真·万字长文：可能是全网最晚的 ChatGPT 技术总结 - TechBeattech

> TechBeat 是荟聚全球华人 AI 精英的成长社区，每周上新来自顶尖大厂、明星创业公司、国际顶级高校相关专业在读博士的最新研究工作。我们希望为 AI 人才打造更专业的服务和体验，加速并陪伴其成长。

最近 ChatGPT 可以说是火遍了全世界，作为由知名人工智能研究机构 OpenAI 于 2022 年 11 月 30 日发布的一个大型语言预训练模型，他的核心在于能够理解人类的自然语言，并使用贴近人类语言风格的方式来进行回复。模型开放使用以来，在人工智能领域引起了巨大的轰动，也成功火出了技术圈。从数据上看，ChatGPT 用户数在 5 天内就达到了 100 万，2 个月就达到了 1 亿；另外，在很多非人工智能领域，已经有机构在尝试用 ChatGPT 去做一些智能生成的事。例如财通证券发布了一篇由 ChatGPT 生成的行业研报，从研报的可读性和专业性上来看，虽然在细节上有很多需要推敲的地方，但是整体框架内容已经比较成熟。对于其他内容生产者来说，应用 ChatGPT 也能够提升个人的生产效率。

ChatGPT 的强大能力是显而易见的，但对于人工智能领域不太熟悉的人，对这种黑盒的技术仍然会担忧或者不信任。恐惧通常来自于不了解，因此本文将为大家全面剖析 ChatGPT 的技术原理，尽量以简单通俗的文字为大家解惑。

通过本文，你可以有以下收获：

1、知道 ChatGPT 是什么

2、ChatGPT 有哪些核心要素

3、ChatGPT 能做哪些事

4、ChatGPT 不能做哪些事

ChatGPT 是什么?

上文说到 ChatGPT 实际上是一个大型语言预训练模型（即 Large Language Model，后面统一简称 LLM）。什么叫 LLM？LLM 指的是利用大量文本数据来训练的语言模型，这种模型可以产生出强大的语言关联能力，能够从上下文中抽取出更多的信息。其实语言模型的研究从很早就开始了，随着算力的发展和数据规模的增长，语言模型的能力随着模型参数量的增加而提升。下图分别展示了 LLM 在参数量和数据量上的进化情况，其中数据量图例展示的是模型在预训练过程中会见到的 token 数量，对于中文来说一个 token 就相当于一个中文字符。

为什么语言模型的参数量和数据量会朝着越来越大的方向发展呢？在早些时间的一些研究已经证明，随着参数量和训练数据量的增大，语言模型的能力会随着参数量的指数增长而线性增长，这种现象被称为 Scaling Law（下图左例）。但是在 2022 年之后，随着进来对大模型的深入研究，人们发现当模型的参数量大于一定程度的时候，模型能力会突然暴涨，模型会突然拥有一些突变能力（Emergent Ability，下图右例），如推理能力、零样本学习能力等（后面均会介绍）。

ChatGPT 真正强大的地方在于他除了能够充分理解我们人类的问题需求外，还能够用流畅的自然语言进行应答，这是以前的语言模型不能实现的。下面，本文将 ChatGPT 一分为二，分别从 GPT 和 Chat 两个维度来介绍 ChatGPT 的机理。值得说明的是：当前 OpenAI 并未放出 ChatGPT 相关的训练细节和论文，也没有开源代码，只能从其技术 BLOG

上获取其大致的训练框架和步骤，因此本文介绍的内容将根据后续实际发布的官方细节而更新。

GPT

GPT 全称 Generative Pre-training Transformer，由 Google 在 2018 年提出的一种预训练语言模型。他的核心是一个 Transformer 结构，主要基于注意力机制来建模序列中不同位置之间的关联关系，最后可用于处理序列生成的任务。通过使用大量的文本数据，GPT 可以生成各种各样的文本，包括对话、新闻报道、小说等等。上面提到了很多次语言模型，这里简单给出语言模型主要的涵义：

给定已知的 token 序列 N_t（对中文来说是字符，对英文来说可能是单词或者词根），通过语言模型来预测 t+1 位置上的 token 是什么。实际上模型输出的是所有 token 在 t+1 位置上的概率向量，然后根据概率最大的准则选择 token。大家在使用 ChatGPT 的时候，一定有发现机器人在生成回复的时候是一个字一个字的顺序，背后的机制就是来自于这边。

对语言模型来说，可能大家之前更熟悉的是 BERT，BERT 是 Google 在 2018 年发布的一种双向语言模型，发布后，其在不同语言理解类任务（如文本分类，信息抽取，文本相似度建模）中都达到了当期时间节点的最好效果。BERT 与上述语言模型的机理有所不同，其训练任务相当于让模型去做完形填空任务（官方称为 Masked Language Model 任务，下文简称 MLM)，并不是遵循文本一个接一个预测的顺序，其模型机制与人类沟通表达的习惯不太符合。图中左半部分是 BERT 的示意图，右半部是 GPT 的示意图，Trm 为一个 Transformer 模型组件，E 为输入的 token 序列，T 为模型生成的 token 序列。

其中，实线部分为该位置的 Trm 能够看到哪些其他位置 token 的上下文知识。可以看到，对于 BERT 来说，每个位置上的 Trm 都能看到任意位置的上下文知识，因此其在具体的自然语言理解任务上会有不错的效果。而 GPT 则是遵循传

统语言模型的模式，例如 index=1 位置的 Trm 是无法看到 index>1 的知识的，因此它在自然语言理解任务上的效果不如 BERT，但是在生成任务上会更符合人类的直觉。业界把 BERT 中的 MLM 模式称为自编码形式 (auto-encoding)，把 GPT 的模式称为自回归形式（auto-regressive）。

大家从 BERT 和 GPT 的对比中可以看到，BERT 在语言理解上似乎更具优势，那为何现在 ChatGPT 的模型基座是 GPT 呢？这就涉及到最近两年逐渐清晰的 NLP 任务大一统趋势了。

NLP 任务大一统

基于 MLM 训练范式得到的 BERT 模型虽然在很多语言理解类任务上有不错的效果下游任务，之后整个业界在处理 NLP 任务的时候通常会遵循预训练模型→下游任务 finetune 的流程：

这种方式与传统的 training from scratch 相比，对下游任务数据的需求量更少，得到的效果也更优。不过，上述方式还是存在一些问题：

1. 处理一个新的任务就需要标注新的语料，对语料的需求比较大，之前已经做过的任务语料无法高效利用。即使是信息抽取下面的不同任务（如实体识别和关系抽取两个任务）也无法通用化。

2. 处理一个新的任务需要针对任务特性设计整体模型方案，虽然 BERT 模型的底座已经确定，但还是需要一

定的设计工作量。例如文本分类的任务和信息抽取的任务的模型方案就完全不同。

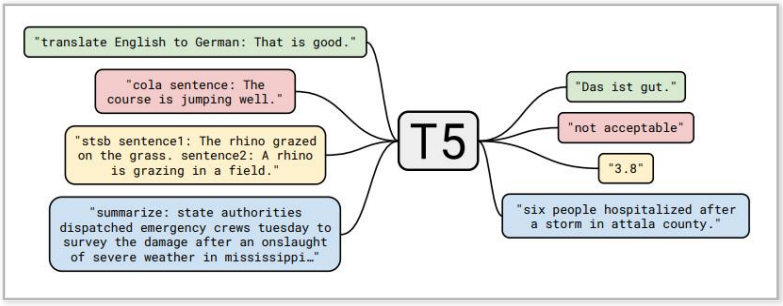
对于要走向通用人工智能方向的人类来说，这种范式很难达到通用，对每个不同任务都用单独的模型方案和数据来训练显然也是低效的。因此，为了让一个模型能够尽量涵盖更多的任务，业界尝试了几种不同的路径来实现这个目标。

- 对 BERT 中的 MLM 进行改造，如引入一些特殊的 Mask 机制，使其能够同时支持多种不同任务，典型的模型如 UniLM https://arxiv.org/abs/1905.03197

- 引入额外的 Decoder，将 BERT 优化改造成能做生成式的模型，典型的工作有 BART（https://arxiv.org/abs/1910.13461），T5（https://arxiv.org/pdf/1910.10683.pdf），百度的 UIE（将任务设计生成 text-to-structure 的形式实现信息抽取的大一统）。我对 T5 比较熟悉，之前也写过相关的分析，这个工作算是比较早地尝试将不同任务通过文本生成的方式进行大一统。如图所示，T5 训练时直接输入了不同下游 NLP 任务的标注数据，通过在原始文本的前端添加任务的提示文本，来让模型学习不同任务的特性。如翻译任务可以是”translate English to German”，分类任务可以是跟具体分类目标有关如”cola sentence”，也可以是一种摘要任务”summarize”。

> 怎么样，是不是觉得跟 ChatGPT 的模式有相似的地方?

这种方式可以同时利用多种 NLP 任务相关的公开数据集，一下子就把预训练任务从语言模型扩展到了更多任务类型中，增强了模型的通用性以及对下游任务的理解能力。



- 除了上面两种方式外，还有其他改造 BERT 的方法就不穷举了，如苏神通过 Gibbs 采样来实现 BERT 模型的文本生成等。
  （https://kexue.fm/archives/8119 ）

虽然有很多大一统的路径，但是 OpenAI 一直坚持着 GPT 的方向不断演化着，2019 年他们发布了 GPT2，这个模型相对于 GPT 来说，主要是扩大了参数量，扩大了训练语料，在构建语料的时候隐式地包含了 multitask 或者 multidomain 的特质，最后在二阶段验证模型的时候并不是直接做有监督的 finetune，而是继续用下游数据做无监督的训练，最后的效果居然还不错，证明了只要模型够大，就能学到足够的知识用于处理一些下游任务。从它的论文名字就可以看出其核心思想： Language models are unsupervised multitask learners 。不过彼时，BERT 及其各种变种在领域中的应用还是更广的，真正让 GPT 系列模型惊艳众人的工作还是要数 2020 年发布的 GPT-3 模型。 Language Models are Few-Shot Learners

GPT-3

首先，说几个跟 GPT-3 相关的数字：

	GPT-2	GPT-3
模型参数量	1.5B	175B
训练语料规模	40GB	600GB

OpenAI 训练初版的 GPT-3，比 GPT-2 整整用了 15 倍的语料，同时模型参数量扩展了 100 多倍。这么多资源的投入，

使得 GPT-3 成为了一个 “庞然巨物”，其产生的效果也是惊人的。除了在很多 NLP 的任务上有着很不错的指标外，其本身也产生了一种前所未有的能力——In-context learning。

何为 In-context learning?

简单来说，就是模型在不更新自身参数的情况下，通过在模型输入中带入新任务的描述与少量的样本，就能让模型” 学习” 到新任务的特征，并且对新任务中的样本产生不错的预测效果。这种能力可以当做是一种小样本学习能力。可以参考下图的例子来理解：其中，task description 和 examples 用来帮助模型学习新任务，最后的 Prompt 用来测试模型是否学会了。

与传统的小样本学习范式还是有所不同，之前主流的小样本学习范式以 Meta-learning 为主，通过将训练数据拆成不同的小任务进行元学习。在学习的过程中，模型的参数是一直在变化的，这是最大的一个不同点。

那不更新参数的小样本学习有什么好处呢?

对于大模型来说，这可是极佳的特性。因为大模型的微调成本通常都极为庞大，很少有公司能够具备微调训练的资源。因此，如果能够通过 In-context learning 的特性，让大模型快速学习下游任务，在相对较小的成本下（对大模型进行前向计算）快速完成算法需求，可以大大提升技术部门的生产力。

In-context learning 的效果固然惊艳，但是对于一些包含复杂上下文或者需要多步推理的任务仍然有其局限性，这也是业界一直以来致力于让人工智能拥有的能力——推理能力。那么大模型具有推理能力吗？对于 GPT-3 来说，答案是可以有，但有一定的限制。我们先来看看它有的部分。

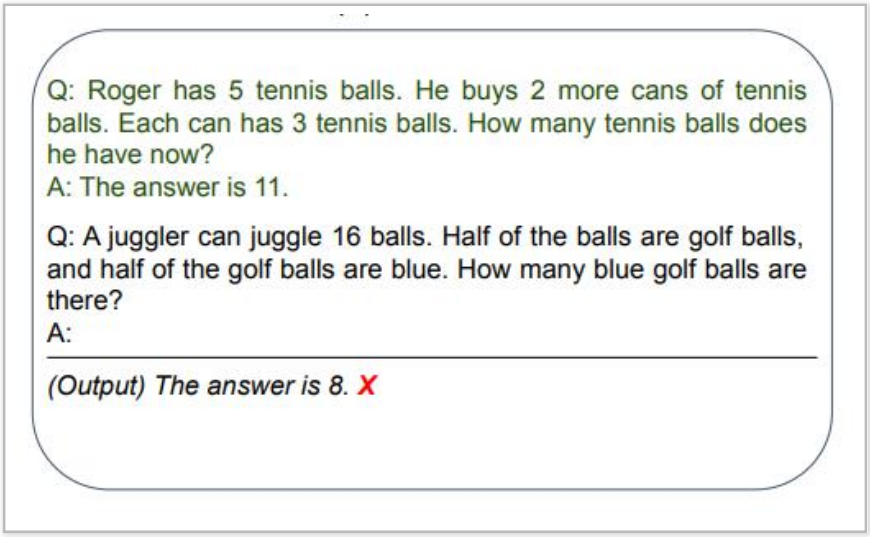
还记得文章开头提到的大模型的涌现能力吧，In-context 正是属于当模型参数量达到一定程度后，突然出现的能力之一。那么除此以外，还有什么能力是涌现的呢？答案就是——Chain-of-thought，即思维链能力。

怎么理解 In-context learning?

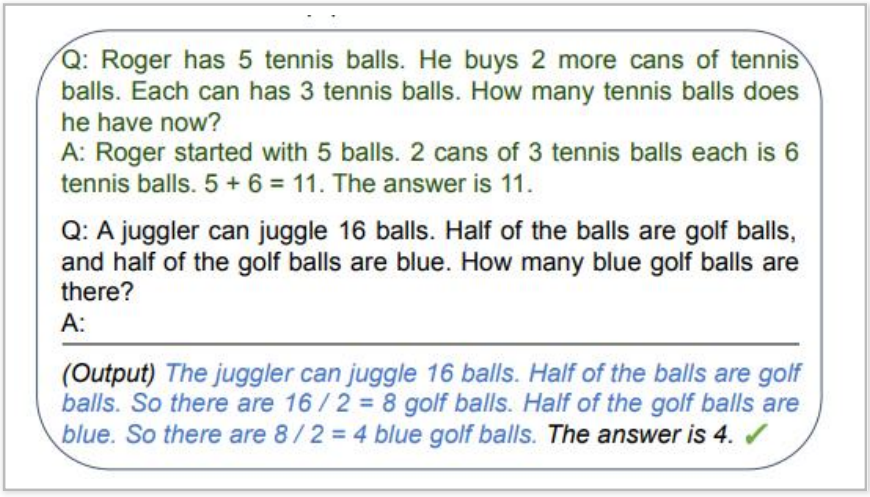
GPT-3 拥有的 In-context learning 能力可以说有很大程度来自于其庞大的参数量和训练数据，但是具体能力来源仍然难以溯源。不过，最近已经有一些论文专门针对其进行了研究，如清华大学、北京大学和微软的研究员共同发表了一篇论文： https://arxiv.org/abs/2212.10559 ，探索了 GPT 作为一个语言模型，可以视作是一个元优化器，并可将 In-context learning 理解为一种隐性的微调。

何为 Chain-of-thought（COT）？

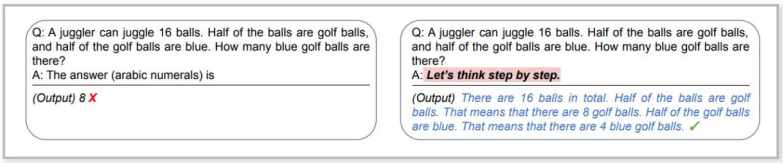
实际上是对输入的 Prompt 采用 Chain-of-thought 的思想进行改写。传统的 Prompt 中，对于一个复杂或者需要多步计算推导的问题样例，会直接给出答案作为 In-context learning 的学习范例与新任务的测试样例输入到大模型中。这样做往往不能得到正确的结果，如图所示：
（ https://arxiv.org/pdf/2205.11916.pdf ）



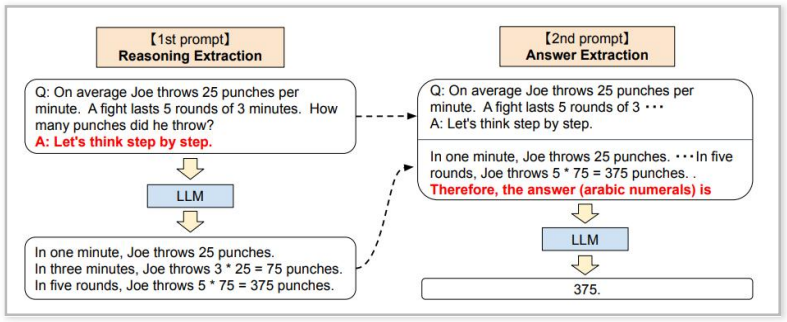
然而，当我们将上述问题范例中的答案再细化一些，对推到出答案的每一个步骤都写出来，再将测试样例一起输入到模型中，此时模型居然能够正确回答了，而且也能够参照范例中的样例进行一定的推理，如图所示：



上述的模型输入中，还带有可参考的问题范例，还属于小样本的范畴。诡异的是，有人使用了一种匪夷所思的方法，让其具备了零样本的推理能力：在问题样例的答案中增加一句 Let’s think step by step. 然后模型居然能够回答出之前不能回答的问题。



当然，上图中模型并未直接给出一个简洁的答案，而是给出了推导答案的步骤，论文中则是将上述 output 与输入模型的 Prompt 拼在一块，再次输入模型，最终得到了简洁的答案输出：



既然大模型具备了 COT 的特性，那么就能说明它具备了推理能力了吗？答案是不确定的。因为在更多的复杂逻辑推理类任务或者计算任务上，大模型还是无法回答。简单来说就是他可以做一些简单的小学应用题，但是稍微复杂一点的问题它就是在瞎猜了。具体的例子可以参考这篇论文中的分析：Limitations of Language Models in Arithmetic and Symbolic Induction

Chain-of-Thought 能力来自于哪儿?

上一小节在介绍 COT 特性的时候，都是统一用 GPT-3 来代表。其实，** 原始的 GPT-3 版本中并没有显著地发现其具备 COT 特性。对于大众来说，像是 chatGPT 突然就有了这样的能力。其实，在 chatGPT 出来之前，openAI 对 GPT-3 做了很多迭代优化工作。而 GPT-3 的 COT 特性就是在这些迭代优化中逐渐展现。但不可否认的是，目前仍然没有确定性的结论说明 COT 特性来自于具体哪些迭代优化。有些观点说是通过引入强化学习，有些观点则是说通过引入了指令微调的训练方式，也有些观点说是通过引入庞大的代码预训练语料，使得模型从代码逻辑中学习到了相应知识。推测的方式则是根据不同时间节点上的模型版本能力差进行排除法，虽然目前我们受限于技术能力只能从这些蛛丝马迹中去发现一

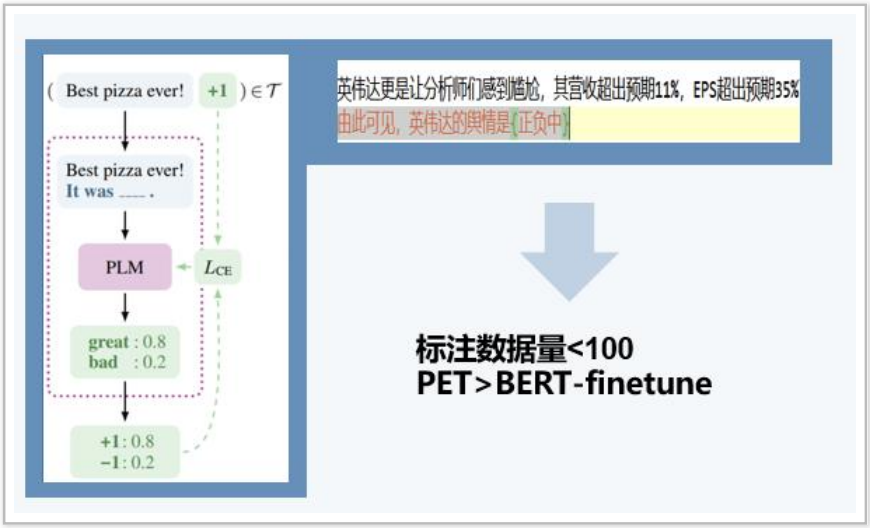
些端倪，但仍然具有一定的借鉴意义。具体的推理过程本文不会重复，感兴趣的可以参考如下博客：https://franxyao.github.io/blog.html 。

Instruction-Tuning 与 RLFH 技术

虽然对于大模型突变能力的来源还不能轻易下结论，但是在其迭代优化过程中，引入的一些技术确实提升了（更准确得说是激活）大模型的能力。根据 OpenAI 的技术博客所述，ChatGPT 的训练方式主要参考了 InstructGPT（https://arxiv.org/abs/2203.02155），而 InstructGPT 主要涉及了两个核心的技术实现：指令微调（Instruction-Tuning）以及基于人工反馈的强化学习（Reinforcement learning from Human Feedback），下面将对其进行介绍。

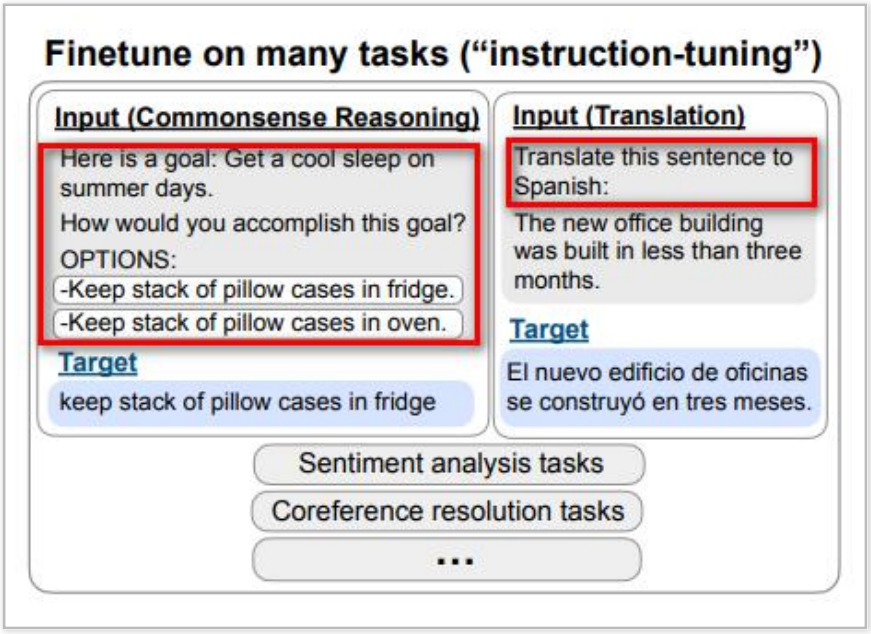
Instruction-Tuning

Instruction-Tuning（下称指令微调）技术，最早来自于谷歌 Deepmind 的 Quoc V.Le 团队在 2021 年发表的论文《Finetuned Language Models Are Zero-Shot Learners》（https://arxiv.org/abs/2109.01652）。在说指令微调前，必须得先介绍下 21 年初开始业界开始关注的 Prompt-learning 范式。2021 年 4 月，我在 InfoQ 的架构师大会上做了一次技术演讲，分享了我们在 Prompt 上的一些研究实践，如下图所示：

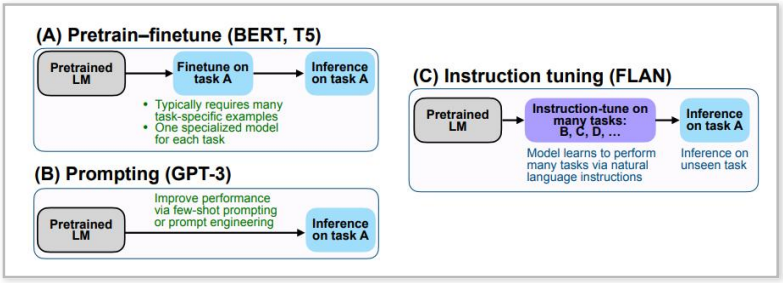


Prompt-learning 最早来自于论文《Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference》https://arxiv.org/abs/2001.07676，当时把里面的范式简称为 PET（Pattern-exploiting Training）。其核心思想为将不同类型的自然语言理解任务与 BERT 预训练中的掩码语言模型任务进行转化靠拢。例如对于图中的实体情感分类任务，本身其分类标签是一个三维的空间。我通过设置一个 prompt 提示文本模板：由此可见，英伟达的舆情是 {}，同时设计一个锚点，将原始分类目标的空间映射到语言模型中的子空间 {正 / 负 / 中}，通过预测锚点位置的 token 间接得到情感标签。这种方式的优点在于能够将下游任务与语言模型在预训练任务中的训练范式达成一致，减少下游任务在模型学习迁移过程中的知识损失，在小样本的场景下比普通的 Finetune 模式会有更好的效果。

Prompt-learning 实际上是一种语言模型能够股泛化不同任务的方式，从广义层面上来看，可以有多种实现方式，例如上面的 PET，本文之前提到的 T5 模型，以及初版的 GPT-3 等。指令微调实际上也可以算作是广义 Prompt-learning 中的一种实现方式（个人愚见）。它的核心思想是尽可能收集不同类型的自然语言处理任务（包括理解和生成），并使用自然语言设计对应的任务指令，让模型试图理解不同任务的指令与特性，最终通过语言模型生成的方式完成不同任务的训练，指令微调实例如下图所示：



那么指令微调与 BERT、T5、GPT-3 等 Prompt 方式有什么区别呢?



1. BERT 类的 Prompt 设计与掩码语言模型任务相关，Prompt 模板和锚点要与任务对应，需要一定量的标注样本进行小样本训练。

2. T5 的 Prompt 更像是在预训练时对不同语言任务的数据打上了不同的标记，让模型对语言任务有了初步的理解，但是不够深入，无法应用在零样本的场景。

3. GPT-3 的 Prompt 中，会基于在模型训练过程中见过的数据，更像是让模型将 Prompt 中的文本进行续写。这种方式可以帮助模型更好地理解用户输入的内容，并产生更准确和自然的输出。但其在零样本场景下效果仍然不佳。

4. 指令微调技术使用 Prompt 来为模型提供一系列指令或者命令，这些指令或命令会告诉模型应该如何进行特定任务的处理。与 GPT-3 中的 Prompt 不同，指令微调中的 Prompt 是针对特定任务和特定的模型进行设计的，相当于是指导模型如何完成任务。指令微调技术提升了模型的零样本学习能力。模型对于未见过的任务也能够理解并尝试处理。在 GPT-3 后续的迭代版本中，加入了指令微调后，即使在 Prompt 中不引入带标注的样本，模型也能够比较好的理解需求并得到不错的效果。

> 目前公开开源的模型 FLAN T5 就是在 T5 模型基础上进行了指令微调的训练，相较于那些动辄几百亿、几千亿参数的大模型来说，这个模型的参数量已经足够亲民，可以作为个人研究或者业务实现的 strong baseline

在 ChatGPT 公开后，各种五花八门的 Prompt 层出不穷。有让其作为一个 linux 终端的，有让其作为一个二次元猫娘的，也有让他写武侠小说的。感觉上 ChatGPT 可以做任何事情，只要你的脑洞足够大。这种通才特质有很大一部分要归功于指令微调。只要我们设计的 Prompt 指令足够清晰完整，模型总能够理解我们要干什么，并尽量按照我们的需求去完成任务。**我认为这是其有别于过往大模型的重要特性之一。**

深度强化学习简述

指令微调技术固然强大，但是其本身也存在一定的缺点：

1. 一些开放性的生成性语言任务并不存在固定正确的答案。因此在构建指令微调的训练集时，就无法覆盖这些任务了。

2. 语言模型在训练的时候，对于所有 token 层面的错误惩罚是同等对待的。然而在文本生成时，有些 token 生成错误是非常严重的，需要加权惩罚。换句话说，语言模型的训练任务目标与人类的偏好存在 gap。

综上，我们需要模型能够学习如何去满足人类的偏好，朝着人类满意的更新模型参数。因此，我们就需要引入人类对模型的奖惩方法（Reward）作为模型的引导，简称 $R\left(s\right) \in \Re$ R(s)∈ℜ.$R\left(s\right)$ R(s) 越高，模型的就越能满足人类偏好。很自然的，我们就能将最大化 $E_{\tilde{s}} \sim p\theta\left(s\right)\left[R\left(\tilde{s}\right)\right]$Es~~pθ(s)[R(s~)] , 即 R 的期望。一般来说，对于神经网络的训练来说，需要设计一个可微的目标函数，这样才能应用梯度下降法来对模型进行参数更新学习。然而，人类的 R 一般很难设计成可微的，因此不能直接用于神经网络的训练中，因此就有了强化学习的诞生。近年来，强化学习领域也在飞速发展，有了 alphaGo 系列的惊艳效果，有很多研究都将强化学习开始与深度学习进行了结合。比较典型的研究为 Policy Gradient methods（基于策略的梯度方法）。基于上述的训练目标函数，我们仍然应用梯度计算来进行参数更新：

$$\theta_{t+1}: = \theta_t + \alpha\nabla_{\theta_t}E_{\hat{s} \sim p_{\theta_t}\left(s\right)}\left[R\left(\hat{s}\right)\right]$$θt+1:=θt+α∇θt Es^~pθt(s)[R(s^)]

对于这个公式有两个问题：

1. 如何估计 R(*) 的期望函数?

2. 如果 R(*) 是一个不可微的函数，该如何计算梯度?

Policy Gradient methods 就是用来解决上述问题的。通过一系列的公式变换（过程就不放了，大家可以参考斯坦福 cs224n），可以得到以下式子：

$$\nabla_\theta E_{\hat{s} \sim p_\theta\left(s\right)}\left[R\left(\hat{s}\right)\right] = E_{\hat{s} \sim p_\theta\left(s\right)}\left[R\left(\hat{s}\right)\nabla_\theta \log p_\theta\left(\hat{s}\right)\right] \approx \frac{1}{m}\sum_{i=1}^{m} R\left(s_i\right)\nabla_\theta \log p_\theta\left($$
∇θEs^~pθ(s)[R(s^)]=Es^~pθ(s)[R(s^)∇θlogpθ(s^)]≈m1 i=1∑mR(si)∇θlogpθ(si)

我们将梯度计算移到了计算期望的式子内。虽然我们不能直接计算期望，但是可以采用蒙特卡洛采样的方法，去采样得到目标梯度的无偏估计。

将上式重新代入梯度更新的式子中，得到：

$\theta_{t+1} := \theta_t + \alpha \frac{1}{m} \sum_{i=1}^{m} R(s_i) \nabla_{\theta_t} \log p_{\theta_t}(s_i)$ θt+1:=θt+αm1 i=1∑mR(si)∇θtlogpθt(si)

此时，在梯度更新时候我们会有两种趋势：

- 当 R 为正的时候，说明对当前策略选择 $s_i$si 有奖励，因此我们需要让梯度沿着最大化 $p_{\theta_t}(s_i)$ pθt(si) 的方向更新
- 当 R 为负的时候，说明对当前策略选择 $s_i$si 有惩罚，因此我们需要让梯度沿着最小化 $p_{\theta_t}(s_i)$ pθt(si) 的方向更新

通过这种方式，我们就让模型逐渐逼近 R 所期望的方向学习。

ChatGPT 也将强化学习的技术进行了应用集成，通过人机结合，成功让模型学会了人类的偏好。这种技术就是 Reinforcement learning from Human Feedback, 以下简称 RLHF。

> 因为本人对强化学习领域不太熟悉，所以不足以完全解释其中的原理机制。因此主要参考斯坦福 cs224n 课程系列中对于该部分的宏观层面讲解。

RLHF

有了上面的强化学习技术，我们现在能够对一些不可微的函数进行梯度学习，我们就能引入一些符合人类期望的奖励函数作为模型训练目标。但是，这套工作流程让然存在一些问题：

- 整个训练过程需要人工不断对模型的策略选择进行奖惩的判断，训练的时间成本陡然上升。

为了降低训练成本，先标注适量的数据集，让人先给出偏好标注。然后，我们基于这个数据训练一个奖励模型 $RM_\phi(s)$

RMϕ(s)，用来自动生成人类对一个数据的偏好回答。

- 人本身会存在主观偏差，因此对数据的标注或者模型策略的评价也会有偏差。

为了能够对人类的主观偏差有一定的鲁棒性，不直接给出一个具体的好坏答复，而是采用一种 Pairwise Comparison 的方式，当生成一个文本输出时，人类可以对其进行成对比较，以指出其中更好或更合适的内容。例如，在文本摘要任务中，人类可以比较两个不同版本的摘要，并选择更好的那一个。这些成对比较可以帮助 InstructGPT 学习到人类的喜好和优先级，从而更好地生成高质量的文本输出。为了实现 Pairwise Comparison，需要设计一些有效的算法和策略，以便生成不同版本的文本输出，并对它们进行比较。具体来说，可以使用类似于基于排序的学习方法的算法来训练模型，并优化生成策略和模型参数，以便更好地满足人类反馈的需求：

$$J_{RM}(\phi) = -\mathbb{E}_{(s^w, s^l)\sim D}\left[\log \sigma(RM_\phi(s^w) - RM_\phi(s^l))\right]$$

"winning" sample　"losing" sample　s^w should score higher than s^l

图中，w 和 l 分别代表两个不同的模型生成结果，从人类的视角看 w 的结果更优，因此 w 的分数应该也要大于 l。

最后我们将 RLHF 的核心步骤串联起来：

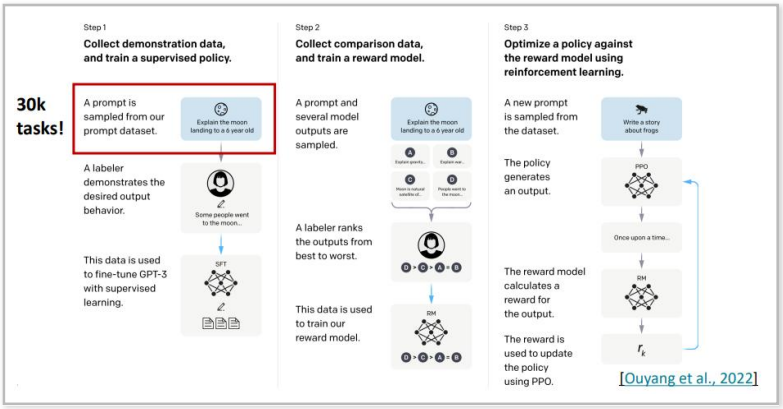
1. 初始状态下有一个通过指令微调方法训练后的语言模型 $p^{PT}(s)$ pPT(s)
2. 标注适量的数据，用于训练一个能够针对语言模型进行打分的 Reward 模型 $RM_\phi(s)$ RMϕ(s)
3. 用 $p^{PT}(s)$ pPT(s) 的权重参数初始化一个新的模型 $p_\phi^{PT}(s)$ pϕPT(s)，使用上面的基于策略的深度强化学习方法优化下面的 Reward：

$$R(s) = RM_\phi(s) - \beta \log \left(\frac{p_\theta^{RL}(s)}{p^{PT}(s)} \right) \quad \text{Pay a price when } p_\theta^{RL}(s) > p^{PT}(s)$$

除了 $RM_\phi(s)$ RMϕ(s) 外，上式还加了一个正则项。这个正则项可以防止通过强化学习更新的模型与原始的语言模型“跑的过于遥远”，可以看成是一条缰绳，让其保持基本的语言模型的特质。

InstructGPT 中的 RLHF

下图为目前最常见的 InstructGPT 训练流程。



1. 与上一小节中的通用 RLHF 流程不同，这里我们需要先用一些标注数据 finetune 一个 SFT 模型。训练任务与 GPT-3 的任务相同，因此数据也是采用 prompt-generation 的方式。构造的数据集的方式比较有讲究，首先要保证任务的多样性足够丰富；其次，对每个样本，标注着需要设计一个指令，然后生成多个问答对于该指令进行组合，用于组成一个小样本的 Prompt；最后就是 OpenAI 收集了实际服务当中产生的一些用户样例，这个数据能够让模型更切合实际使用的数据分布。

2. 构建 RM 数据集，并训练得到 $RM_\phi(s)$ RMϕ(s) 。为了减少人工的成本，会先用步骤 1 中得到的 SFT 模型为每个数据的 Prompt 产生 K 个生成结果，并引入人工根据结果进行质量排序。排序后的数据可以用来构

建 Pairwise Comparison 的数据，用于训练得到 $RM_\phi\left(s\right)$ RMϕ(s) 。

3. 基于策略优化的强化学习方法，以步骤 1 得到的 SFT 模型作为权重初始化模型，利用步骤 2 $RM_\phi\left(s\right)$ RMϕ(s) 对样本生成进行打分。

ChatGPT 中的 RLHF

根据 OpenAI 发布的技术博客所述，ChatGPT 的训练方式与 InstructGPT 几乎相同，仅在收集数据的时候采用了不同的方式，具体细节并没有公布，只提到他们让人工的标注人员同时扮演对话过程中的用户与机器人，并通过一系列准则规范指导他们如何编排对话中的回复，最终生成了对话场景的训练数据。最终，他们将其与 InstructGPT 的数据集进行的融合，并统一转化为对话的形式。另外，在训练 Reward 模型时，他们通过让人工标注人员与对话机器人进行对话来产生会话语料，并从中选择一个模型生成的消息，通过采样的方式生成多个不同的补全文本，并由标注人员进行打分排序，形成 Pairwise Comparison 数据。

ChatGPT 训练的工程难度

至此，本文将 ChatGPT 相关的技术要点已经做了一个整体的介绍，通过上文描述，我们可以看到 OpenAI 在研发 ChatGPT 的过程中投入了非常多的成本与研发精力，另外要训练出这样一个体量的模型，对于工程化的要求也是非常高的，包括对数据的清洗、大规模分布式训练的工程化以及大模型大数量下的训练稳定性技术等。就我个人而言，之前有研究并实施过 BERT-LARGE 模型的预训练，其参数量肯定不能与 ChatGPT 相比，但在训练中，也遇到过 loss 飘飞、训练中断卡壳的情况。因此，这样一个成果是算法与工程紧密结合的产物，其效果之好也就不奇怪了。

ChatGPT 的能与不能

当前，伴随着 ChatGPT 的接口开放，已经涌现出了很多有趣的应用。我按照自己的观察，总结了 ChatGPT 擅长做的以及不擅长做的事。

ChatGPT 的能

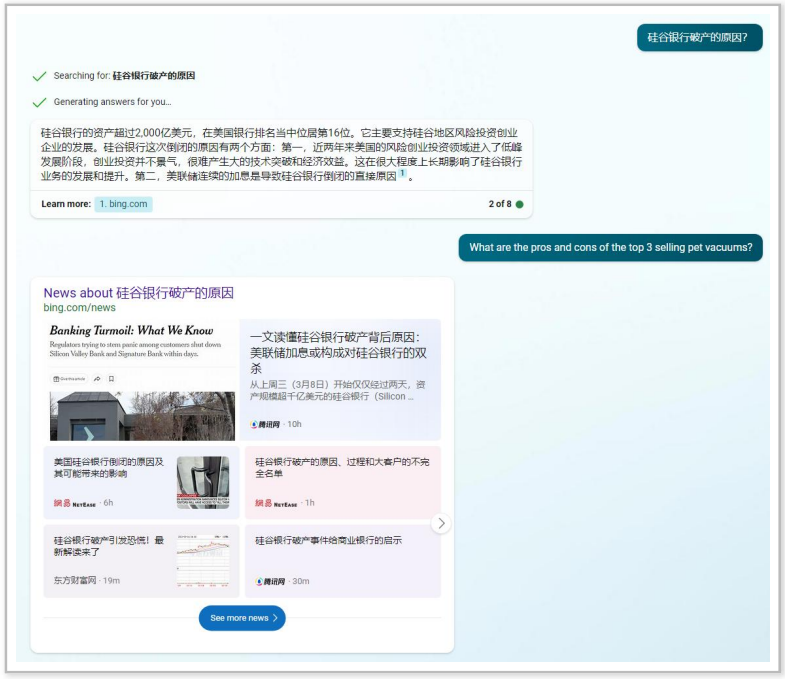
ChatGPT 虽然以对话机器人的产品形态提供服务，但是其本质上还是基于语言模型的能力。在应用层面上，他有三个强大的特质：

1. 对人类输入的需求理解能力特别强。
2. 善于进行知识的检索与整合。
3. 生成能力很强，在长距离生成过程中不会有重复、不通顺、机械等传统生成模型固有的问题。

因此，适合其大展身手的场景可包括：

基于搜索的问答

目前最典型的场景就是继承了 ChatGPT 的 New Bing。ChatGPT 本身存在知识信息无法自更新的缺点，导致产生的回复无法紧跟时代。因此，将搜索引擎与 ChatGPT 进行集成就显得水到渠成了。据了解，微软通过一种称为 “普罗米修斯” 的模型机制，将搜索与 ChatGPT 的生成能力进行了整合。以最近比较火的 “硅谷银行破产” 事件为例，我们有如下提问：



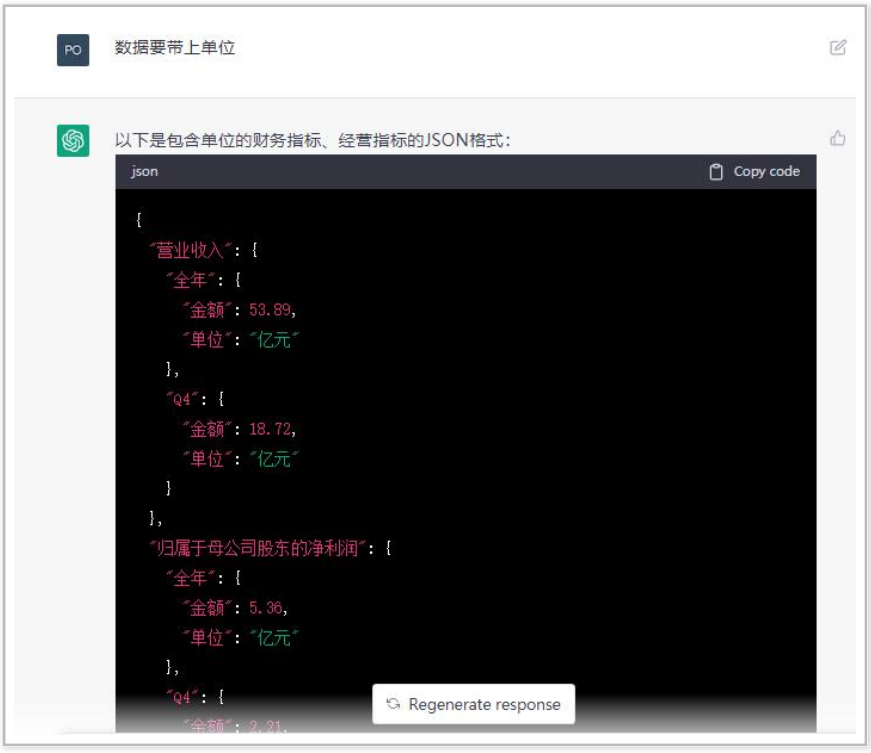
可以看到 New Bing 回答得还不错。从 New Bing 的回复方式可以去猜测其运行机制：先通过搜索引擎召回与问题相关的网络实时信息，并提取其中的文本。然后将所有相关文本构造成 Prompt 输入到 ChatGPT 中，要求其生成完整的回答。另外，在回答中还会标识出答案内容的来源。

除了 New Bing 之外，基于文档的辅助阅读也是非常典型的场景。最近比较火的 ChatPDF 能够上传论文等 PDF 文件，并支持对文档的 QA 问答。这实际上也是一种问答搜索。

处理各种基础的 NLP 任务

我们可以将他包装成一个通用的 NLP 工具平台，处理各种任务，包括但不限于文本分类、信息抽取、文本摘要、机器翻译等。通过上述章节的介绍可知，GPT-3 系列模型支持小样本和零样本学习的能力，因此应用他来做 NLP 任务可以降低人工标注的成本，并得到一个强大的 baseline。我们尝试了对文档进行信息抽取的任务，如研报公告中的财务经营指标抽取：

可以看到上面我采用的是零样本的模式，但是 ChatGPT 以几乎 100% 的准确率将所有指标抽了出来。不过，抽取出来的数据没有单位，我们还可以让他做个修正：



与其他组件的整合

基于 ChatGPT 强大的理解能力，我们可以把它作为一个人类与其他场景工具进行沟通的中间桥梁，大大提升个人的生产力。

- 例如日常办公涉及到的 OFFICE 全家桶，目前已经有了很多集成的产品，例如 ChatBCG，通过输入文字需求，就能自动生成 PPT 大纲以及每页的大致内容（当然，还不能自动生成多样的背景样式）；
  ChatExcel，通过输入文字需求，能够让其实现表格的基本处理、函数计算、分组过滤排序等复杂操作。

> 2023 年 3 月 17 日，微软宣布在 OFFICE 全家桶中集成 GPT-4。打工人的生产力一下子就提升数倍！

- 另外，还可以与其他模态的模型工具进行整合，例如 OpenAI 开放的 API 中就包括了 Whisper，一个语音识别的模型，人们可以通过 Whisper 将语音转文本，最终将文本送到 GPT-3 的接口中。另外，ChatGPT 也可以与图像视觉的大模型进行结合，提供文生图的功能，例如今年大热的 stable diffusion 模型。之前图像生成非常依赖输入的 Prompt 质量。我们可以让 ChatGPT 辅助生成一个高质量的 Prompt，然后输入到 stable diffusion 中，就能产生更符合需求的图像。

实际上，Meta 在 2 月份就发表了一篇论文 ToolFormer https://arxiv.org/abs/2302.04761 ），研究了如何使用自监督的方式，让大模型如何决定什么时候调用外部的 API 来帮助其完成任务。可以预见，后面会有越来越多的产品出来，我倒是希望能有一款根据文本要求自动画流程图的工具，毕竟受苦与画图很久了。

文字创作

作为一个生成式大模型，创作能力可以说是他的看家本领。ChatGPT 的创作场景格外丰富，只有你想不到，没有他做不到：

- 合并撰写工作周报与工作小结、小说创作、电影剧本创作等。但对于专业度和准确性比较高的场景，就不太能胜任了，例如金融场景中的研报生成，即使是将具体的财务数据连同要求一起输入模型，最后生成的结果中也会有一些事实性的数据错误，这种错误是无法容忍的。

- 可以作为一个 AI 辅助训练工具。当受限于成本无法使用 ChatGPT 直接提供 AI 能力时，不妨可以将 ChatGPT 视作一个数据增强器，生成任务所需要的训练语料，再辅以少量的人工进行核验，就能以较低的成本获得高质量的语料。

- 上述提到的 RLHF 训练流程也可以通过引入 ChatGPT 来减少人工的投入。具体来说就是将 Human feedback 替换为 ChatGPT feedback。早在 2022 年 12 月就有相关的论文介绍了这种思路：Constitutional AI: Harmlessness from AI Feedback (arxiv.org)

其实 ChatGPT 的应用场景还有很多，碍于篇幅，就不穷举出来了，大家可以自行关注相关媒体网站。

ChatGPT 的不能

ChatGPT 目前的应用非常广泛，看似是一个能干的多面手，但他也有目前无法胜任的场景。比较典型的就是推理分析。虽然在引入了代码以及其他迭代优化后，chatGPT 初步具备了一定的推理能力，但对于复杂的推理分析计算类任务，他回答错误的概率仍然非常大。这里特别推荐知乎上看到一个关于 ChatGPT 能力探索的博文：https://www.zhihu.com/question/582979328/answer/2899810576 。作者通过设计了一系列缜密的实验，来不断探

索 ChatGPT 的能力。从结果上可以看到机器的能力在某些场景上还是无法模仿人类的思维能力。

另外，在 ChatGPT 的训练过程中，使用了 RLHF 来引导模型按照人类偏好进行学习。然而，这种学习方式也可能导致模型过分迎合人类的偏好，而忽略正确答案。因此大家可以看到 ChatGPT 经常会一本正经的胡说八道。在专业领域，我们需要他知之为知之，不知为不知，不然我们就必须要引入人工来审核他的答案。

最后，应用大模型时绕不过的一个问题就是数据隐私安全。无论是 ChatGPT，还是国内即将推出的大模型，由于 B 端客户很少有硬件资源能够匹配上，很难进行私有化本地部署，通常是以 LaaS 的形式提供服务。而且目前大模型在专业垂直领域的效果还是未知的，因此通常需要使用领域语料进行微调，这就意味着数据要流出到模型服务提供方。一般大型公司对于数据的流出是非常慎重的，因此如何在安全合规的条件下，完成这一条链路的流转，是目前亟需解决的问题。

> 额外提一个应用：代码生成。这个场景既是能也是不能。他在 python 语言的编码能力上确实不错，甚至能生成一段 textcnn 的实现；但是在 java 或者其他编程语言上，他的生成质量就相对较差了，而且生成的代码质量也不如一个经验丰富的工程师，在代码执行性能上暂时还无法满足需求。

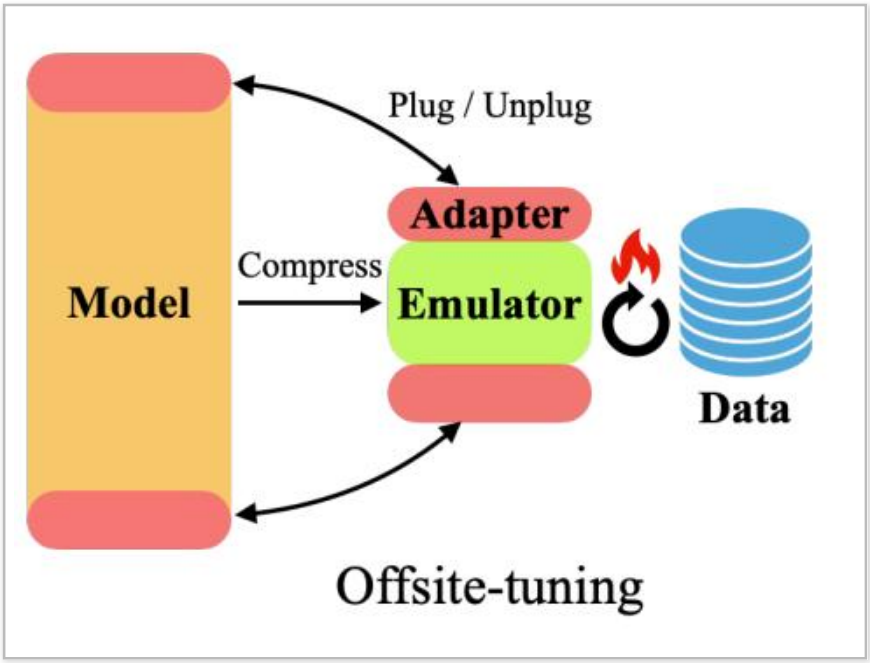
关于大模型的可研究方向

关于 ChatGPT 的内容到这也就基本写完了。作为一名 NLP 领域的从业者，我也跟其他人一样，被其强大的能力所震惊，同时也在思考自己未来还能在这个领域做哪些事情，大概想了一些方向，欢迎共同讨论：

- 用更少的参数量，达到更好的效果。无论是之前 DeepMind 的 Chinchilla(70B)，还是最近 Meta 的 LLaMA（65B）， 亦或是 3 月 14 日智谱团队刚发布

的 ChatGLM（6B），他们的参数量都小于 GPT-3（175B），但是其模型效果都能够匹配上 GPT-3。在 LLaMA 的论文中，Meta 表示他们用了更多的语料 token 来训练，这有可能意味着目前大模型的参数对知识的利用率还有很大的上升空间。我们可以通过精简参数，扩大语料规模来提升大模型的能力。

- 上面提到大模型应用时的数据隐私问题，目前也有一些可行的方法来解决。比如通过隐私计算的方式，让数据在流出时处于加密的状态。另外，也有一些学者在研究其他方法保护数据的隐私，例如 Offsite-Tuning（https://arxiv.org/pdf/2302.04870v1.pdf），这种方法的核心思想是设计了一个 adapter（可以理解为一个由神经网络构成的组件）与仿真器（可以理解为大模型的一个压缩版本）并提供给用户，用户在仿真器的帮助下使用领域数据对 adapter 参数进行微调，最后将微调好的 adapter 组件层插入到大模型上组成了一个完整的新模型用于提供服务：



- 高效设计与应用 ChatGPT 的 Prompt 范式。例如我们可以设计一个工具平台，将不同类型的 NLP 任务包装成一种配置式的产品。用户针对自己的任务需求，

只需要提供需求的详细描述，以及问题的样例，就能快速得到一个能力实例，并应用在自己的场景中；另外，我们还可以研究如何高效地设计一个 Prompt 来解决复杂的场景问题。如 Least-to-Most(https://arxiv.org/abs/2205.10625 。这篇论文所述，对于一个复杂问题，我们可以帮助 LLM 先自己拆解问题，形成为了解决问题 X，需要先解决问题 Y1,Y2... 的形式, 然后让模型分别去解决子问题，最后将所有子问题的解决过程拼在一块送到模型中，输出答案。这种方式可以有机结合 COT 的特性，可以用于处理一些比较复杂的问题。

结束语

在本文的最后来一些鸡汤吧：时代的车轮是不断向前的，技术的更迭也会给这个时代带来不可估量的影响。虽然 ChatGPT 的出现可能会对业界带来不小的冲击，但我们应该将目光放到更广阔的天地，在那儿将有更多丰富的未知世界等着我们去探索。

以此自勉!

全文完