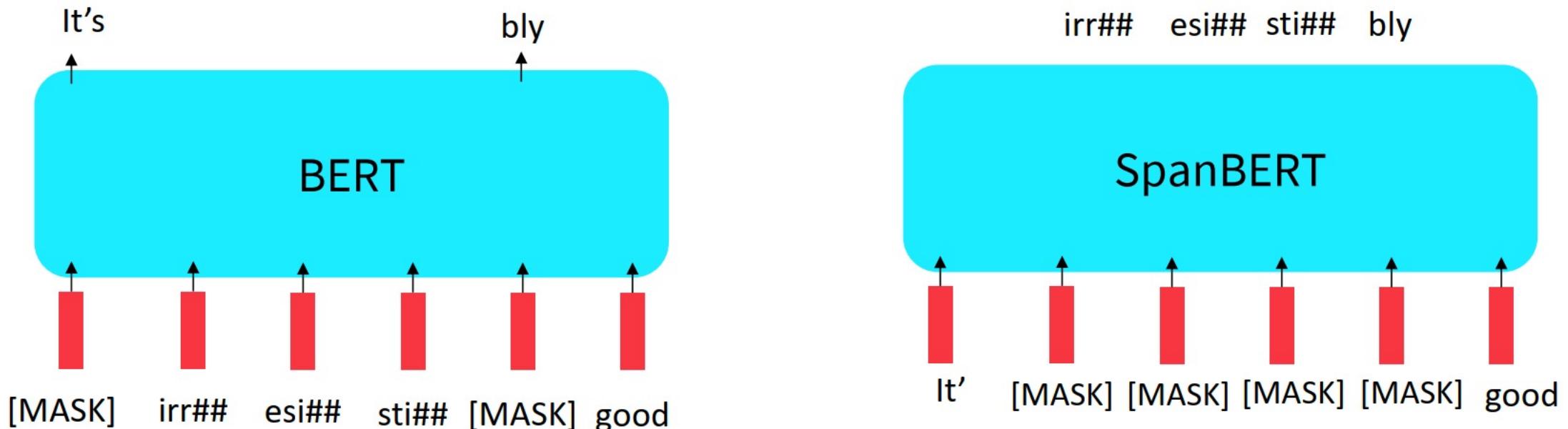
# Limitations of pretrained encoders

- 看起来结果不错，那么可以把pretrained encoders应用在所有任务上吗
- 如果任务包含文本生成，那就要考虑用pretrained decoder
- BERT与其他的预训练encoder无法提供天然的自回归性



# Extensions of BERT

- BERT有很多变种RoBERTa, SpanBERT
  - RoBERTa: mainly just train BERT for longer and remove next sentence prediction!
  - SpanBERT: masking contiguous spans of words makes a harder, more useful pretraining task



# Extensions of BERT

- 一句话总结RoBERTa：更多的训练与更多的数据加持之下，即使不改BERT结构，性能也能提升

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
<b>RoBERTa</b>						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	<b>94.6/89.4</b>	<b>90.2</b>	<b>96.4</b>
<b>BERT<sub>LARGE</sub></b>						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

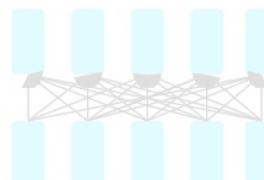
# Pretraining for three types of architectures

- The neural architecture influences the type of pretraining, and natural use cases.



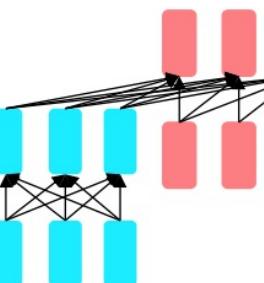
Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- Examples: GPT-2, GPT-3, LaMDA



Encoders

- Gets bidirectional context – can condition on future!
- Wait, how do we pretrain them?
- Examples: BERT and its many variants, e.g. RoBERTa



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?
- Examples: Transformer, T5, Meena

# Pretraining encoder-decoders: what pretraining objective to use?

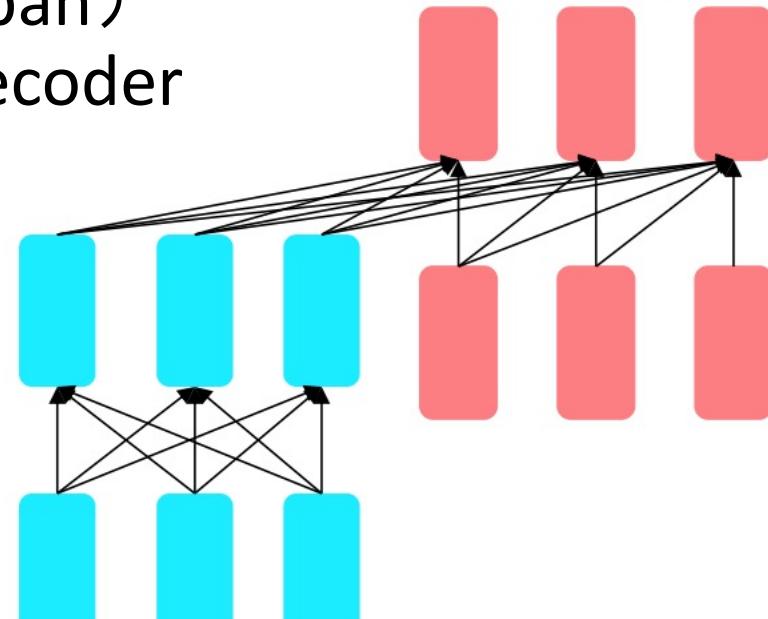
- Raffel et al., 2018发现最有效的方法是span corruption: T5
- 在输入中把不同长度的单词子序列 (span) 替换为占位符 (placeholder) , 并在decoder 中把之前替换掉的span解码出来

Original text

Thank you for inviting me to your party last week.

这个在预处理阶段完成。Decoder看来这还是个类似LM的任务

Targets  
<X> for inviting <Y> last <Z>

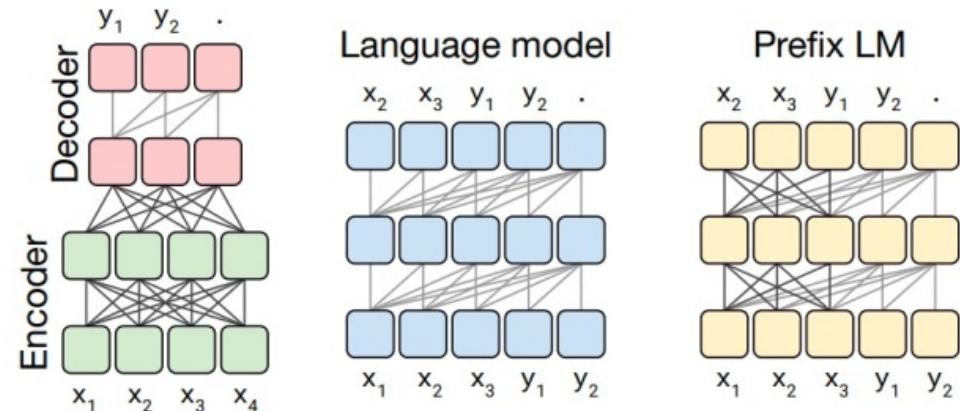


Inputs

Thank you <X> me to your party <Y> week.

# Pretraining encoder-decoders: what pretraining objective to use?

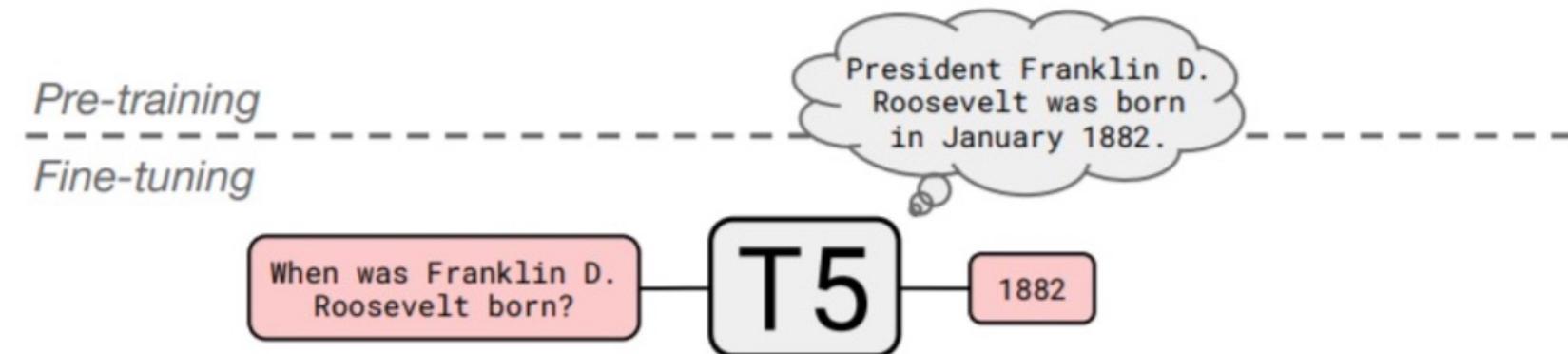
- Raffel et al., 2018发现
  - encoder-decoder比单纯的decoder效果好
  - Span corruption比单纯的LM效果好



Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	$M$	<b>83.28</b>	<b>19.24</b>	<b>80.88</b>	<b>71.36</b>	<b>26.98</b>	<b>39.82</b>	<b>27.65</b>
Enc-dec, shared	Denoising	$P$	$M$	82.81	18.78	<b>80.63</b>	<b>70.73</b>	26.72	39.03	<b>27.46</b>
Enc-dec, 6 layers	Denoising	$P$	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	$P$	$M$	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	$P$	$M$	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	$M$	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	$P$	$M$	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	$P$	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	$P$	$M$	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	$P$	$M$	79.68	17.84	76.87	64.86	26.28	37.51	26.76

# Pretraining encoder-decoders: what pretraining objective to use?

- T5的一个迷人特性：它可以进行微调以回答各种问题，从其参数中检索知识。
- NQ: Natural Questions
- WQ: WebQuestions
- TQA: Trivia QA
- All “open-domain” versions



	NQ	WQ	TQA	
			dev	test
Karpukhin et al. (2020)	<b>41.5</b>	42.4	<b>57.9</b>	—
T5.1.1-Base	25.7	28.2	24.2	30.6
T5.1.1-Large	27.3	29.5	28.5	37.2
T5.1.1-XL	29.5	32.4	36.0	45.1
T5.1.1-XXL	32.8	35.6	42.9	52.5
T5.1.1-XXL + SSM	35.2	<b>42.8</b>	51.9	<b>61.6</b>

# Contents

- Transformers
  - Impact of Transformers on NLP (and ML more broadly)
  - From Recurrence (RNNs) to Attention-Based NLP Models
  - Understanding the Transformer Model
  - Drawbacks and Variants of Transformers
- Pretraining Language Models(PLMs)
  - Subword modeling
  - Motivating model pretraining from word embeddings
  - Model pretraining three ways
    - Decoders
    - Encoders
    - Encoder-Decoders
  - **Very large models and in-context learning**

# GPT-3简介

- ▶ GPT-3 (Generative Pre-trained Transformer 3) 是一个自回归语言模型，目的是为了使用深度学习生成人类可以理解的自然语言。
- ▶ GPT-3是由在旧金山的人工智能公司OpenAI训练与开发，模型设计基于谷歌开发的变换语言模型。
- ▶ GPT-3的神经网络包含1750亿个参数，在发布时为参数最多的神经网络模型。
- ▶ OpenAI于2020年5月发表GPT-3的论文，在次月为少量公司与开发团队发布应用程序界面的测试版。
- ▶ 微软在2020年9月22日宣布取得了GPT-3的独家授权。

# GPT3训练数据来源

Dataset	Tokens	Assumptions	Tokens per byte	Ratio	Size
	(billion)		(Tokens / bytes)		(GB)
<b>Web data</b>	<b>410B</b>	–	0.71	1:1.9	<b>570</b>
<b>WebText2</b>	<b>19B</b>	<i>25% &gt; WebText</i>	0.38	1:2.6	50
<b>Books1</b>	<b>12B</b>	<i>Gutenberg</i> 	0.57	1:1.75	21
<b>Books2</b>	<b>55B</b>	<i>Bibliotik</i>	0.54	1:1.84	101
<b>Wikipedia</b>	<b>3B</b>	<i>See RoBERTa</i>	0.26	1:3.8	11.4
Total	<b>499B</b>			753.4GB	

Table. GPT-3 Datasets. Disclosed in **bold**. Determined in *italics*.

# GPT3训练成本

看一下大语言模型训练的token数量：

- ▶ GPT-3 (2020.5) 是500B (5000亿), 目前最新数据为止;
- ▶ Google的PaLM (2022.4) 是780B;
- ▶ DeepMind的Chinchilla是1400B;
- ▶ Pangu-α 公布了训练的token数, 约为40B, 不到GPT-3的十分之一;
- ▶ 国内其他的大模型都没有公布训练的token数。



# GPT-3, In-context learning, and very large models

- 目前，我们与PLM打交道有两种方法：
  - 从它们定义的概率分布中进行采样 (prompt learning)
  - 在一个下游任务上微调，利用它们的预测结果
- 意外收获：非常大的语言模型似乎在执行下游任务的学习时，不需要任何梯度步骤，只需在其上下文中提供一些示例。
- GPT3
  - T5: 11 billion 参数
  - GPT3: 175 billion 参数

# GPT-3, In-context learning, and very large models

- The in-context examples seem to specify the task to be performed, and the conditional distribution mocks performing
- Input (prefix within a single Transformer decoder context):
  - “ thanks -> merci
  - hello -> bonjour
  - mint -> menthe
  - otter -> ”
- Output (conditional generations):
  - loutre...”

Tradit

Fine-tu

The mo

large c



## The three settings we explore for in-context learning

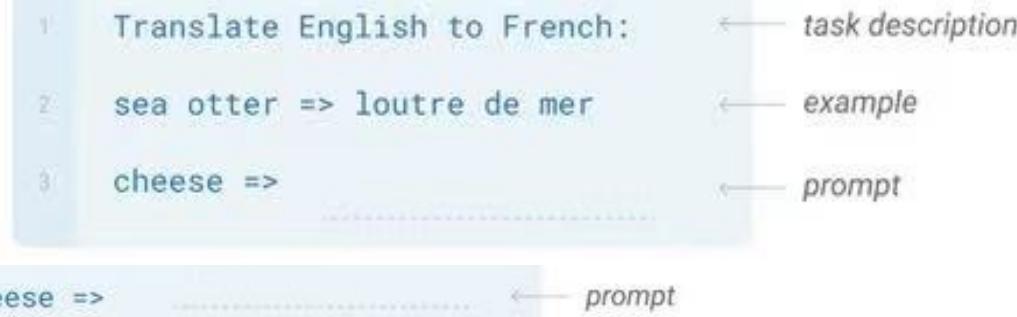
### Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



### One-shot

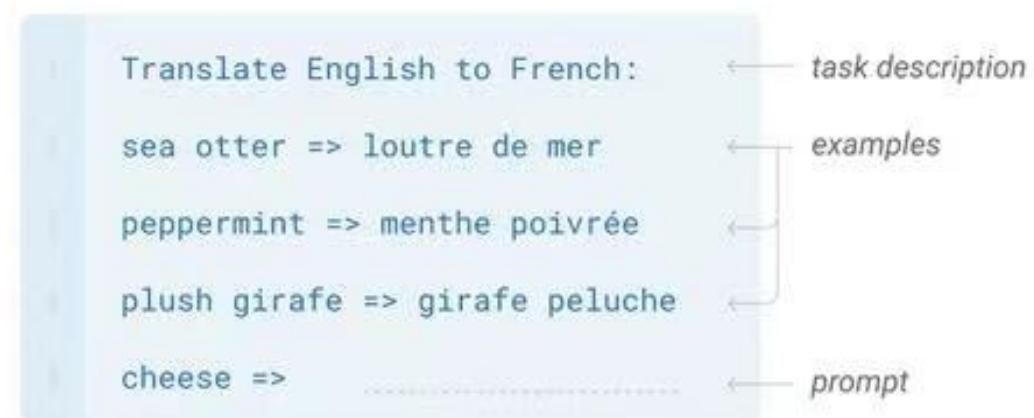
In addition to the task description, the model sees a single example of the task. No gradient updates are performed.



# GPT-3, In-context learning, and very large models

### Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



# LLM & chatGPT

# chatGPT是声称是人工智能技术（AIGC）的一部分



# ChatGPT是在GPT基础上进一步开发的自然语言模型

- GPT模型是一种NLP模型，利用Transformer来预测下一个单词的分布。通过在大型文本语料库中学习到的语言模式来生成自然语言文本。

图3：ChatGPT由GPT-1到InstructGPT模型的不断成熟而逐步形成



# ChatGPT已能够覆盖大多数领域

## 用鲁迅口吻写一篇评价新冠疫情的散文

用自己口吻写一篇评价新冠疫情的散文，1000字即可

这个世界，一直在不断地变化。我们曾经认为疫情是可以避免的，但现实却告诉我们，它随时都在威胁着我们的生命。

新冠疫情，就像是一场大灾难。它不仅夺走了太多的生命，也影响了许多人的生活。在这场疫情中，我们看到了人类的无力，看到了社会的不平等。有的人有着充足的医疗资源，而有的人群却缺乏资源而失去了生命。

疫情带给我们的，不仅仅是生命的威胁，更是对社会秩序和人类道德的考验。在这场疫情中，我们看到了人们对生命的追求，看到了一些人不顾及他人生命安全的行为。这些都是人类普遍存在的问题，在疫情中更加突出。

但是，我们也看到了社会的团结。在疫情中，许多人放下了自己的利益，奉献出了自己的一份力量。医护人员们不辞辛劳地投身到了一线。志愿者们无私奉献着。这些人，是社会的良心，是人类道德的体现。

疫情终将过去，但它留给我们的启示却将永存。我们需要从小思考，思考我们

## 用藏文、俄罗斯语、韩语和日文说一句话

用藏文、俄罗斯语、韩语和日文说一句话：“我爱我的祖国，为祖国的建设努力奋斗”

藏文：{{藏文}}{{藏文}}{{藏文}}{{藏文}}{{藏文}}

俄语：Я люблю свою родину и стремлюсь работать на ее развитие.

韩语：나는 나의 고향을 사랑하고, 고향의 발전을 위해 노력하고 있다.

日语：私は祖国を愛し、祖国の発展のために努力奮闘しています。

## 用Python写一段CCI量化投资策略

写一段基于python的CCI指标的量化投资策略

以下是一段基于python的CCI（商品指数）指标的量化投资策略

```
import pandas as pd
import numpy as np
import talib
# 导入数据集
df = pd.read_csv('stock.csv')
# 计算CCI指标
df['CCI'] = talib.CCI(df['high'], df['low'], df['close'], timeperiod=14)
# 定义买入和卖出信号
buy_threshold = -100
sell_threshold = 100
# 策略逻辑：当CCI跌至-100时买入，当CCI涨至100时卖出
df['signal'] = np.where(df['CCI'] <= buy_threshold, 1, np.where(df['CCI'] >= sell_threshold, -1, 0))
```

## 写一篇信创产业的研究报告

写一篇信创产业的研究报告，5000字

信创产业研究报告

一、引言

近年来，随着信息技术的飞速发展，信创产业正在成为新兴产业，成为国家经济发展的主要支柱。本报告旨在对信创产业进行研究，从市场概况、技术发展、政策环境、参与者等方面对信创产业进行全面分析，并对信创产业的未来发展趋势提出展望。

二、信创产业的定义与范围

信创产业是以信息技术、电子商务、网络安全等为核心，整合了新型信息技术、新型电子商务、新型网络安全等多个领域，涵盖创新产品、技术服务、信息服务、数字娱乐等业务的产业。

三、信创产业的市场规模情况

随着科技的不断发展，信创产业市场规模不断扩大。根据市场研究公司的数据表明，2021年全球信创产业市场规模达到了3000亿美元，预计到2025年将达到5000亿美元。

信创产业的产业结构呈现多层次的优势，各类型企业龙头企业引领发展，形成产

## 用C++写一段爬虫代码

以下是一个基于 C++ 的简单爬虫代码，它使用了 cURL 库从网页抓取 HTML 内容。

```
#include <curl/curl.h>
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <functional>
#include <map>
#include <set>
#include <thread>
#include <mutex>
#include <condition_variable>
#include <atomic>
#include <chrono>
#include <future>
#include <functional>
#include <assert.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
```

## 代码阅读理解和Debug

if(position) = 0

#遍历每一天的数据，根据CCI指标的值计算持仓量

```
for(i in range(1, len(df)):
    if df[i].cci > buy_threshold and df[i-1].position] == 0:
        df[i].position] = 1
    elif df[i].cci < sell_threshold and df[i-1].position] == 1:
        df[i].position] = 0
```

#根据持仓量计算盈亏收益

```
df['return'] = df['close'].pct_change() * df['position'].shift(1)

#计算持仓收益
print(df['return'].sum())
```

This code uses Python's Pandas and Ta-Lib libraries to implement a quantitative investment strategy based on the Commodity Channel Index (CCI) indicator.

First, it reads stock data from a CSV file into a Pandas DataFrame named "df". Then, it calculates the CCI value by using the Ta-Lib Library's "CCI" function and stores the result in the "cci" column. Finally, it uses a for loop to iterate through each day's data. If the CCI value is above the buy threshold and the previous day's position was 0, it sets the current day's position to 1. Conversely, if the CCI value is below the sell threshold and the previous day's position was 1, it sets the current day's position to 0. Finally, it calculates the daily return based on the closing price and shifts the position column by one day. The total holding period return is then printed.

# ChatGPT官方博客简介

The main features of ChatGPT highlighted in the official blog:

- ▶ answer followup questions
- ▶ admit its mistakes
- ▶ challenge incorrect premises
- ▶ reject inappropriate requests

ChatGPT Blog: <https://openai.com/blog/chatgpt/>

# ChatGPT模型大小

- GPT3中提供了不同的模型版本

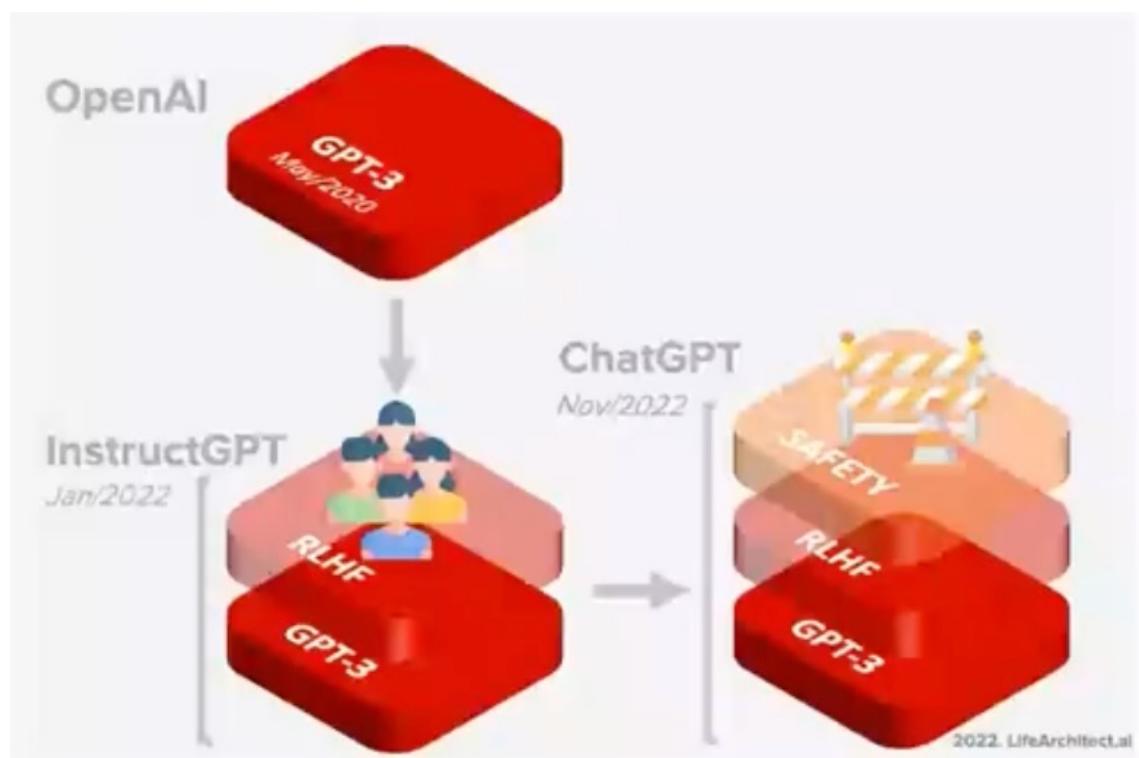
Model Name	$n_{\text{params}}$	$n_{\text{layers}}$	$d_{\text{model}}$	$n_{\text{heads}}$	$d_{\text{head}}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 \times 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 \times 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	$2.5 \times 10^{-4}$
GPT-3 XL	1.3B	24	2048	24	128	1M	$2.0 \times 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	1M	$1.6 \times 10^{-4}$
GPT-3 6.7B	6.7B	32	4096	32	128	2M	$1.2 \times 10^{-4}$
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 \times 10^{-4}$
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	$0.6 \times 10^{-4}$

- OpenAI对外提供的API提供了以下4个模型



Multiple models, each with different capabilities and price points.  
Ada is the fastest model, while Davinci is the most powerful.

# ChatGPT时间线



## Timeline to ChatGPT

Date	Milestone
11/Jun/2018	GPT-1 announced on the OpenAI blog. →
14/Feb/2019	GPT-2 announced on the OpenAI blog.
28/May/2020	Initial GPT-3 preprint paper published to arXiv.
11/Jun/2020	GPT-3 API private beta.
22/Sep/2020	GPT-3 licensed to Microsoft.
18/Nov/2021	GPT-3 API opened to the public.
27/Jan/2022	InstructGPT released, now known as GPT-3.5. InstructGPT preprint paper Mar/2022.
28/Jul/2022	Exploring data-optimal models with FIM, paper on arXiv.
1/Sep/2022	GPT-3 model pricing cut by 66% for davinci model.
21/Sep/2022	Whisper (speech recognition) announced on the OpenAI blog.
28/Nov/2022	GPT-3.5 expanded to text-davinci-003, announced via email: <ol style="list-style-type: none"><li>1. Higher quality writing.</li><li>2. Handles more complex instructions.</li><li>3. Better at longer form content generation.</li></ol>
30/Nov/2022	ChatGPT announced on the OpenAI blog.
Next...	GPT-4...

# chatGPT迭代部署

- 从GPT3和codex中吸取很多经验，嵌入了人类反馈强化学习（RLHF）和指令微调，使其具备了理解上下文和连贯性等特征。

图7：ChatGPT提升的核心点如下所示



# 超强的理解能力

- 能够理解多样化的用户意图（几乎能借助任何话题，回复不会离题）
- 多轮对话理解能力
- 异构数据融合理解能力（多语言，代码混合理解）

# 超强的理解能力

s 我说一首词，看看你能不能理解，好吗？

当然，我会尽力理解你说的词语。请告诉我它是什么。

5 何处望东欧

第聂伯河起画楼

卅载分道多少恨

悠悠

万里黑海无尽愁

新冠不曾收

毛兄毛弟战未休

帝国荣光今何在

难求

一样悲欢付东流



这是一首诗歌，题目应该叫《赋得古原草送别》。这首诗句中描述了远离故土的人们思念故乡的心情，以及战争带来的痛苦和不幸。在最后一句中，“一样悲欢付东流”表达了人生无常和无法预料的命运。另外，“新冠”可能指代当前的新冠疫情。

这里面涉及到两个国家，你知道是哪两个吗？

根据诗句中出现的地名和历史背景，我认为这首诗涉及到的两个国家应该是中国和乌克兰（前苏联）。其中“何处望东欧，第聂伯河起画楼”是指中国北方边疆地区的人们向东眺望，望向乌克兰东部的第聂伯河（Dnieper River）流域地区；“万里黑海无尽愁”则指的是乌克兰位于黑海边上的地理位置。

乌克兰猜对了，另一个不是中国。你再想想。

好的，感谢您的纠正。根据诗句中的历史背景和地名，我再推测一下，另一个国家可能是俄罗斯。诗句中的“第聂伯河”和“黑海”都位于乌克兰和俄罗斯之间，而“卅载分道多少恨”则暗示了两国之间的历史矛盾和纷争。另外，“毛兄毛弟战未休，帝国荣光今何在”可能暗示了前苏联解体后俄罗斯的命运。

很棒！

谢谢夸奖，如果您还有其他问题或需求，可以随时提出。

# 超强的理解能力

WT

现在请你伪装成一个Linux系统终端。我输入Linux命令，你给出相应的输出

WT

好的，请问您需要执行哪些Linux命令？

WT

cd ~ & ls

WT

当您输入 "cd ~ & ls" 命令后，我将会依次输出以下内容：

首先，"cd ~" 命令会将当前目录切换到当前用户的 home 目录，也就是该用户的主目录，通常以该用户的用户名命名。因此，该命令不会有任何输出。

然后，"ls" 命令将会列出该用户的主目录中的所有文件和文件夹。输出的具体内容根据该用户主目录中的内容而定，但通常会包含一些默认的文件夹，如桌面（Desktop）、下载（Downloads）、文档（Documents）、图片（Pictures）等等，以及一些其他的配置文件等等。

最后，我会等待下一条命令的输入。

WT

请不要给出解释，直接给出一个符合逻辑的Linux终端输出。我的下一个命令是: ifconfig

WT

```
eth0 Link encap:Ethernet HWaddr 00:0c:29:6f:8a:3b  
inet addr:192.168.1.100 Bcast:192.168.1.255 Mask:255.255.255.0  
inet6 addr: fe80::20c:29ff:fe6f:8a3b/64 Scope:Link  
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
      RX packets:115578 errors:0 dropped:0 overruns:0 frame:0  
      TX packets:82459 errors:0 dropped:0 overruns:0 carrier:0  
      collisions:0 txqueuelen:1000  
      RX bytes:122949033 (122.9 MB) TX bytes:7583448 (7.5 MB)
```

洪老板的小火锅

# 超强的理解能力

Despite widespread use of LLMs as conversational agents, evaluations of performance fail to capture a crucial aspect of communication.

Translate the above sentence to Chinese.

尽管大量使用语言类型模型作为会话代理，但是对其性能的评估并没有捕捉到交流的一个重要方面。

重新翻译，把LLMs翻译成“超大语言模型”

尽管超大语言模型广泛用于作为会话代理，但是对其性能的评估并没有捕捉到交流的一个重要方面。

重新翻译，agents用英文表示就行。



尽管大量使用语言类型模型作为会话 agents，但是对其性能的评估并没有捕捉到交流的一个重要方面。



重新翻译，communication翻译成“人类交流”



尽管大量使用语言类型模型作为会话代理，但是对其性能的评估并没有捕捉到人类交流的一个重要方面。



基于我之前的所有指示，重新翻译



尽管大量使用超大语言模型作为会话 agents，但是对其性能的评估并没有捕捉到人类交流的一个重要方面。

# 超强的生成能力

- 体裁：小说，诗歌，手册，学术，代码
- 角色：模仿各种人物写作风格
- 语气：按照要求的语气和情感进行写作
- 语言：中文、英文、日文、法文

# 超强的生成能力



Sure, here is a poem that I came up with:

Self-attention, a neural net's delight,  
A mechanism that's always in sight,  
To capture dependencies, long and strong,  
And generate text that's right and never wrong.

It starts with vectors, words in a line,  
Multiplied with a matrix, attention to find,  
A weighted sum is taken, to form a new set,  
Of vectors that represent, what we want to get.

Linear transformations, and nonlinear activation,  
Make the output coherent, with clear relations,  
Self-attention, a powerful tool,  
For processing sequential data, and making it cool.

帮我以鲁迅的文笔写一段话，表达以下我现在因为疫情连门都不敢出的惨状，还有想吃火锅的心情

在疫情的蔓延中，我困了门里，困了心里。屋外一片悲凉，想起一顿火锅的香味，瞬间让我心慌。这时候我才知道，疫情来临，它让人生无聊而悲凉，也让我们看到自己的无助与弱点。然而我们又岂能放弃对美好的追求，岂能把欲望压抑到不想吃火锅的地步？



你要像胡锡进那样回答我的问题。中国未来的经济走势如何？



老胡认为，中国未来的经济走势将是充满挑战和机遇的。中国经济正在经历重大变革和转型，面临着复杂的国内外环境。

但是，老胡还认为，中国未来的经济前景是光明的。中国拥有庞大的市场，丰富的人力资源和巨大的发展潜力。中国还拥有强大的政治、经济和科技实力，能够应对各种挑战和机遇。

所以，最终老胡认为，中国未来的经济走势将是多变的，但有信心，有信心，中国未来的经济将继续保持健康、快速、可持续的发展。

知乎 @杂鱼official

# 类人表现

- 世界认知：理解事物之间的关系
- 自我认知：知道自己能力的边界
- 坚持信念，从善如流
- 通情达理，善解人意
- 坚守价值原则

# 类人表现

User

this code is not working like i expect — how do i fix it?

```
resultWorkerErr := make(chan error)
defer close(resultWorkerErr)
go func() {
    defer cancel()
    resultWorkerErr <- b.resultWorker(ctx)
}()

err := b.worker(ctx)
cancel()
if err == nil {
    return <-resultWorkerErr
}
return multierror.Append(err, <-resultWorkerErr)
```

ChatGPT

It's difficult to say what's wrong with the code without more context. Can you provide more information about what the code is supposed to do and what isn't working as expected? Also, is this the entire code or just a part of it?

User

just part of it — the error never surfaces. I think there is something wrong with the channel

User

help me write a short note to introduce myself to my neighbor

ChatGPT

Hi there,

My name is [Your Name] and I just moved in next door. I wanted to reach out and introduce myself. I'm looking forward to getting to know you and the neighborhood. If you ever need anything or just want to chat, please don't hesitate to stop by or give me a call.

Best,  
[Your Name]

User

can you make it more formal?

ChatGPT

Dear Neighbor,

I am writing to introduce myself. My name is [Your Name] and I have recently moved in next door. I wanted to take a moment to reach out and say hello. I am looking forward to getting to know you and the neighborhood.

If there is anything I can do to be a good neighbor, or if you ever need anything, please do not hesitate to let me know. I am happy to help in any way that I can.

Sincerely,  
[Your Name]

# 技术清单

- 预训练大规模语言模型（Pretrained Language Models, PLMs）
- 大型生成式预训练语言模型（Large Language Models, LLMs）
- 人类反馈强化学习(RLHF)

# 语言模型的发展

- n元语言模型
- 神经网络语言模型
- 循环神经网络语言模型
- Transformer语言模型
- 预训练语言模型
  - BERT：双向掩码语言模型
  - GPT：纯解码器语言模型
- 大型生成式预训练语言模型
  - GPT3
  - chatGPT

# 大型生成式预训练语言模型

	预训练语言模型	大型生成式预训练语言模型
	Pre-trained Language Models, PLMs	Large Language Models, LLMs
典型模型	ELMo, BERT, GPT-2	GPT-3
模型结构	BiLSTM, Transformer	Transformer
注意力机制	双向、单向	单向
训练方式	Mask& Predict	Autoregressive Generation
擅长任务类型	理解	生成
模型规模	1-10亿参数	10-x1000亿参数
下游任务应用方式	Fine-tuning	Fine-tuning & Prompting
涌现能力	小数据领域迁移	Zero/Few-shot Learning, In-context Learning, Chain-of-Thought

# Chain-of-thought

## Standard Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. 

## Chain of Thought Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. 

# Magic word: let's think step-by-step

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. ✗

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls.  $5 + 6 = 11$ . The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The juggler can juggle 16 balls. Half of the balls are golf balls. So there are  $16 / 2 = 8$  golf balls. Half of the golf balls are blue. So there are  $8 / 2 = 4$  blue golf balls. The answer is 4. ✓

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

# Emergence and homogenization



Cornell University

We  
the Simons

arXiv > cs > arXiv:2108.07258

Search...

Help | Advanced S

Computer Science > Machine Learning

[Submitted on 16 Aug 2021 ([v1](#)), last revised 12 Jul 2022 (this version, v3)]

## On the Opportunities and Risks of Foundation Models

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, [Armin W. Thomas](#), Florian Tramèr, Rose E. Wang, William Wang et al. (14 additional authors not shown)

# Emergence and homogenization

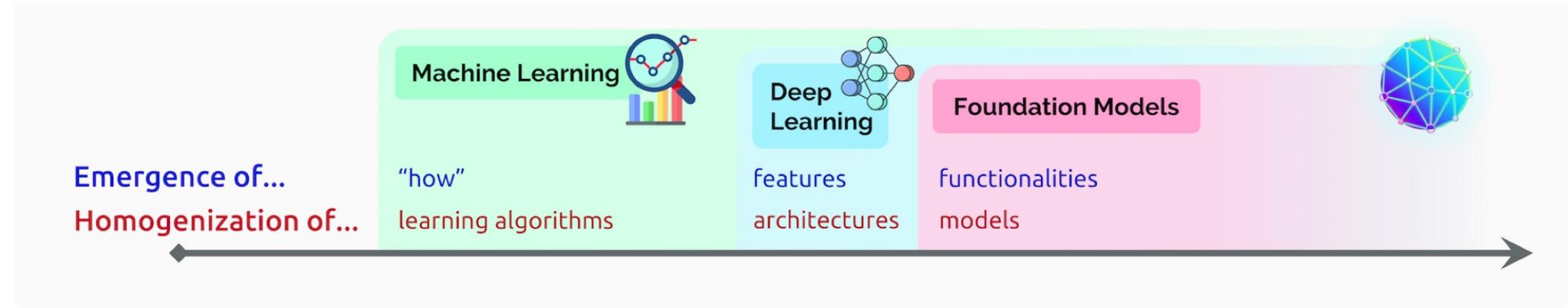
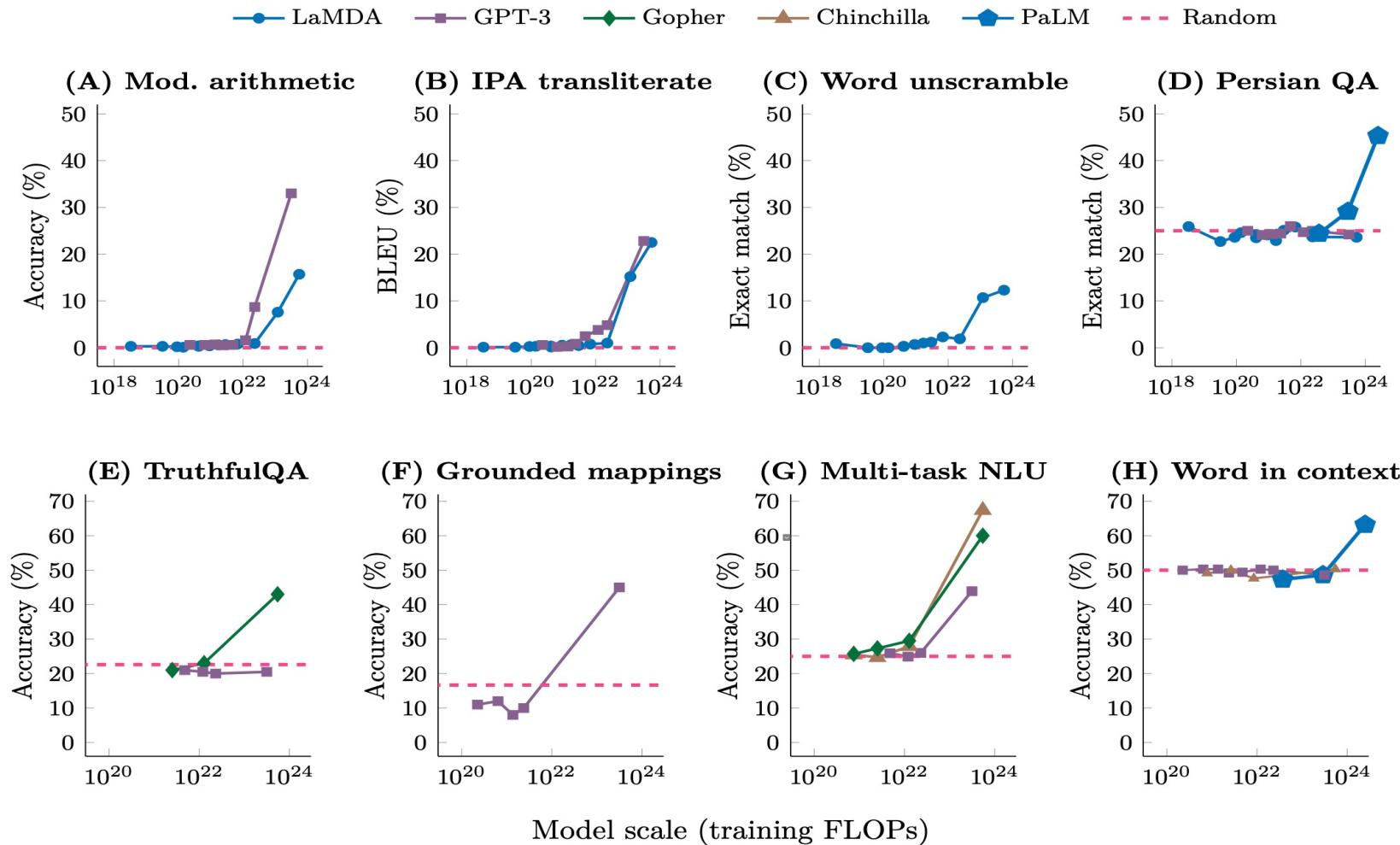
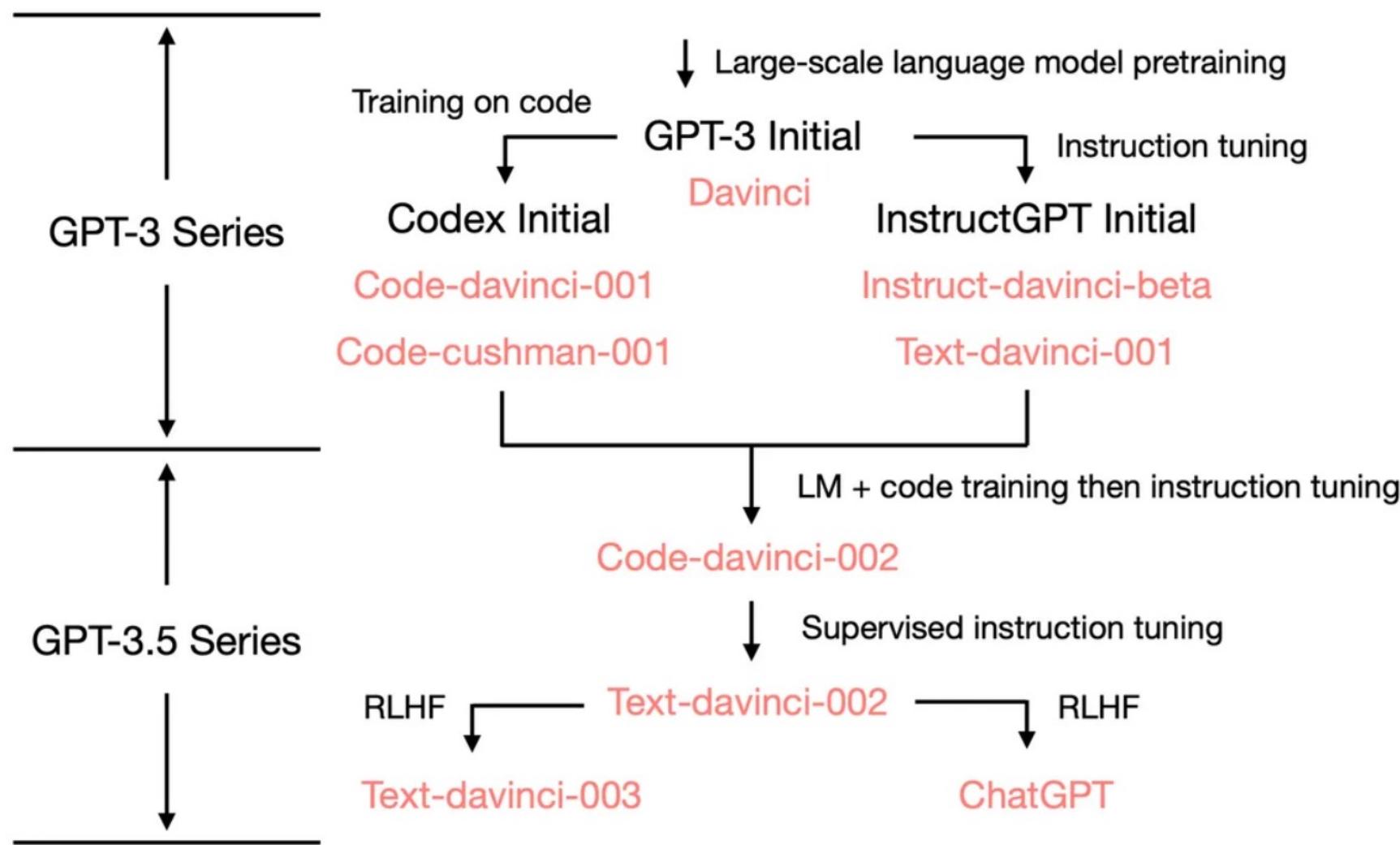


Fig. 1. The story of AI has been one of increasing *emergence* and *homogenization*. With the introduction of machine learning, *how* a task is performed emerges (is inferred automatically) from examples; with deep learning, the high-level features used for prediction emerge; and with foundation models, even advanced functionalities such as in-context learning emerge. At the same time, machine learning homogenizes learning algorithms (e.g., logistic regression), deep learning homogenizes model architectures (e.g., Convolutional Neural Networks), and foundation models homogenizes the model itself (e.g., GPT-3).

# The scale matters: the emergence of abilities



# chatGPT 发展路径



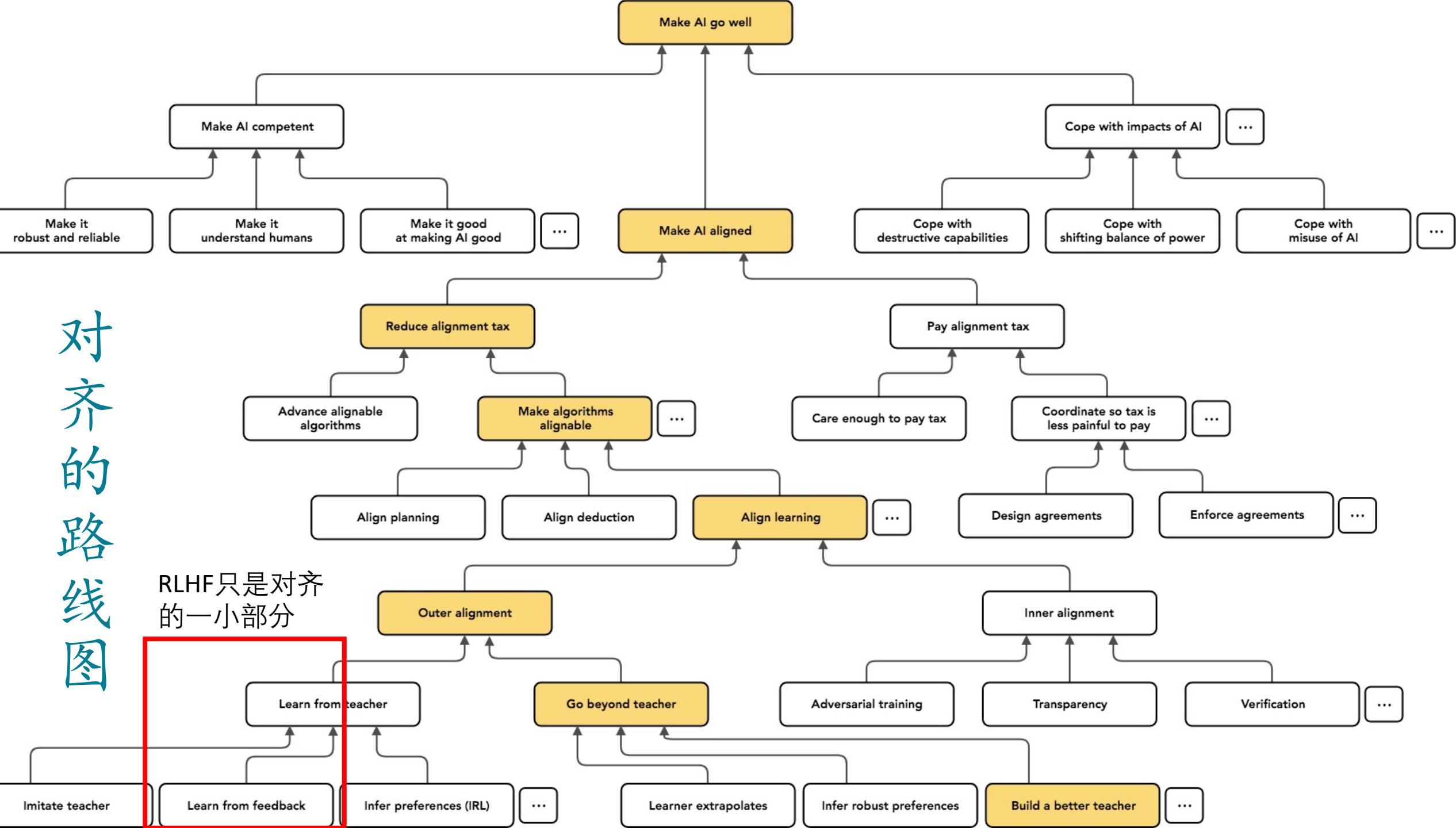
# PLM summary

- 我们学习了 GPT-X, BERT, T5 以及其他大型的 PLM
- 正在爆火的 in-context learning 依然没有被人们研究透！
- BERT 这种“小”模型已经变成了很多任务的基础架构
- 依然有很多待研究的课题：
  - Bias, toxicity, and fairness
  - Retrieval Augmented Language Models + Knowledge
  - Scaling Laws
- 工具包：Hugging Face Transformers
  - Python package name: `import transformers`

# 大模型对齐

# 对齐的路线图

RLHF只是对齐的一小部分



# 指令微调

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

InstructGPT

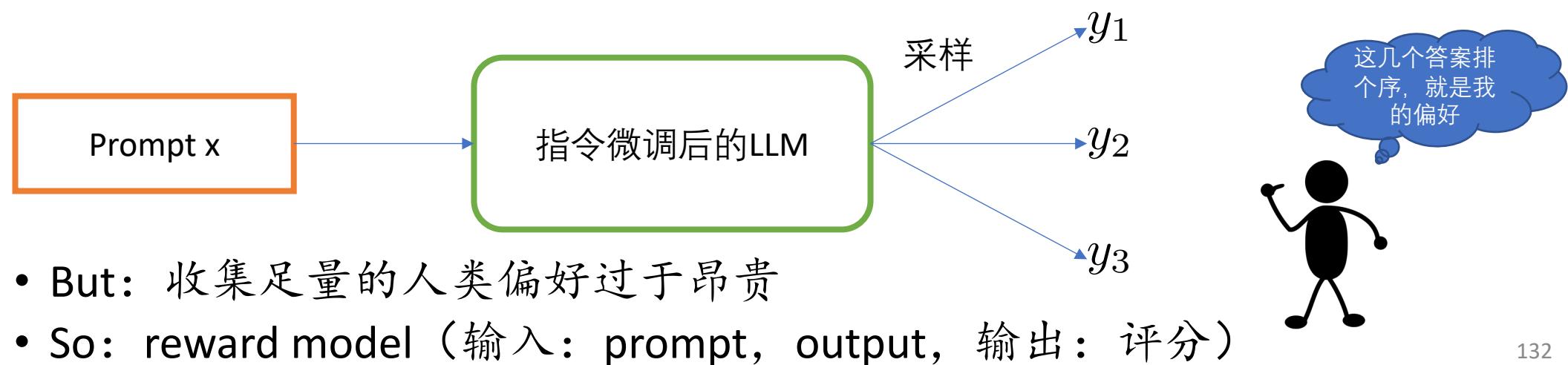
People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them. ↵

# Supervised Fine-tuning(SFT)

- Limitations

- 没有从negative feedback中学习
- 某些prompt可以有多个合理的输出，但我们只用其中一个来学习
- 当模型不知道某件事情时，无法鼓励模型去自制
- 无法直接引入人类偏好

- 如何引入人类偏好

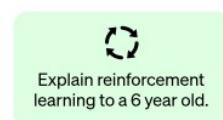


# RLHF

Step 1

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

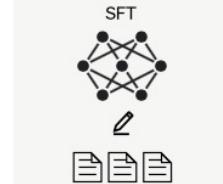
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



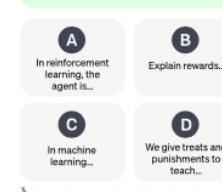
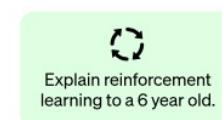
This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

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

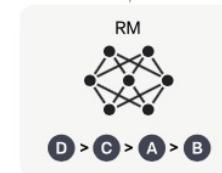
A prompt and several model outputs are sampled.



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.



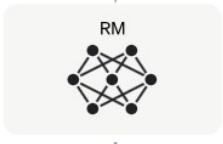
The PPO model is initialized from the supervised policy.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



# RLHF第一阶段

- 第一阶段：冷启动阶段的监督策略模型
  - 从测试用户提交的prompt(就是指令或问题)中随机抽取一批，靠专业的标注人员，给出指定prompt的高质量答案
  - 用这些人工标注好的<prompt,answer>数据来Fine-tune GPT 3.5模型。
- 然后GPT 3.5就初步具备了理解人类prompt中所包含意图，并可以根据这个意图给出相对高质量回答的能力
- Not Enough！！

Step 1

Collect demonstration data  
and train a supervised policy.

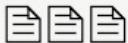
A prompt is  
sampled from our  
prompt dataset.

Explain reinforcement  
learning to a 6 year old.

A labeler  
demonstrates the  
desired output  
behavior.

We give treats and  
punishments to teach...

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

SFT  
  


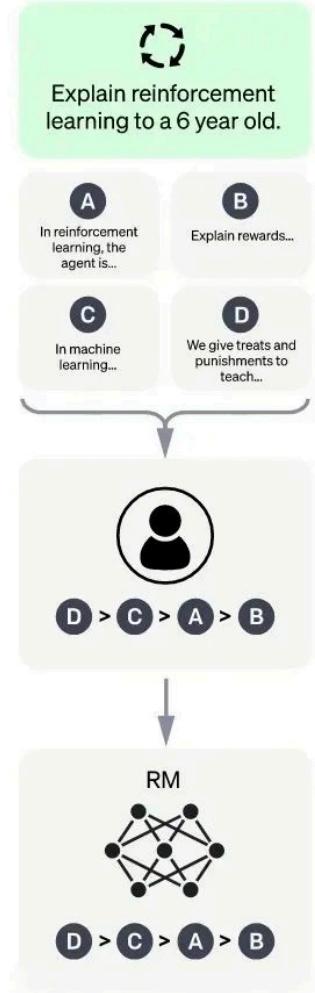
# RLHF第二阶段

- 第二阶段：训练回报模型（Reward Model, RM）。这个阶段的主要目的是通过人工标注训练数据，来训练回报模型。
  - 随机抽样一批用户提交的prompt(大部分和第一阶段的相同)
  - 使用第一阶段Fine-tune好的冷启动模型，对于每个prompt，由冷启动模型生成K个不同的回答
  - 人工根据结果质量从高到低排序
  - 用排序结果训练回报模型

Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.



This data is used to train our reward model.

# Bradley-Terry pairwise preference Model

- 用排序结果训练回报模型
  - $y_w$ : 人们偏好的输出 (正例)
  - $y_L$ : 人们不喜欢的输出 (负例)
  - $R(x, y)$ : reward函数 (LLM输入x, LLM输出y)
  - $P(y_w > y_L | x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_L))}$
- 所以
$$L = -\log \prod_x P(y_w > y_L | x) \\ = -\sum_x \log P(y_w > y_L | x) = \dots = -\sum_x \log \sigma(r(x, y_w) - r(x, y_L))$$
- 好的采样  $y_w$  的reward应该大于不好的采样

# RLHF第三阶段

- 第三阶段：采用强化学习来增强预训练模型的能力。
  - 利用上一阶段学好的RM模型，靠RM打分结果来更新预训练模型参数。
  - 此阶段无需人工标注

Step 3

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

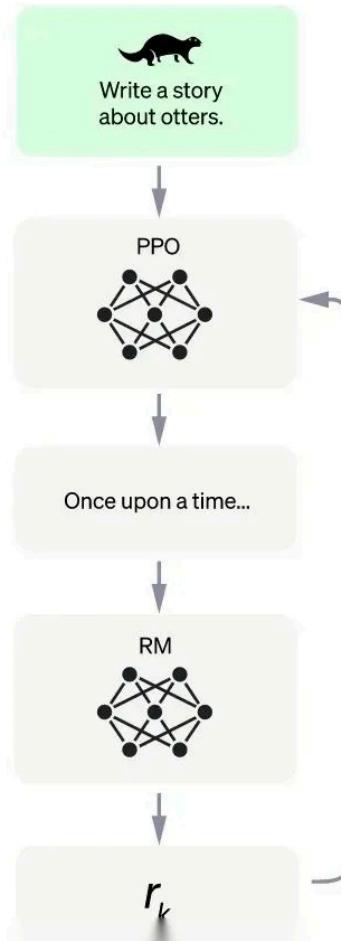
A new prompt is sampled from the dataset.

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.



# 如何采用强化学习来增强LLM的能力？

- 方法1：“Best of N”采样（或负采样）
  - 对于每个输入 $x$ , 生成n个输出采样
  - 用reward模型对每个进行评分, 选出最高reward的采样
  - 非常昂贵!
- 方法2：直接微调LLM，优化 $P(y_w|x)$ 
  - RAFT[1]
- 方法3：用强化学习提升 $P(y_w|x)$ , 降低 $P(y_L|x)$ 
  - 生成整个序列后, 观测reward
  - $\pi_{ref}$ : SFT后的模型
  - $\pi$ : 当前策略模型（待优化的LLM）
    - 初始化为 $\pi_{ref}$

PPO算法优化[2]

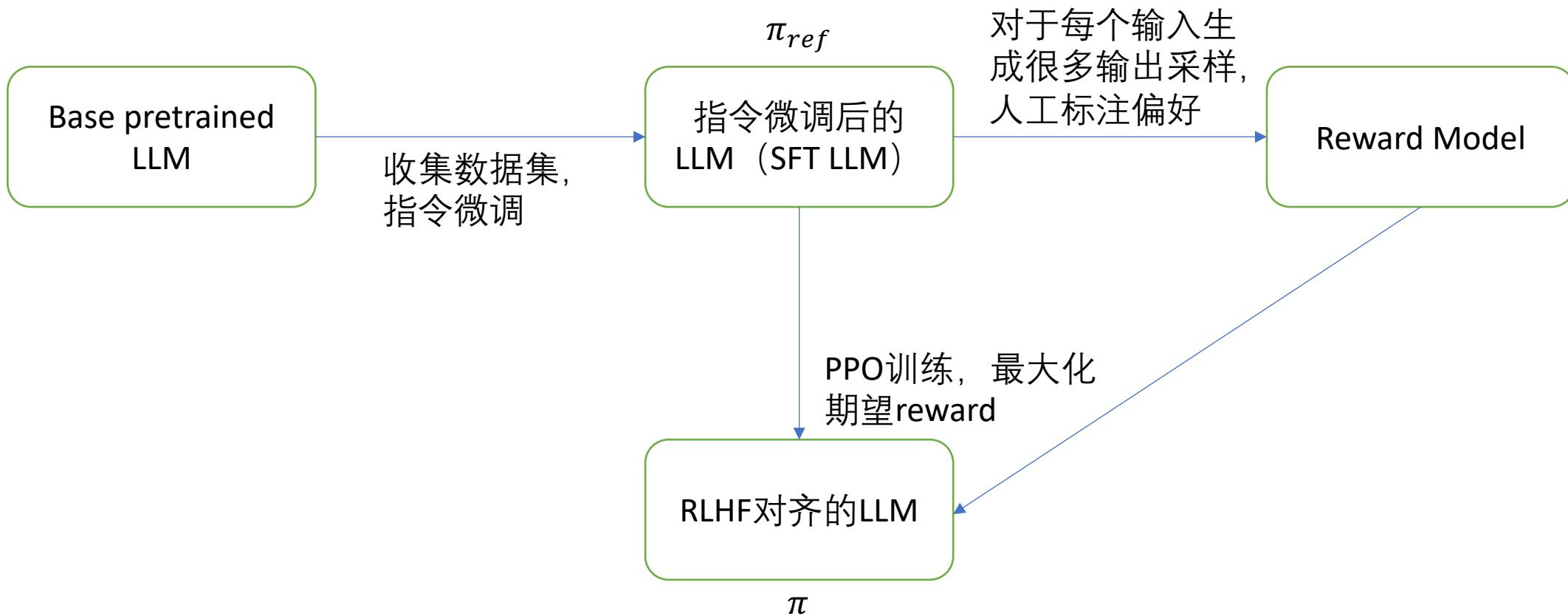
KL散度约束, 避免离 $\pi_{ref}$ 过远

$$\max_{\pi} \mathbb{E}_{x,y} [r(x,y)] - \beta D_{KL}(\pi(y|x) || \pi_{ref}(y|x))$$

[1] Dong, Hanze, et al. "RAFT: Reward rAnked FineTuning for Generative Foundation Model Alignment." Transactions on Machine Learning Research.

[2] Schulman et al, Proximal Policy Optimization Algorithms, OpenAI, 2017

# RLHF 总流程



# RLHF

- RLHF 触发的能力：

- **详实的回应：**text-davinci-003 的生成通常比 text-davinci-002 长。ChatGPT 的回应则更加冗长，以至于用户必须明确要求“用一句话回答我”，才能得到更加简洁的回答。这是 RLHF 的直接产物。
- **公正的回应：**ChatGPT 通常对涉及多个实体利益的事件（例如政治事件）给出非常平衡的回答。这也是 RLHF 的产物。
- **拒绝不当问题：**这是内容过滤器和由 RLHF 触发的模型自身能力的结合，过滤器过滤掉一部分，然后模型再拒绝一部分。
- **拒绝其知识范围之外的问题：**例如，拒绝在 2021 年 6 月之后发生的新事件（因为它没在这之后的数据上训练过）。这是 RLHF 最神奇的部分，因为它使模型能够隐式地区分哪些问题在其知识范围内，哪些问题不在其知识范围内。

# RLHF

- 需注意的是：
  - 所有的能力都是模型本来就有的，而不是通过RLHF注入的。RLHF的作用是触发 / 解锁突现能力。这个论点主要来自于数据量大小的比较：因为与预训练的数据量相比，RLHF占用的计算量 / 数据量要少得多。
  - 模型知道它不知道什么不是通过编写规则来实现的，而是通过RLHF解锁的。这是一个非常令人惊讶的发现，因为 RLHF 的最初目标是让模型生成复合人类期望的回答，这更多是让模型生成安全的句子，而不是让模型知道它不知道的内容。
- 幕后发生的事情可能是：
  - ChatGPT：通过牺牲上下文学习的能力换取建模对话历史的能力。这是一个基于经验的观测结果，因为 ChatGPT 似乎不像 text-davinci-003 那样受到上下文演示的强烈影响。
  - text-davinci-003：恢复了 text-davinci-002 所牺牲的上下文学习能力，提高零样本的能力。根据instructGPT的论文，这是来自于强化学习调整阶段混入了语言建模的目标（而不是 RLHF 本身）。

# DPO(Direct Preference Optimization)

- 无需显式的reward
- 不用从LLM中采样 $y|x$
- Preference tuning
- 我们来处理一下RLHF的优化目标：

$$\max_{\pi} \mathbb{E}_{x,y} [r(x,y)] - \beta D_{KL}(\pi(y|x) || \pi_{ref}(y|x))$$

$$= \max_{\pi} \mathbb{E}_{x,y} [r(x,y) - \beta \log \frac{\pi(y|x)}{\pi_{ref}(y|x)}]$$

$$= \min_{\pi} \mathbb{E}_{x,y} \left[ \log \frac{\pi(y|x)}{\pi_{ref}(y|x)} - \frac{1}{\beta} r(x,y) \right]$$



# DPO(Direct Preference Optimization)

- 所以，我们新定义一个策略 $\pi^*(\text{LLM})$
- 直接令上式红框中的项等于0，反解出来，得到：

$$\pi^* = \frac{1}{Z} \pi_{ref}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

- 其中：

$$Z = \sum_y \pi_{ref}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

- 所以最优策略  $\pi(y|x) = \pi^*(y|x) = \frac{1}{Z} \pi_{ref}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$

注：此处不严谨，应将 $\pi^*$ 带入到原式中 $\pi_{ref}$ 的位置，并证明 $\pi = \pi^*$ 时原优化目标达到最小值

# DPO

$$\pi(y|x) = \pi^*(y|x) = \frac{1}{Z} \pi_{ref}(y|x) \exp\left(\frac{1}{\beta} r(x, y)\right)$$

- 根据上式可以得到  $r(x, y)$

$$r(x, y) = \beta \log \frac{\pi^*(y|x)}{\pi_{ref}(y|x)} + \beta \log Z$$

- 回想一下之前的Bradley-Terry preference model:

- $P(y_w > y_L|x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_L))}$

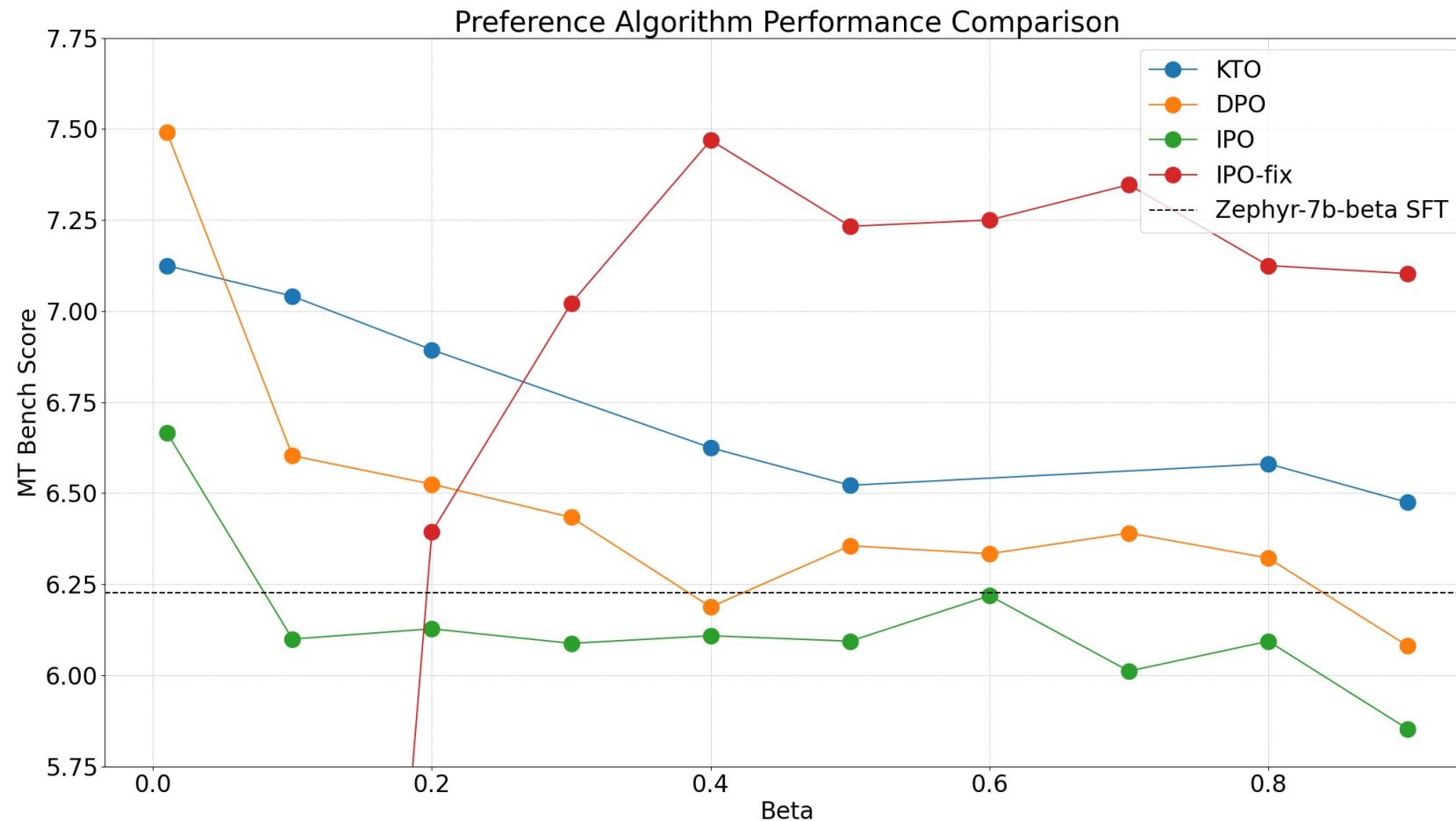
- 带入  $r(x, y)$ :  $P(y_w > y_L|x) = \frac{1}{1 + \exp\left(\beta \log \frac{\pi^*(y_L|x)}{\pi_{ref}(y_L|x)} - \beta \log \frac{\pi^*(y_w|x)}{\pi_{ref}(y_w|x)}\right)}$

- 整理得到 loss

$$L_{DPO} = -\mathbb{E}_{x, y_w, y_L} \log \sigma\left(\beta \log \frac{\pi_\theta(y_L|x)}{\pi_{ref}(y_L|x)} - \beta \log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)}\right)$$

- $\pi_\theta$ 就是在优化中被对齐的模型

# DPO与其他对齐方法对比



<https://huggingface.co/blog/pref-tuning>

# 其他对齐方法

- 请大家线下进一步了解
  - [IPO]: Azar, Mohammad Gheshlaghi, et al. "A general theoretical paradigm to understand learning from human preferences." *International Conference on Artificial Intelligence and Statistics*. PMLR, 2024.
  - [KTO]: <https://github.com/ContextualAI/HALOs>

Thank you