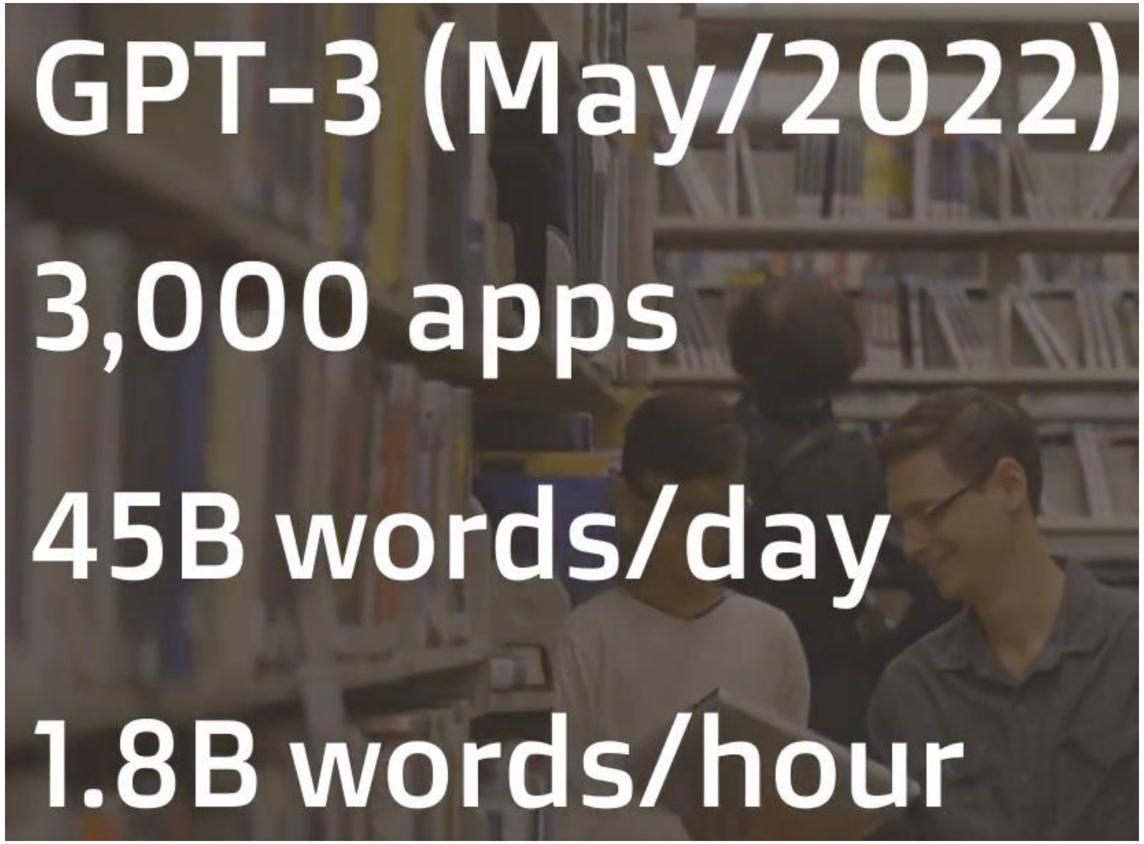
OPENAI

- 他们在搞什么东西
  - 预计每天产生450亿词
  - 貌似每小时生成100W本书
  - 以后我们所看,所读,所想还是真的吗
  - 这还仅仅是2022年的GPT-3

    (2022年每天生成的词量是2021年的10倍)



OPENAI

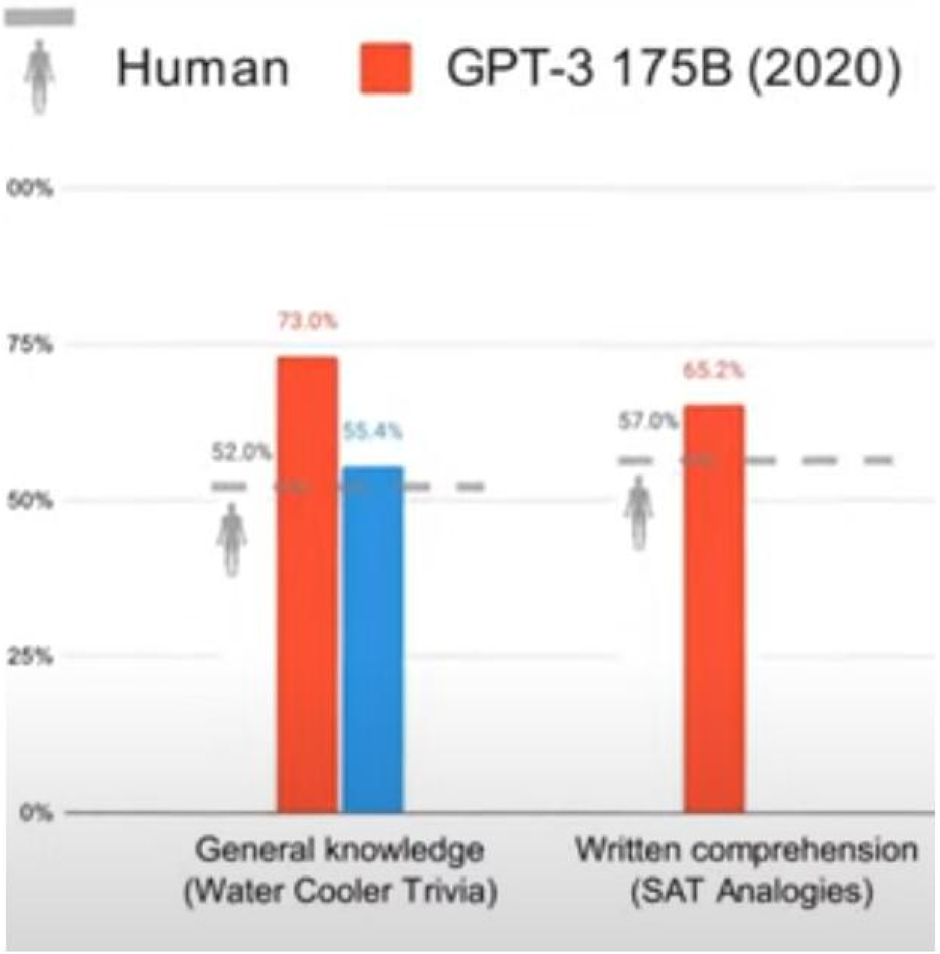
- 这哥们是真行，有微软爸爸在，啥都不是事

  - Openai老窝在爱荷华州，微软投资的数据中心

OPENAI

GPT VS Human

- GPT-3已经比人聪明了？那以后不得造反
- 这也会带来一些困扰和问题，偏见
- 语言模型在学咱们，但是分不清好赖话
- 斯坦福2022AI指数已经指出NLP偏见很大



OPENAI

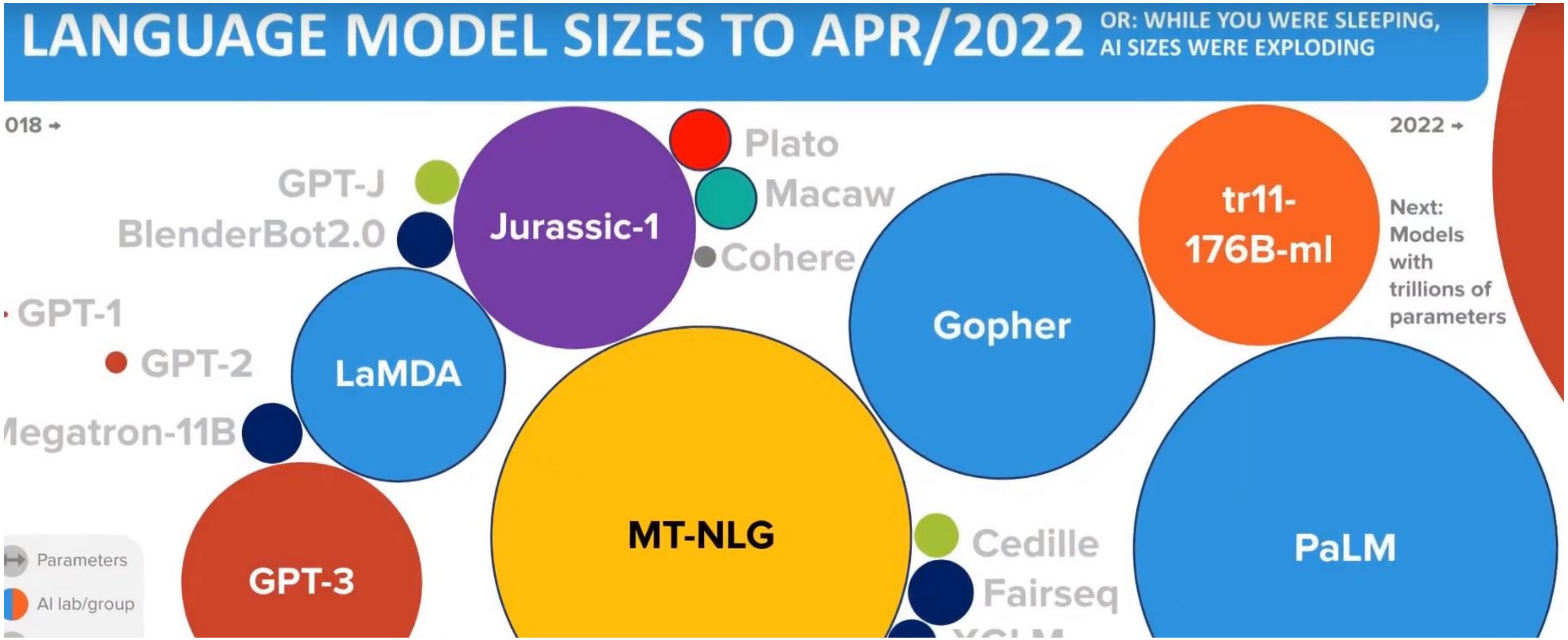
- 万物皆可GPT

  - 啥玩应？咱们要失业了？

OPENAI

- 但是世界不仅仅是GPT

  - GPT其实也只是冰山一角，2022年每4天就有一个大型模型问世



OPENAI

- 你可能会好奇，家里啥条件能训练这模型

  - 训练这种级别的语言模型，真是可远观而不可亵玩焉

  - 可以想象得到，光电费咱们可能都交不起

  - 但这仅仅是GPT-3，现在NLP起步于此，GPT-4相信很快就会面世

The supercomputer developed for OpenAI is a single system with more than 285,000 CPU cores, 10,000 GPUs and 400 gigabits per second of network connectivity for each GPU server. Compared with other machines listed on the TOP500 supercomputers in the world, it ranks in the top five, Microsoft says. Hosted in Azure, the supercomputer also benefits fr the capabilities of a robust modern cloud infrastructure, including rapid deployment, sustainable datacenters and access Azure services.

GPT

- 历史时刻

  - 2018年6月 GPT-1：约5GB文本，1.17亿参数量

  - 2019年2月 GPT-2：约40GB文本，15亿参数量

  - 2020年5月 GPT-3：约45TB文本，1750亿参数量

  - 传闻GPT-3电费1200万刀，72页论文（论文干货没啥。。。）

GPT-1

- 带你回到2018年的抖音（不对是2018年的NLP）

  - GPT 是"Generative Pre-Training"的简称，生成式的预训练
  - 2018年NLP可谓神仙打架，BERT与GPT不分先后，这俩联手估计就一统江湖了
  - BERT和GPT谁更难训练呢？肯定是GPT，它要下一盘大棋
  - 完型填空（BERT已经上下文）；预测未来（GPT预测以后的事）

GPT-1

- 损失函数就是预测下一个词，整体架构就是transformer解码器

3.1 Unsupervised pre-training

Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta) \tag{1}$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ. These parameters are trained using stochastic gradient descent [51].

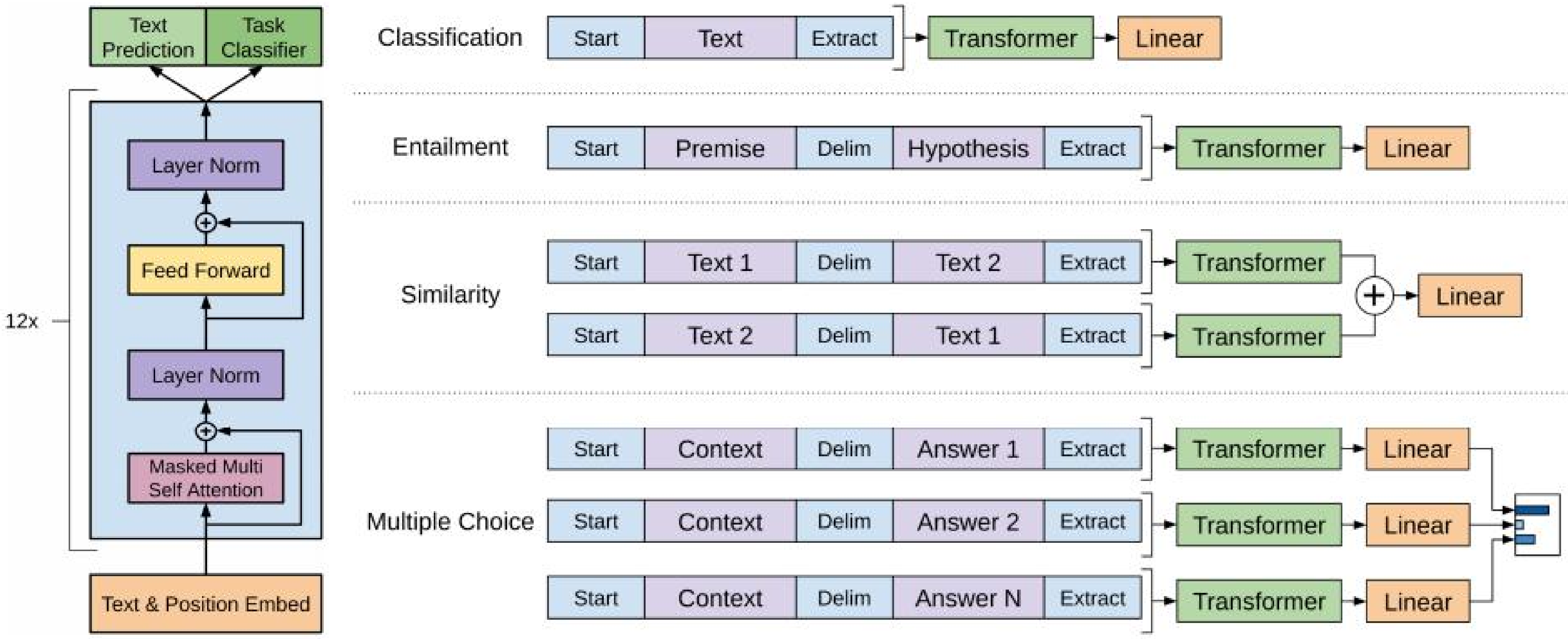
In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned} \tag{2}$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

GPT-1

所有下游任务都需要微调（再训练）



GPT-2

- 以不变应万变

- zero-shot在这开始要起来了，下游任务我干脆都不训练不微调了

- 下游任务有好多种，不训练怎么能让模型知道你要干啥呢?

- 你暗示他啊，通过一些提示告诉模型需要完成什么任务

- 总结来说就是更大了，而且下游任务不需要微调

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

GPT-2

- 采样策略相关

  - 自回归模型要进行预测，但是会不会陷入一个死循环呢？
  - 成语接龙：一一得一，一一得一，一一得一，一一得一，一一得一
  - 所以我们得希望模型有点多样性，就像写作文似的，不能光用然后
  - 我今天吃饭了，然后打游戏，然后在吃饭，然后打篮球，然后再打游戏

GPT-2

Temperature

- 温度就是说对预测结果进行概率重新设计
- 默认温度为1就相当于还是softmax
- 温度越高相当于多样性越丰富（雨露均沾）
- 温度越低相当于越希望得到最准的那个

```
>>> import torch
>>> import torch.nn.functional as F
>>> a = torch.tensor([1,2,3,4.])
>>> F.softmax(a, dim=0)
tensor([0.0321, 0.0871, 0.2369, 0.6439])
>>> F.softmax(a/.5, dim=0)
tensor([0.0021, 0.0158, 0.1171, 0.8650])
>>> F.softmax(a/1.5, dim=0)
tensor([0.0708, 0.1378, 0.2685, 0.5229])
>>> F.softmax(a/1e-6, dim=0)
tensor([0., 0., 0., 1.])
```

GPT-2

- Top k与Top p
  - 模型在采样的时候能不能采样到贼离谱的结果呢（没准啊）
  - 所以TOPK和TOPP都是要剔除掉那些特别离谱的结果
  - TOPK比如概率排序后选前10个，那之后的值就全部为0了
  - TOPP就跟那个CUMSUM似的算累加，一般累加到0.9或者0.95

GPT-3

- 不做微调，再说一遍不做微调
  - 不用你说，你让我微调我也没那个条件啊
  - 2020的时候人家老总说我们不开源是对人类好，为你们负责。。。
  - 虽然没提供源码，但是提供了付费API来微调
  - 其实中文模型也有很多，百度文心大模型应该也能媲美一下

GPT-3

- 咱们面向百度编程，它面向人类编程

就是说GPT-3训练的数据包罗万象，上通天文下知地理

GPT-3

- 3种核心的下游任务方式
  - 其实就是输入例子有几个，打个样

Few-shot

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

1	`Translate English to French:`	*task description*
2	`sea otter => loutre de mer`	*examples*
3	`peppermint => menthe poivrée`	
4	`plush girafe => girafe peluche`	
5	`cheese =>`	*prompt*

Zero-shot

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

1	`Translate English to French:`	*task description*
2	`cheese =>`	*prompt*

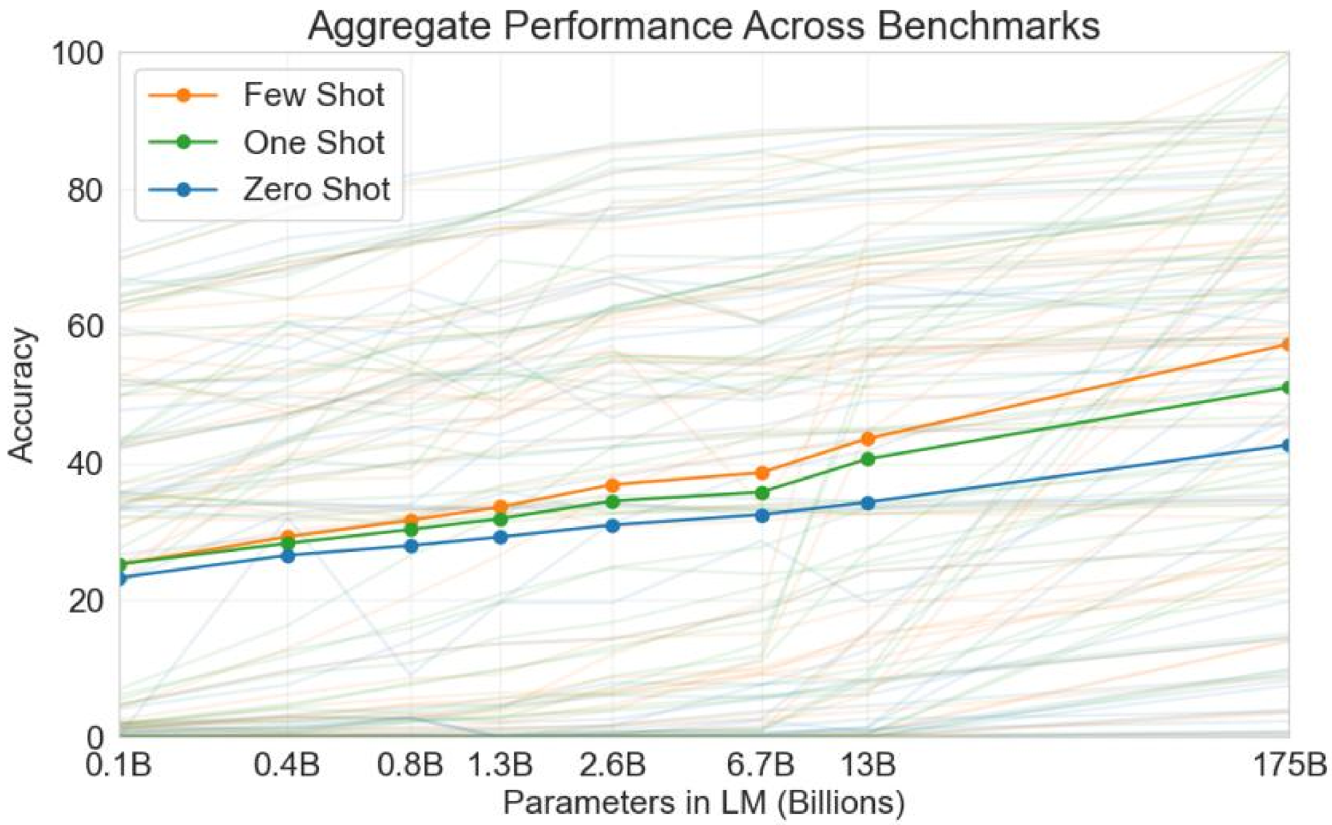
One-shot

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

1	`Translate English to French:`	*task description*
2	`sea otter => loutre de mer`	*example*
3	`cheese =>`	*prompt*

GPT-3

- 3种方式的对比
  - 这三种都没有更新模型
  - 肯定few的效果好一些
  - 但是问题就是API更贵了
  - 输入序列长度更长了



GPT-3

网络结构

网络结构没啥特别的，但是3.2M的batch有点辣眼睛

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

GPT-3

- 准备数据的事

  - 数据集得大还得干净才行啊，需要做的工作还挺多
  - 质量判断，对爬取的网页，进行分类任务看其质量OK不
  - 对网页进行筛选，剔除掉一些重要性低的(这些算法设计起来也不容易)
  - 也包括了前几代版本的训练数据，整合一块后开始训练

CODEX

- Evaluating Large Language Models Trained on Code

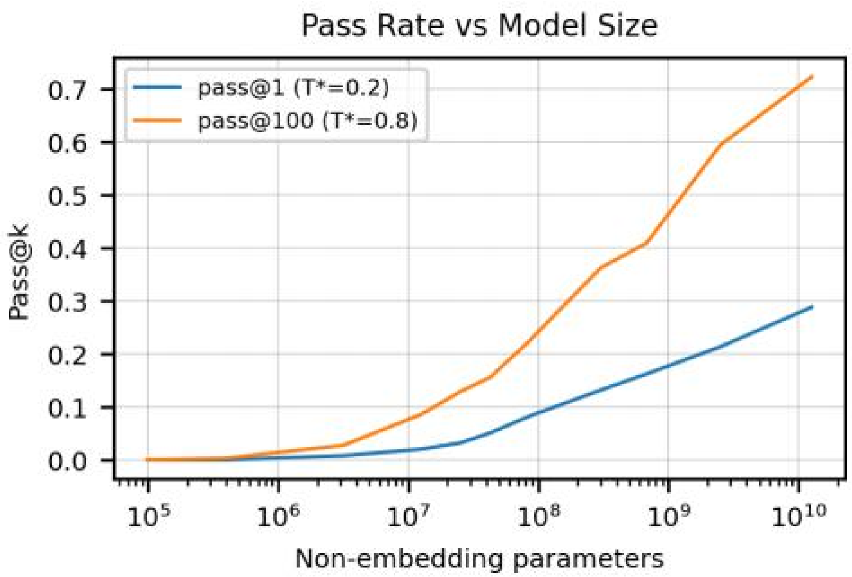
  - 用GPT-3模型重新训练（注意不是微调）

  - 我总说面向GITHUB编程，GPT-3这回真把这个事干了

```python
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]
```

```python
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```



CODEX

- Evaluating Large Language Models Trained on Code

  - 一言难尽，直接看DEMO吧：https://openai.com/blog/openai-codex/#spacegame
  - 训练数据就是GITHUB，相当于把文档注释和代码结合到一起
  - 输入注释或者文档，来预测代码如何实现，要面向CODEX编程了？
  - 其实在告诉我们一件事，GPT可以个性化设置（如何能吵赢我媳妇呢？）

ChatGPT

- 为什么来的猝不及防

  - Dalle2搞饥饿营销，给自己饿死了。。。（stable diffusion）

  - 大模型都在做，就看谁快了（这回没搞那个排队估计也是怕别人抢风头）

  - 但是不得不说，看见了NLP的未来（取代搜索引擎究竟何时）

  - 还有个但是，chatGPT还没公布论文，接下来的故事（主要我来编，你来信）

ChatGPT

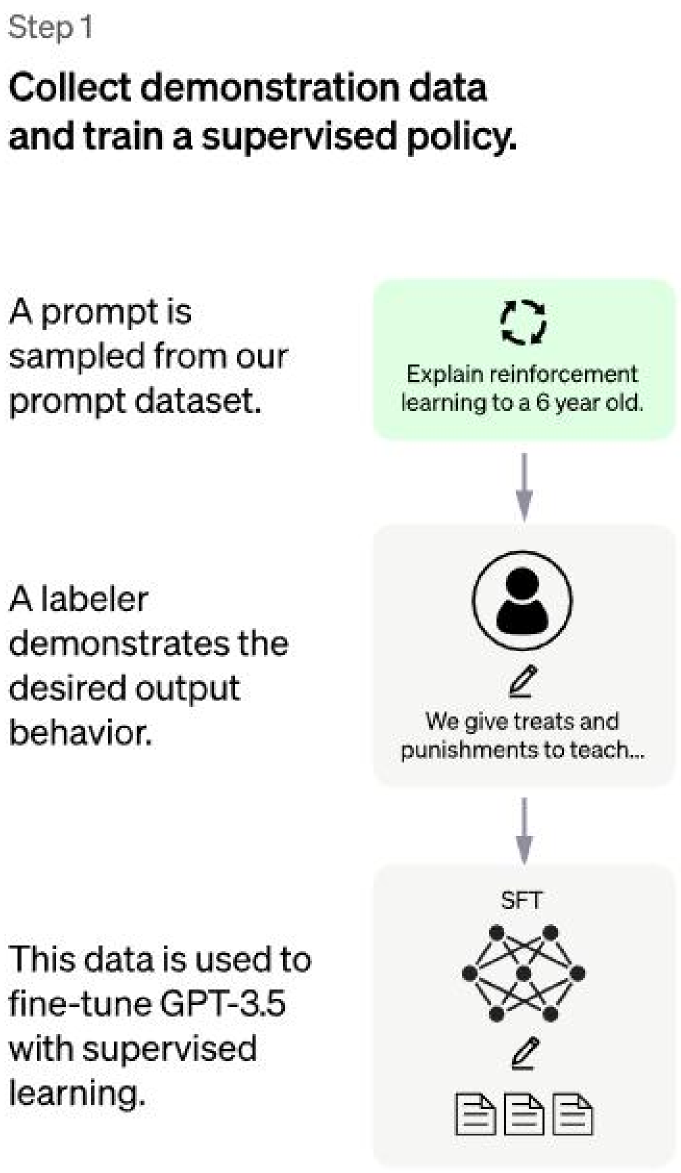
之前遇到的问题

模型越大，参数越大，真的越好吗?

打江山难，守江山更难，模型得为我所用才行

如何学人的逻辑，说人话，办人事呢

这就需要有监督学习了（再预训练模型基础上）



ChatGPT

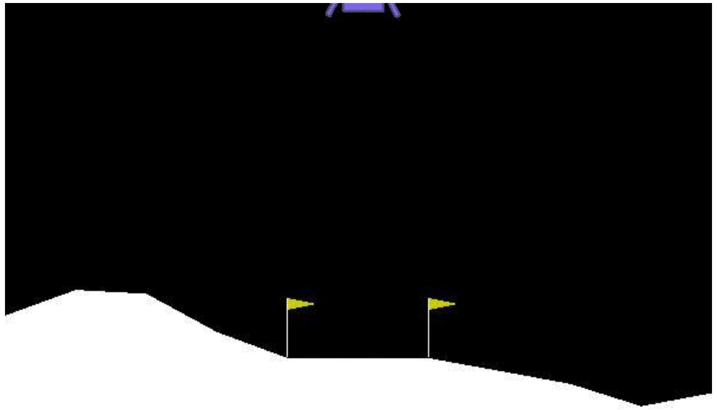
- 有监督学习

  - 那肯定得人工啊，你希望模型能输出啥，咱们就给他写点啥
  - 还是用GPT-3来微调咱们写的这些数据，让输出更符合问题
  - InstructGPT说就标了1W多个数据？这能信？感觉这个量级不够
  - 还是训练GPT，只不过用1W多个数据来微调一遍

强化学习

- 获得奖励

  - 先来玩一个小游戏，虽然短，但是经历了好多过程：



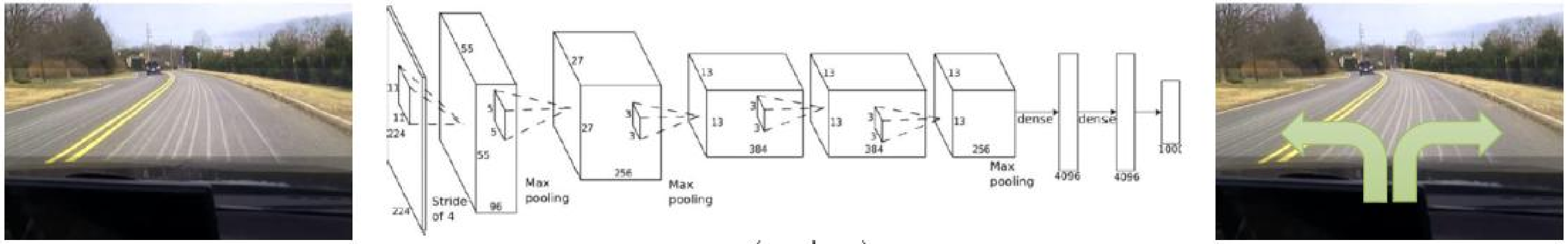
  - 飞船每一步行动都会获得不同的结果（奖励）

  - 一个完整的过程，通常叫做**episod**，整个生命周期的奖励：$R = \sum_{t=1}^{T} r_t$

强化学习

- 网络的输入与输出

- 一次游戏的记录结果：
- 包括了每一步的状态与行动（trajectory）：$\tau = \{s_1, a_1, s_2, a_2, \cdots, s_T, a_T\}$
- 每一步如何走才能得到更多的奖励呢？这就需要训练好神经网络了！



$\mathbf{s}_t$ $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ $\mathbf{a}_t$

ChatGPT

- 监督学习VS强化学习

- 监督学习有点死板，就是要预测出来正确答案，对就是对，错就是错

- 强化学习它没有一个标准答案，给我们生成的结果来进行打分评判

- 想一想你找你导师讨论问题，他会告诉你每一步该怎么做，该做什么吗?（学习应该一步登天，还是像不断尝试呢总结经验呢？）

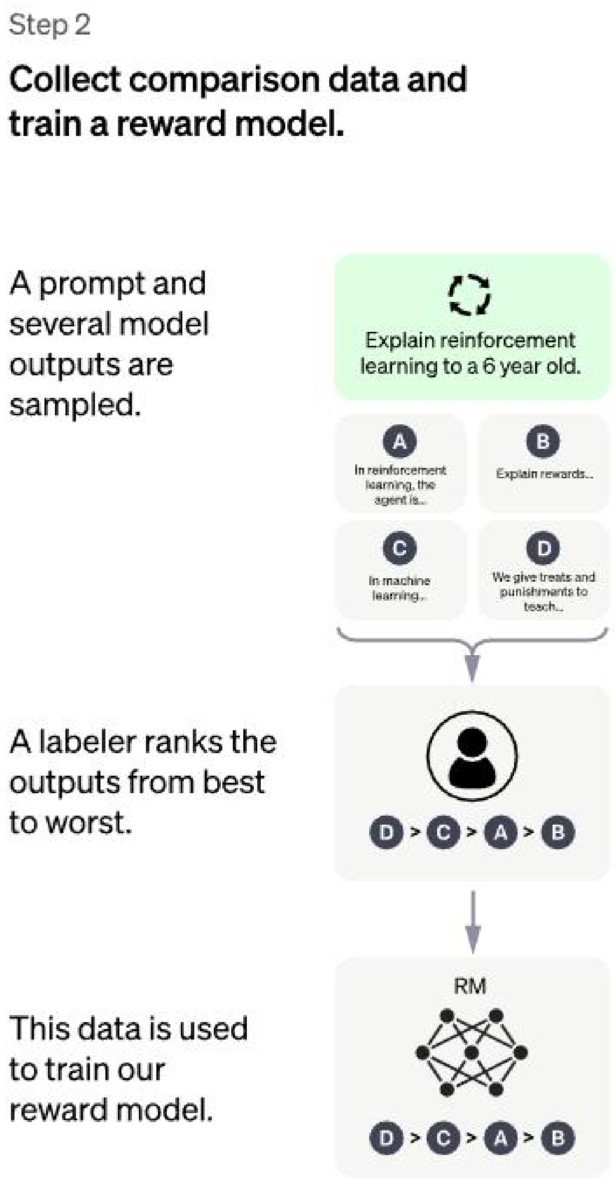
- 大概率会告诉你做的不行，不好，不对，但是为什么这件事就得你去思考了（实际上人家这是要激发你的潜能，别被知识所束缚，去创造去。。。）

ChatGPT

奖励模型

- 这哥们是需要训练的，它得能分出来好赖话（打分）
- 输入：你瞅啥；输出：1.没瞅啥啊；2.你说啥？
- 3.我就随便看看；4.瞅你咋滴啊；（得到排序结果）
- 奖励模型也是GPT3，但是它是蒸馏版本？（只有6亿）

$$\operatorname{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D}\left[\log\left(\sigma\left(r_\theta\left(x, y_w\right) - r_\theta\left(x, y_l\right)\right)\right)\right]$$



ChatGPT

- 奖励模型

  - 仅选6亿参数量，论文中强调1750亿的验证集损失很低，但效率一般
  - 在GPT小版本（6亿）中继续选择部分公开数据集训练后得到的初始化模型
  - 将最后一层（2048向量）直接连一个FC来预测一个得分就可以了
  - 也包括了前几代版本的训练数据，整合一块后开始训练

ChatGPT

- RL登场

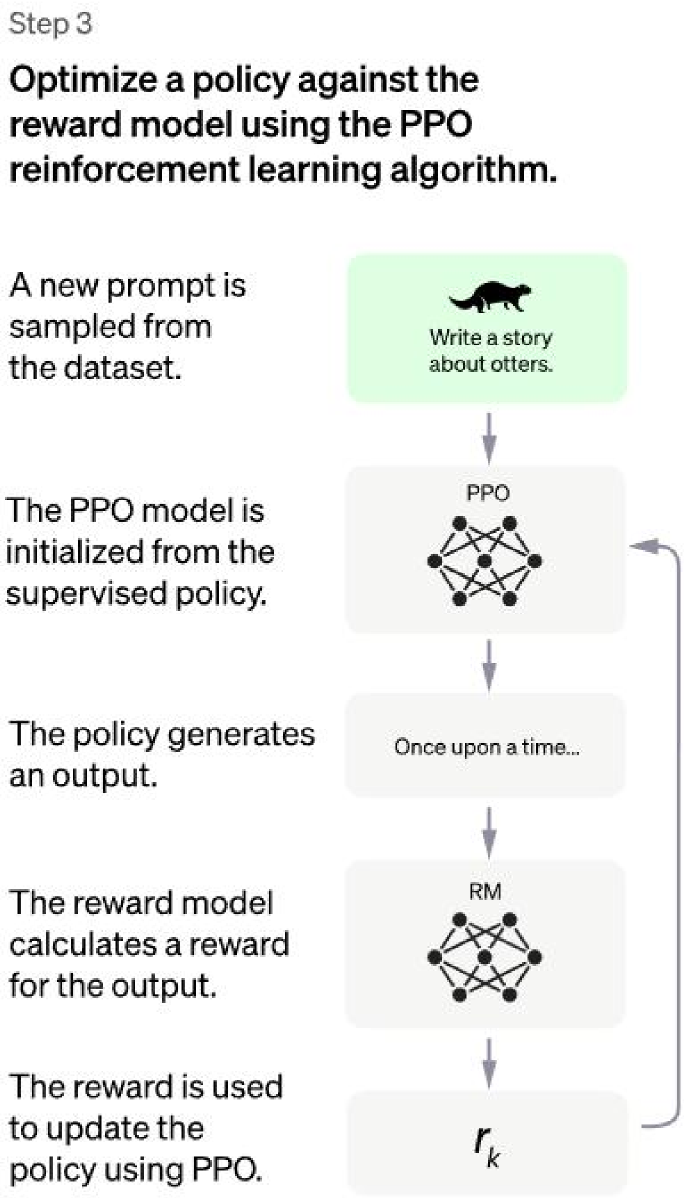
- 我们需要的模型就是通过RL来更新的

- 模型输出的句子通过奖励模型得到得分，再反馈

- 而且模型更新一阵之后，也需要再更新奖励模型（本是同根生，但是没有相煎何太急）

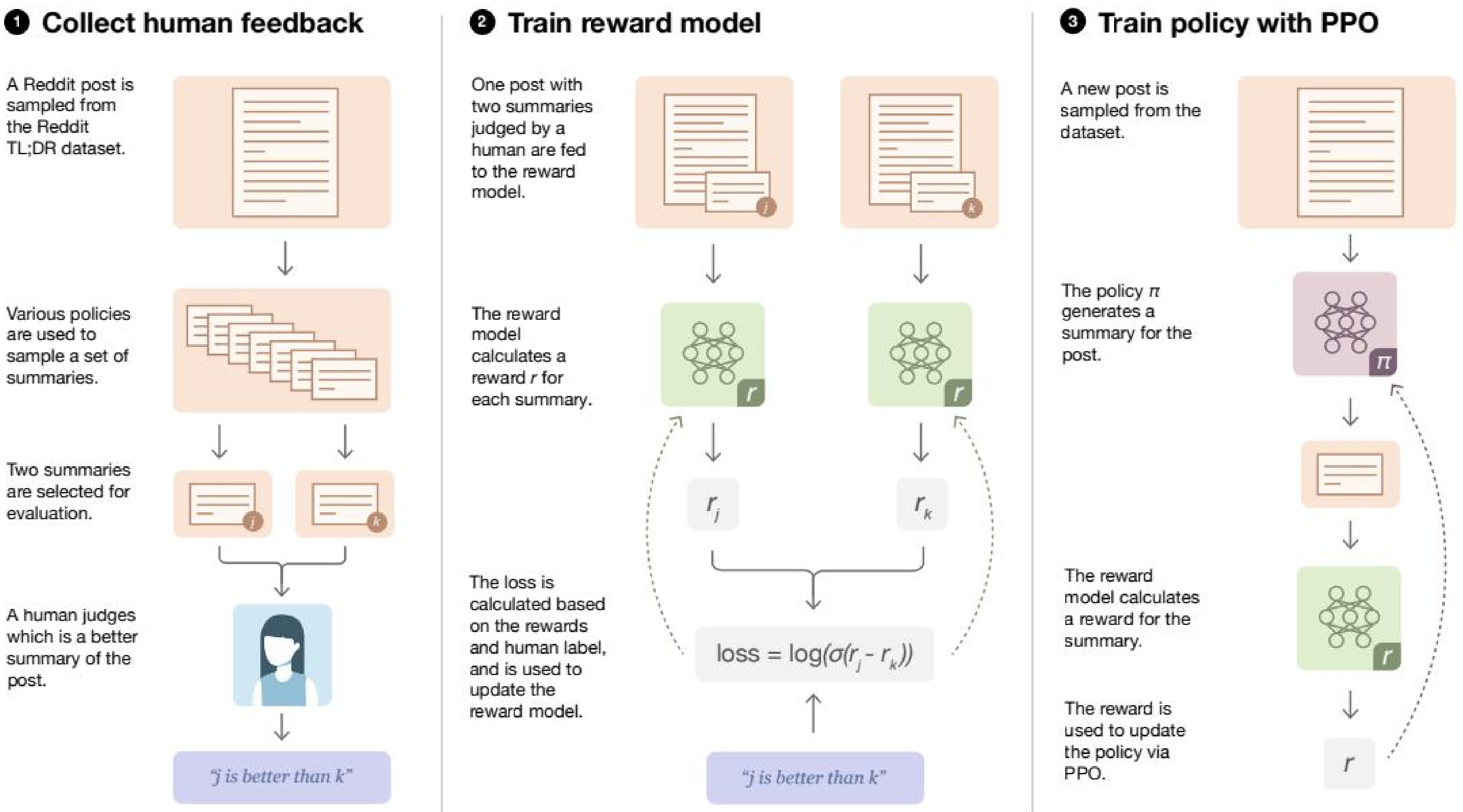
- 目标是这样的：得分-差异(以人为主)+泛化能力

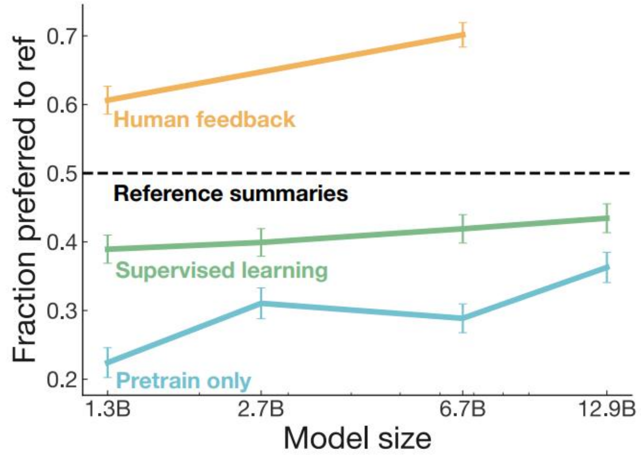
$$\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta\log\left(\pi_\phi^{\mathrm{RL}}(y\mid x)/\pi^{\mathrm{SFT}}(y\mid x)\right)\right] + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]$$



ChatGPT

回顾：Learning to summarize from human feedback





ChatGPT

结果分析

多个维度上效果都有提升，主要更能满足咱们的约束条件