

LLM大模型综述

文章目录

一、理论篇

1. 什么是LLM
2. LLM发展历史
3. LLM构建流程
4. LLM主流架构
5. LLM评估指标
6. LLM排行榜
7. LLaMA的网络结构
8. 强化学习与有监督学习的区别
9. LLM从海量文本中学习到了什么
10. LLM的知识存储在网络的什么地方
11. 如何修改LLM中的某些知识
12. 如何消除模型幻觉
13. 如何平衡训练数据量、模型参数、增加epoch的关系
14. ChatGPT的优缺点与最大贡献
15. GPT3在算数方面的能力
16. 思维链prompting是啥
17. chatGPT中的语言模型跟传统语言模型最大区别
18. 代码预训练可以增强LLM推理能力
19. 大模型智能涌现的参数临界点
20. flash-attention为什么能加速
21. 什么是bpe算法
22. Decoder在训练时，输入是真实标注label，但实际使用时输入的是上一次的解码结果
23. gpt4距离超越人类智力还有多远

二、实践篇

1. 如何下载llama2模型
2. 免费试用gpt4的网站
3. prompt提示应该怎么写效果会更好
4. 预训练数据集概览
5. InstructGPT模型微调数据集
6. 指令微调数据集格式
7. 训练数据准备阶段
8. 如何对训练加速
9. ChatGLM3实践
10. Ilma-index能比较方便构建文档索引，并直接提问
11. 金融领域评测数据集
12. 如何加速Qwen模型的推理速度
13. LoRa和QLoRA有什么区别？
14. 训练大模型的一键流程框架
15. 中文llama3
16. GaLore比lora更加高效降低所需GPU显存
17. chatGPT的输入有长度限制，怎么办
19. 如何制作“针对某个pdf的问答机器人”
20. 开源项目
 - a. RLHF的开源实现
 - b. 目前复现水平最接近chatGPT的开源模型是Vicuna
 - c. 其他
21. GPT模型的token和汉字的换算关系
22. GPT3.5的价格如何？
23. 什么是LoRA训练
24. 目前已有的大规模参数训练框架

三、应用篇

1. Arc搜索
2. 秘塔搜索
3. lepton搜索
4. dify.ai
5. 如何使用chatGPT预测股价

四、参考文档

待阅读

一、理论篇

1. 什么是LLM

LLM其实就是large language model，大语言模型。AGI其实就是Artificial General Intelligence。NLP大致可以分为两大任务：**NLP理解类任务**和**NLP生成类任务**。这两类任务的差异主要体现在输入输出形式上。

1. 理解类任务的特点是，输入一个句子（文章），或者两个句子，模型最后判断属于哪个类别，所以本质上都是分类任务，比如文本分类、句子关系判断、情感倾向判断等。
2. 生成类任务的特点是，给定输入文本，对应地，模型要生成一串输出文本，比如聊天机器人、机器翻译、文本摘要、问答系统等。

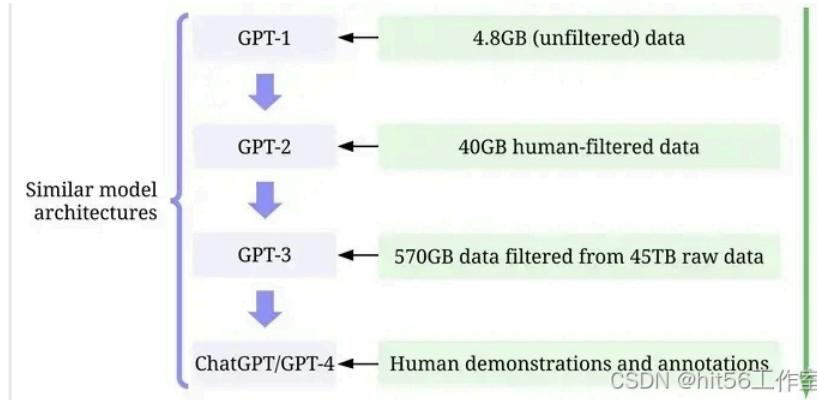
NLP 各种任务其实收敛到了两个不同的预训练模型框架里：

1. 对于NLP理解类任务，其技术体系统一到了以Bert为代表的“预训练+Fine-tuning”模式；
2. 对于NLP生成类任务，其技术体系统一到了以GPT为代表的“自回归语言模型（即从左到右单向语言模型）+Zero /Few Shot Prompt”模式。

2. LLM发展历史

1. 【Transformer模型】：2017年6月，Google发布论文《Attention is all you need》，首次提出Transformer模型，成为GPT发展的基础。
2. 【GPT模型】：2018年6月，OpenAI发布论文《Improving Language Understanding by Generative Pre-Training》，首次提出GPT模型。
3. 【GPT2模型】：2019年2月，OpenAI发布论文《Language Models are Unsupervised Multitask Learners》，GPT2使用了与GPT1相同的模型和架构，但GPT2更加侧重于Zero-shot设定下语言模型的能力。在GPT1的基础上引入任务相关信息作为输出预测的条件，将GPT1中的条件概率 $p(\text{output}|\text{input})$ 变为 $p(\text{output}|\text{input}; \text{task})$ ；并继续增大训练的数据规模以及模型本身的参数量，最终在Zero-shot的设置下对多个任务都展示了巨大的潜力。这样的思想事实上是在传达只要模型足够大，学到的知识足够多，任何有监督任务都可以通过无监督的方式来完成，即任何任务都可以视作生成任务。
4. 【GPT3模型】：2020年5月，OpenAI发布论文《Language Models are Few-Shot Learners》。GPT3使用了与GPT2相同的模型和架构。GPT3最显著的特点就是大。大体现在两方面，一方面是模型本身规模大，参数量众多，具有96层Transformer Decoder Layer，每一层有96个128维的注意力头，单词嵌入的维度也达到了12288；另一方面是训练过程中使用到的数据集规模大，达到了45TB。在这样的模型规模与数据量的情况下，GPT3在多个任务上均展现出了非常优异的性能，延续GPT2将无监督模型应用到有监督任务的思想，GPT3在Few-shot, One-shot和Zero-shot等设置下的任务表现都得到了显著的提升。
5. 【Instruction GPT模型】：2022年2月底，OpenAI发布论文《Training language models to follow instructions with human feedback》，该论文公布了InstructionGPT，它跟GPT3的网络结构是一样的，区别在于训练阶段的不同，instructGPT使用了标注数据进行fine-tune。
6. 【ChatGPT模型】：2022年11月30日，OpenAI推出ChatGPT，全网火爆。

总结：GPT系列模型架构相同，主要差异点在于数据的size和quality：



GPT3从45TB数据里面只用了570GB，约1.2%，由此可见数据预处理过滤之厉害程度！[手动狗头]

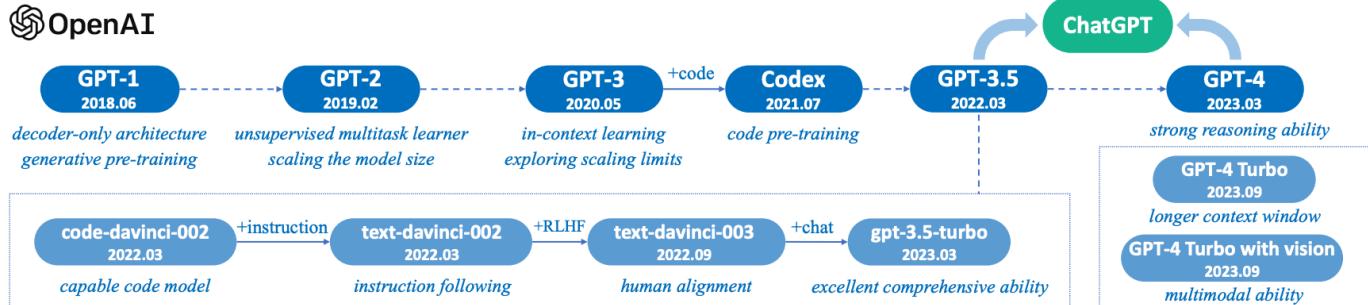


Fig. 4: A brief illustration for the technical evolution of GPT-series models. We plot this figure mainly based on the papers, blog articles and official APIs from OpenAI. Here, solid lines denote that there exists an explicit evidence (e.g., the official statement that a new model is developed based on a base model) on the evolution path between two models, while dashed lines denote a relatively weaker evolution relation.

基于Transformer的预训练模型参数量对比如下：

表 2.1: 基于 Transformer 的预训练模型对比

模型	架构	参数量	数据集	机构
BERT	Enc	Base = 110M, Large = 340M	Wikipedia, BookCorpus	Google
ALBERT	Enc	Base = 12M, Large = 18M, XLarge = 60M	Wikipedia, BookCorpus	Google
RoBERTa	Enc	356M	Wikipedia, BookCorpus	Meta/华盛顿大学
GPT-1	Dec	117M	BookCorpus	OpenAI
GPT-2	Dec	1542M	WebText	OpenAI
GPT-3	Dec	175B	Common Crawl, WebText2, Books1, Books2 and Wikipedia	OpenAI
BART	Enc-Dec	400M	English Wikipedia, BookCorpus	Meta
T5	Enc-Dec	11B	C4	Google
Switch Transformers	Enc-Dec	1.6T	C4	Google

CSDN @zh515858237

3. LLM构建流程

1.4 大语言模型构建流程



CSDN @zh515858237

(1) 预训练阶段: 使用超大规模文本对模型进行训练, 训练任务为“预测下一个token”, 训练的数据量往往需要几万亿token。

(2) 对齐阶段:

2.1 指令微调: 使用指令数据, 让模型的输出格式与人类对齐, 使其具备chat的能力。从5.4w人工标注的指令集中抽取1.3w, 在GPT-3大模型上微调。也就是说从测试用户提交的 prompt 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, 然后用这些人工标注好的 < prompt, answer > 数据来 Fine-tune GPT 3.5 模型, 从而让 GPT 3.5 初步具备理解指令中蕴含的意图的能力;

2.2 奖励函数: 基于新模型生成一批数据集<prompt,response>, 重组为3.3w排序对形式, 人工标注后, 用于训练奖励模型。奖励模型结构同基座LLM, 论文里全部用6B级别, 规模大了反而不好。也就是说, 随机抽样一批用户提交的 prompt, 然后使用第一阶段 Fine-tune 好的冷启动模型为每个 prompt 生成 K 个不同的回答, 再让标注人员对 K 个结果进行排序, 以此作为训练数据, 通过 pair-wise learning to rank 模式来训练reward model;

2.3 强化学习: 使用人类反馈或者偏好数据来训练模型, 使模型的输出更加符合人类的价值观或者预期行为。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。RLHF的具体实现, RM奖励模型作为critic (评论家), SFT阶段的大模型作为actor (行动家), 二者相互配合, actor学习指令集, critic评估打分, 再更新权重, 进入下一轮。论文里对比两种损失函数, 后采用混合预训练损失PPT_ptx, 兼顾预训练的效果。

4. LLM主流架构

The Practical Guides for Large Language Models按照模型结构整理了大模型的进化树。现有 LLMs 的主流架构大致可以分为三大类, 即Encoder-Decoder、Causal Decoder、Prefix Decoder:

TABLE 5: Model cards of several selected LLMs with public configuration details. Here, PE denotes position embedding, #L denotes the number of layers, #H denotes the number of attention heads, d_{model} denotes the size of hidden states, and MCL denotes the maximum context length during training.

Model	Category	Size	Normalization	PE	Activation	Bias	#L	#H	d_{model}	MCL
GPT3 [55]	Causal decoder	175B	Pre LayerNorm	Learned	GeLU	✓	96	96	12288	2048
PanGu- α [84]	Causal decoder	207B	Pre LayerNorm	Learned	GeLU	✓	64	128	16384	1024
OPT [90]	Causal decoder	175B	Pre LayerNorm	Learned	ReLU	✓	96	96	12288	2048
PaLM [56]	Causal decoder	540B	Pre LayerNorm	RoPE	SwiGLU	✗	118	48	18432	2048
BLOOM [78]	Causal decoder	176B	Pre LayerNorm	ALiBi	GeLU	✓	70	112	14336	2048
MT-NLG [113]	Causal decoder	530B	-	-	-	-	105	128	20480	2048
Gopher [64]	Causal decoder	280B	Pre RMSNorm	Relative	-	-	80	128	16384	2048
Chinchilla [34]	Causal decoder	70B	Pre RMSNorm	Relative	-	-	80	64	8192	-
Galactica [35]	Causal decoder	120B	Pre LayerNorm	Learned	GeLU	✗	96	80	10240	2048
LaMDA [68]	Causal decoder	137B	-	Relative	GeLU	-	64	128	8192	-
Jurassic-1 [107]	Causal decoder	178B	Pre LayerNorm	Learned	GeLU	✓	76	96	13824	2048
LLaMA [57]	Causal decoder	65B	Pre RMSNorm	RoPE	SwiGLU	✗	80	64	8192	2048
LLaMA 2 [99]	Causal decoder	70B	Pre RMSNorm	RePE	SwiGLU	✗	80	64	8192	4096
Falcon [141]	Causal decoder	40B	Pre LayerNorm	RoPE	GeLU	✗	60	64	8192	2048
GLM-130B [93]	Prefix decoder	130B	Post DeepNorm	RoPE	GeGLU	✓	70	96	12288	2048
T5 [82]	Encoder-decoder	11B	Pre RMSNorm	Relative	ReLU	✗	24	128	1024	512

CSDN @hit56实验室

Encoder-Only结构：BERT系列

Decoder-Only结构：GPT系列，成员最多

Encoder-Decoder结构：BART、T5

5. LLM评估指标

测试集名称	排行榜	语言	备注
MMLU (Massive Multitask Language Understanding)	排行榜	英文	这是最值得注意的测试集，它考虑了 57 个学科，从人文到社科到理工多个大类的综合知识能力。DeepMind 的 Gopher 和 Chinchilla 这两个模型甚至只看 MMLU 的分数。
C-Eval (A Multi-Level Multi-Discipline Chinese Evaluation Suite for Foundation Models)	排行榜	中文	覆盖人文，社科，理工，其他专业四个大方向，52 个学科（微积分，线代 ...），从中学到大学研究生以及职业考试，一共 13948 道题目的中文知识和推理型测试集，管它叫 C-Eval，来帮助中文社区研发大模型。是上海交大和清华联合研发的中文大语言模型测试集。
GSM8k (Grade School Math 8K)	排行榜	数学	是由 OpenAI 发布的一个由 8.5K 高质量的语言多样化的小学数学应用题组成的数据集，要求根据给定的场景和两个可能的解决方案，选择最合理的方案。
BBH (BIG-Bench Hard)		英文	BBH是一个挑战性任务 Big-Bench 的子集。Big-Bench 目前包括 204 项任务。任务主题涉及语言学、儿童发展、数学、常识推理、生物学、物理学、社会偏见、软件开发等方面。BBH 是从 204 项 Big-Bench 评测基准任务中大模型表现不好的任务单独拿出来形成的评测基准。

6. LLM排行榜

国内排行榜：[opencompass](#)（上海AI实验室推出的）国外排行榜：[chatbot-arena-leaderboard](#)

从下面不同的LLMs预训练数据源分布，可以看出：垂直领域的模型之所以有价值，是因为有了垂直领域的数据：

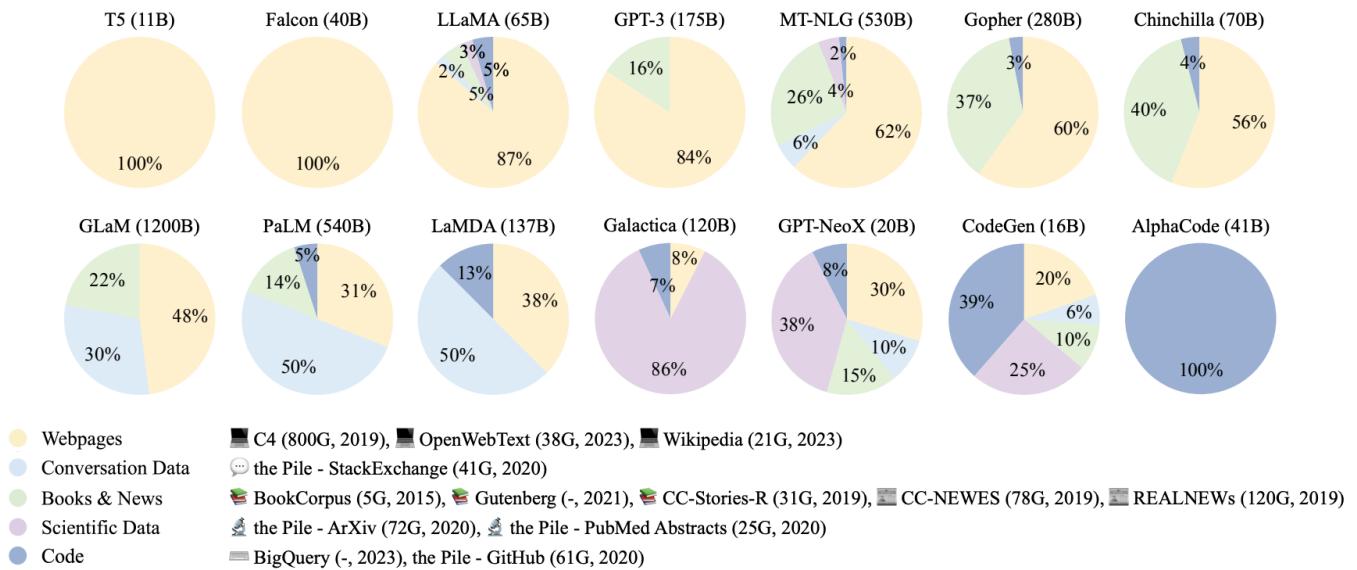


Fig. 6: Ratios of various data sources in the pre-training data for existing LLMs.

CSDN @hit56实验室

7. LLaMA的网络结构

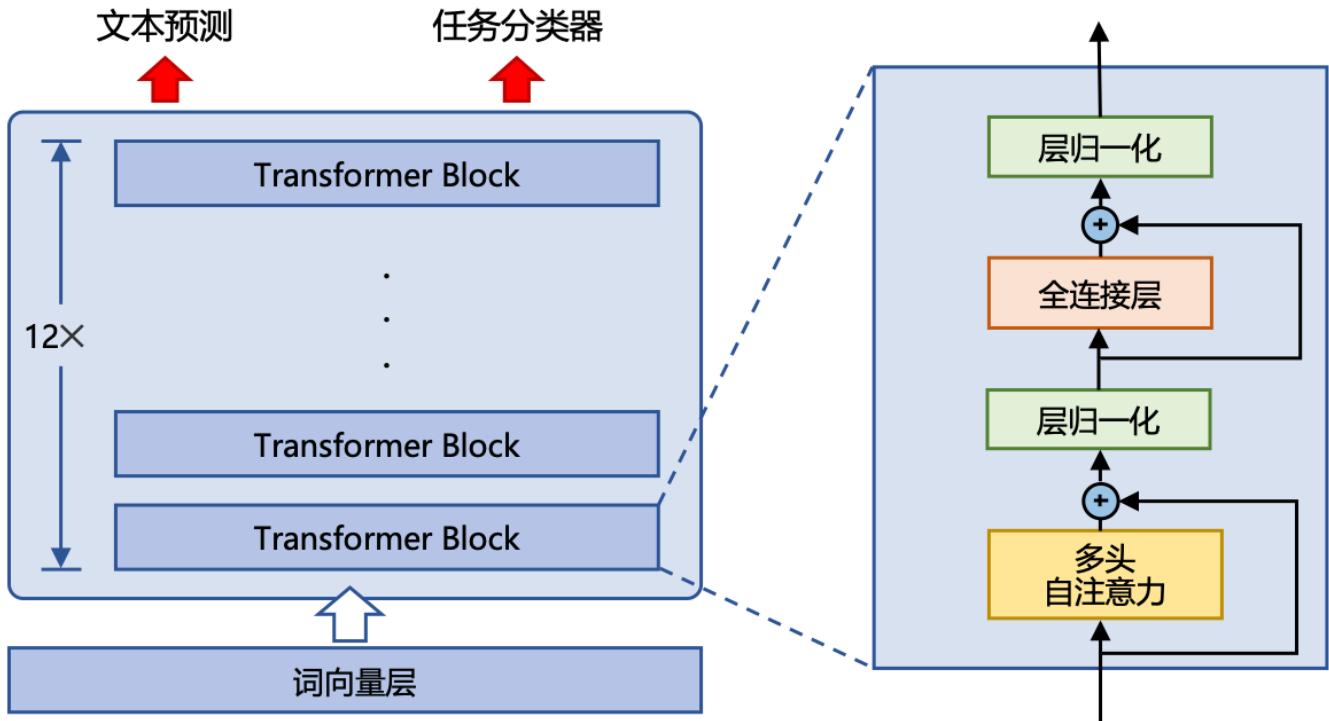


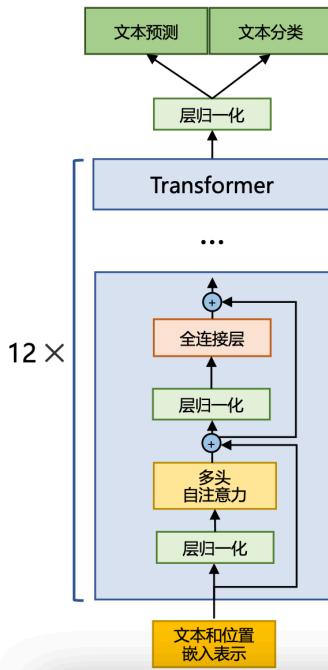
图2.3 GPT 的模型结构

CSDN @hit56工作室

当前，绝大多数大语言模型都采用了类似GPT的架构，使用基于Transformer架构构建的仅由解码器组成的网络结构，采用自回归的方式构建语言模型，但是在层归一化

位置、归一化函数、激活函数、位置编码等细节上各有不同。LLaMA的模型结构如下：

2.3.1 LLaMA 的模型结构



文献[37] 介绍了LLaMA 采用的Transformer 结构和细节，与2.1 节介绍的Transformer 结构的不同之处为：

- **前置层归一化 (Pre-normalization)**
- RMSNorm 归一化函数 (Normalizing Function)
- SwiGLU激活函数
- 旋转位置嵌入 (Rotary Positional Embeddings , RoPE)

使用的Transformer 结构与GPT-2 类似，如图2.4 所示。



8. 强化学习与有监督学习的区别

(1) 强化学习相较于有监督学习更有可能考虑整体影响

有监督学习针对单个词元进行反馈，其目标是要求模型针对给定的输入给出确切的答案。而强化学习是针对整个输出文本进行反馈，并不针对特定的词元。反馈粒度的不同，使强化学习更适合大语言模型，既可以兼顾表达多样性，又可以增强对微小变化的敏感性。自然语言十分灵活，可以用多种不同的方式表达相同的语义。而有监督学习很难支持上述学习方式。强化学习则可以允许模型给出不同的多样性表达。

另外，有监督微调通常采用交叉熵损失作为损失函数，由于总和规则，造成这种损失对个别词元变化不敏感。如果改变个别的词元，只会对整体损失产生小的影响。但是，一个否定词可以完全改变文本的整体含义。强化学习则可以通过奖励函数同时兼顾多样性和微小变化敏感性两个方面。

(2) 强化学习更容易解决幻觉问题

用户在大语言模型上主要有三类输入：(a) 文本型 (Text-Grounded)，用户输入相关文本和问题，让模型基于所提供的文本生成答案 (例如，“本文中提到的人名和地名有哪些”)；(b) 求知型 (Knowledge-Seeking)，用户仅提出问题，模型根据内在知识提供真实回答 (例如，“流感的常见原因是什么”)；(c) 创造型 (Creative)，用户提供问题或说明，让模型进行创造性输出 (例如，“写一个关于……的故事”)。有监督学习算法非常容易使得求知型查询产生幻觉。在模型并不包含或者知道答案的情况下，有监督训练仍然会促使模型给出答案。而使用强化学习方法，则可以通过定制奖励函数，将正确答案赋予非常高的分数，将放弃回答的答案赋予中低分数，将不正确的答案赋予非常高的负分，使得模型学会依赖内部知识选择放弃回答，从而在一定程度上缓解模型的幻觉问题。

(3) 强化学习可以更好地解决多轮对话奖励累积问题

多轮对话能力是大语言模型重要的基础能力之一。多轮对话是否达成最终目标，需要考虑多次交互过程的整体情况，因此很难使用有监督学习的方法构建。而使用强化学习方法，可以通过构建奖励函数，根据整个对话的背景及连贯性对当前模型输出的优劣进行判断。

9. LLM从海量文本中学习到了什么

1. 学到了“语言类知识”

LLM可以学习各种层次的语言学知识：浅层语言知识比如词法、词性、句法等知识存储在Transformer的低层和中层，而抽象的语言知识比如语义类知识，广泛分布在Transformer的中层和高层结构中。

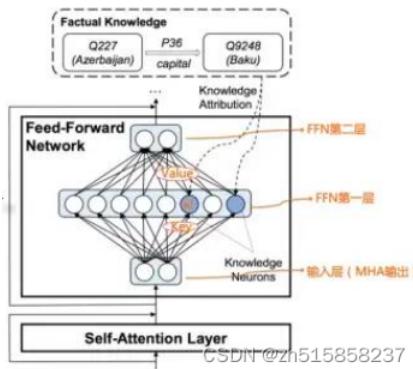
2. 学到了“世界知识”

世界知识指的是在这个世界上发生的一些真实事件 (事实型知识, Factual Knowledge)，以及一些常识性知识(Common Sense Knowledge)。比如“拜登是现任美国总统”、“拜登是美国人”、“乌克兰总统泽连斯基与美国总统拜登举行会晤”，这些都是和拜登相关的事实类知识；而“人有两只眼睛”、“太阳从东方升起”这些属于常识性知识。LLM确实从训练数据中吸收了大量世界知识，而这类知识主要分布在Transformer的中层和高层，尤其聚集在中层。而且，随着Transformer模型层深增加，能够学到的知识数量逐渐以指数级增加 (可参考：BERTnesia: Investigating the capture and forgetting of knowledge in BERT)。LLM可以看作是一种以模型参数体现的隐式知识图谱。

《When Do You Need Billions of Words of Pre-training Data?》这篇文章研究了预训练模型学到的知识量与训练数据量的关系。它的结论是：对于Bert类型的语言模型来说，由于语言学知识相对有限且静态，只需1000万到1亿单词的语料，就能学好句法语义等语言学知识，但是如果要学习事实类知识，由于事实类知识则数量巨大，且处于不断变化过程中，则要更多的训练数据。随着训练数据量的增加，预训练模型在各种下游任务中效果越好，这说明从增量的训练数据中学到的主要是世界知识。

10. LLM的知识存储在网络的什么地方

比如“中国的首都是北京”这条知识，以三元组表达就是<北京, is-capital-of, 中国>，其中“is-capital-of”代表实体间关系。这条知识它存储在LLM的哪里呢？



《Transformer Feed-Forward Layers Are Key-Value Memories》给出了一个比较新颖的观察视角，它把Transformer的FFN看成存储大量具体知识的Key-Value存储器。FFN的第一层是个MLP宽隐层，这是Key层；第二层是MLP窄隐层，是Value层。FFN的输入层其实是某个单词对应的MHA的输出结果Embedding，也就是通过Self Attention，将整个句子有关的输入上下文集成到一起的Embedding，代表了整个输入句子的整体信息。

我们假设上图的节点就是记载<北京, is-capital-of, 中国>这条知识的Key-Value存储器，它的Key向量，用于检测“中国的首都是...”这个知识模式，它的Value向量，基本存储了与单词“北京”的Embedding比较接近的向量。当Transformer的输入是“中国的首都是[Mask]”的时候，节点从输入层探测到这个知识模式，所以产生较大的响应输出。我们假设Key层其它神经元对这个输入都没有任何响应，那么对应的Value层的节点，其实只会接收到“北京”这个Value对应的单词embedding，并通过的大响应值，进行了进一步的数值放大。于是，Mask位置对应的输出，就自然会输出“北京”这个单词。

11. 如何修改LLM中的某些知识

既然我们已知具体的某条世界知识存储在某个或者某些FFN节点的参数里，自然会引发另外一个问题：我们能否修正LLM模型里存储的错误或者过时的知识呢？比如对于问题：“英国的现任首相是谁？”鉴于近年来英国首相频繁更迭，你猜LLM更倾向输出“鲍里斯”还是更青睐“苏纳克”？很明显训练数据中包含“鲍里斯”的数据会更多，这种情况很可能LLM会给出错误回答，于是我们就有修正LLM里存储的过时知识的必要性。

目前有三类不同方法来修正LLM里蕴含的知识：

1. 从训练数据的源头来修正知识。《Towards Tracing Factual Knowledge in Language Models Back to the Training Data》这篇文章的研究目标是：对于指定的某条知识，我们是否可以定位到是哪些训练数据导致LLM学会了这条知识？答案是肯定的，这意味着我们可以逆向追踪到某条知识对应的训练数据源头。如果利用这项技术，假设我们想要删除某条知识，则可首先定位到其对应的数据源头，删除数据源，然后重新预训练整个LLM模型，这样即可达成删除LLM中相关知识的目的。但是这里有个问题，如果修正一小部分知识，我们就需要重新做一次模型预训练，这样做明显成本太高。所以这种方法不会太有发展前景，可能比较适合那种对于某个特定类别数据的一次性大规模删除场合，不适合少量多次的常规知识修正场景，比如可能比较适合用来做去除偏见等去toxic内容的处理。
2. 对LLM模型做一次fine-tuning来修正知识。一个直观能想到的方法是：我们可以根据要修正的新知识来构建训练数据，然后让LLM模型在这个训练数据上做fine-tuning，这样指导LLM记住新的知识，遗忘旧的知识。这个方法简单直观，但是也有一些问题，首先它会带来灾难遗忘问题，就是说除了忘掉该忘的知识，还忘掉了不该忘的知识，导致这么做了之后有些下游任务效果下降。另外，因为目前的LLM模型规模非常大，即使是做fine-tuning，如果次数频繁，其实成本也相当高。对这种方法感兴趣的可以参考《Modifying Memories in Transformer Models》。
3. 直接修改LLM里某些知识对应的模型参数来修正知识。假设我们想要把旧知识<英国, 现任首相, 鲍里斯>，修正到<英国, 现任首相, 苏纳克>。首先我们想办法在LLM模型参数中，定位到存储旧知识的FFN节点，然后可以强行调整更改FFN中对应的模型参数，将旧知识替换成新的知识。可以看出，这种方法涉及到两项关键技术：首先是如何在LLM参数空间中定位某条知识的具体存储位置；其次是如何修正模型参数，来实现旧知识到新知识的修正。关于这类技术的细节，可以参考《Locating and Editing Factual Associations in GPT》和《Mass-Editing Memory in a Transformer》。理解这个修正LLM知识的过程，其实对于更深入理解LLM的内部运作机制是很有帮助的。

12. 如何消除模型幻觉

首先，为什么会产生幻觉？这主要原因是用于训练语言模型的大数据语料库在收集时难免会包含一些错误的信息，这些错误知识都会被学习，存储在模型参数中，相关研究表明模型生成文本时会优先考虑自身参数化的知识，所以更倾向生成幻觉内容。

因此有以下几个可能的手段来减少幻觉：

1. **数据阶段：** 使用置信度更高的数据，消除原始数据中本来的错误和不一致地方。
2. **训练阶段：** 有监督学习算法非常容易使得求知型查询产生幻觉。在模型并不包含或者知道答案的情况下，有监督训练仍然会促使模型给出答案。而使用强化学习方法，则可以通过定制奖励函数，将正确答案赋予非常的分数，将放弃回答的答案赋予中低分数，将不正确的答案赋予非常的负分，使得模型学会依赖内部知识选择放弃回答，从而在一定程度上缓解模型的幻觉问题。
3. **基于后处理：** 在生成输出后，对其进行迭代评估和调整。对于摘要等任务，只有在生成整个摘要后才能准确评估，因此后期修正方法更为有效。缺点是不改变模型本身。
4. **基于知识检索增强的方式：** 模型幻觉很大程度来源之一是外部知识选择不正确，因此用更强的检索模型搜索知识，返回更加有用的知识，也是消除对话回复幻觉的有效途径。

13. 如何平衡训练数据量、模型参数、增加epoch的关系

研究证明：当我们独立增加训练数据量、模型参数规模或者延长模型训练时间（比如从1个Epoch到2个Epoch），预训练模型在测试集上的Loss都会单调降低，也就是说模型效果越来越好。

既然三个因素都重要，那么我们在实际做预训练的时候，就有一个算力如何分配的决策问题：假设用于训练LLM的算力总预算（比如多少GPU小时或者GPU天）给定，那么是应该多增加数据量、减少模型参数呢？还是说数据量和模型规模同时增加，减少训练步数呢？此消彼长，某个要素规模增长，就要降低其它因素的规模，以维持总算力不变，所以这里有各种可能的算力分配方案。最终OpenAI选择了同时增加训练数据量和模型参数，但是采用早停策略(early stopping)来减少训练步数的方案。因为它证明

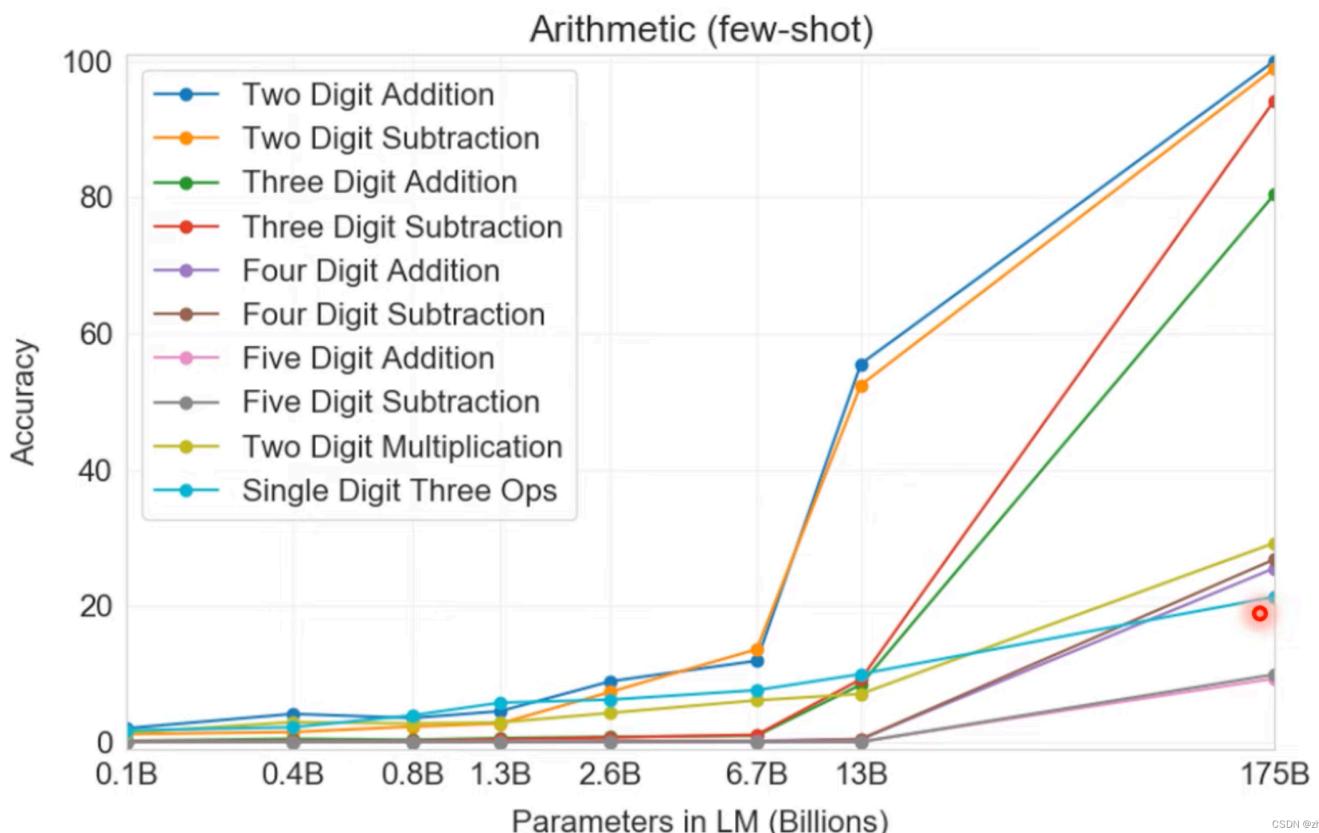
了：对于训练数据量和模型参数这两个要素，如果只单独增加其中某一个，这不是最好的选择，最好能按照一定比例同时增加两者，它的结论是优先增加模型参数，然后才是训练数据量。假设用于训练LLM的算力总预算增加了**10倍**，那么应该增加**5.5倍**的模型参数量，**1.8倍**的训练数据量，此时模型效果最佳。

14. ChatGPT的优缺点与最大贡献

1. 优点：有很强的归纳能力和泛化记忆能力
2. 缺点：在**符号推理、输出可控和可解释**方面还较弱，并且容易犯事实性错误。
3. 最大贡献：基本实现了理想LLM的接口层，让LLM适配人的习惯命令表达方式，而不是反过来让人去适配LLM，绞尽脑汁地想出一个能Work的命令（这就是instruct技术出来之前，prompt技术在做的事情），而这增加了LLM的易用性和用户体验。是InstructGPT/ChatGPT首先意识到这个问题，并给出了很好的解决方案，这也是它最大的技术贡献。相对之前的few shot prompting，它是一种更符合人类表达习惯的人和LLM进行交互的人机接口技术。

15. GPT3在算数方面的能力

Context →	Q: What is 17 minus 14?
A:	
Target Completion →	3



16. 思维链prompting是什么

思维链 (few-shot CoT,Chain of Thought) Prompting，这个方向目前是LLM推理研究的主方向。

Standard Prompting	Chain of Thought Prompting
<p>Input</p> <p>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?</p> <p>A: The answer is 11.</p> <p>Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?</p>	<p>Input</p> <p>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?</p> <p>A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.</p> <p>Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?</p>
<p>Model Output</p> <p>A: The answer is 27. ✗</p>	<p>Model Output</p> <p>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. ✓</p>

From:Chain of thought prompting elicits reasoning in large language models @zh515858237

CoT的主体思想其实很直白；为了教会LLM模型学会推理，给出一些人工写好的推理示例，示例里把得到最终答案前，一步步的具体推理步骤说清楚，而这些人工写的详细推理过程，就是思维链Prompting，具体例子可参照上图中蓝色文字部分。CoT的意思是让LLM模型明白一个道理；就是在推理过程中，步子不要迈得太大，否则很容易出

错，改变思维模式，化大问题为小问题，步步为营，积小胜为大胜。最早明确提出CoT这个概念的文章是“Chain of thought prompting elicits reasoning in large language models”，论文发布于22年1月份，虽然做法很简单，但是应用CoT后LLM模型的推理能力得到了巨大提升，GSM8K数学推理测试集准确率提高到60.1%左右。

17. chatGPT中的语言模型跟传统语言模型最大区别

目标不一样。传统语言模型主要是预测一句话中下一个词是什么。而instructGPT的目标是：Follow the user's instructions helpfully, honestly, and harmlessly..

18. 代码预训练可以增强LLM推理能力

以“Self Consistency”方法为例，在大多数据集合上的性能提升，都直接超过了20到50个百分点，这是很恐怖的性能提升，而其实在具体推理模型层面，我们什么也没做，仅仅是预训练的时候除了文本，额外加入了程序代码而已。

越大的LLM模型学习效率越高，也就是说相同训练数据量，模型越大任务效果越好，说明面对的即使是同样的一批训练数据，更大的LLM模型相对规模小一些的模型，从中学到了更多的知识。

当模型参数规模未能达到某个阈值时，模型基本不具备解决此类任务的任何能力，体现为其性能和随机选择答案效果相当，但是当模型规模跨过阈值，LLM模型对此类任务的效果就出现突然的性能增长。也就是说，模型规模是解锁(unlock)LLM新能力的关键，随着模型规模越来越大，会逐渐解锁LLM越来越多的新能力。思维链 (Chain of Thought) Prompting是典型的增强LLM推理能力的技术，能大幅提升此类任务的效果。

问题是，为何LLM会出现这种“涌现能力”现象呢？上述文章以及《Emergent Abilities of Large Language Models》给出了几个可能的解释：

一种可能解释是有些任务的评价指标不够平滑。比如说有些生成任务的判断标准，它要求模型输出的字符串，要和标准答案完全匹配才算对，否则就是0分。所以，即使随着模型增大，其效果在逐步变好，体现为输出了更多的正确字符片段，但是因为没有完全对，只要有任何小错误都给0分，只有当模型足够大，输出片段全部正确才能得分。也就是说，因为指标不够平滑，所以不能体现LLM其实正在逐步改善任务效果这一现实，看起来就是“涌现能力”这种外在表现。

另外一种可能的解释是：有些任务由若干中间步骤构成，随着模型规模增大，解决每个步骤的能力也在逐步增强，但是只要有一个中间步骤是错的，最终答案就是错的，于是也会导致这种表面的“涌现能力”现象。

如果你细想，会发现In Context Learning是个很神奇的技术。它神奇在哪里呢？神奇在你提供给LLM几个样本示例 $\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \dots, \langle x_n, y_n \rangle$ ，然后给它 x_m ，LLM竟然能够成功预测对应的 y_m 。听到这你会反问：这有什么神奇的呢？Fine-tuning不就是这样工作的吗？你要这么问的话，说明你对这个问题想得还不够深入。

Fine-tuning和In Context Learning表面看似都提供了一些例子给LLM，但两者有质的不同（参考上图示意）：Fine-tuning拿这些例子当作训练数据，利用反向传播去修正LLM的模型参数，而修正模型参数这个动作，确实体现了LLM从这些例子学习的过程。但是，In Context Learning只是拿出例子让LLM看了一眼，并没有根据例子，用反向传播去修正LLM模型参数的动作，就要求它去预测新例子。既然没有修正模型参数，这意味着貌似LLM并未经历一个学习过程，如果没有经历学习过程，那它为何能够做到仅看一眼，就能预测对新例子呢？这正是In Context Learning的神奇之处。

能够有效增加LLM模型Instruct泛化能力的因素包括：增加多任务的任务数量、增加LLM模型大小、提供CoT Prompting，以及增加任务的多样性。如果采取任意一项措施，都可以增加LLM模型的Instruct理解能力。

当模型规模足够大的时候，LLM本身是具备推理能力的，在简单推理问题上，LLM已经达到了很好的能力，但是复杂推理问题上，还需要更多深入的研究。

如果梳理现有LLM推理相关工作的话，我把它们归到两大类，体现出挖掘或促进LLM推理能力不同的技术思路：第一类研究比较多，可以统称为基于Prompt的方法，核心思想是通过合适的提示语或提示样本，更好地激发出LLM本身就具备的推理能力，Google在这个方向做了大量很有成效的工作。第二类做法是在预训练过程中引入程序代码，和文本一起参与预训练，以此进一步增强LLM的推理能力，这应该是OpenAI实践出的思路。比如ChatGPT肯定具备很强的推理能力，但它并不要求用户必须提供一些推理示例，所以ChatGPT强大的推理能力，大概率来源于使用代码参与GPT 3.5的预训练。

这两种思路其实大方向是迥异的：利用代码增强LLM推理能力，这体现出一种通过增加多样性的训练数据，来直接增强LLM推理能力的思路；而基于Prompt的方法，它并不会促进LLM本身的推理能力，只是让LLM在解决问题过程中更好地展示出这种能力的技术方法。

19. 大模型智能涌现的参数临界点

当模型参数超过 50B 时，往往会产生质变，出现一种涌现能力

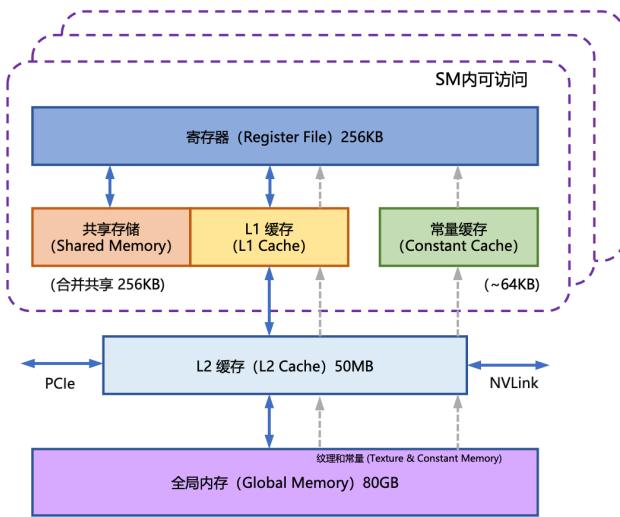
为何LLM会出现这种“涌现能力”现象呢？《Emergent Abilities of Large Language Models》给出了几个可能的解释：

1. 一种可能解释是有些任务的评价指标不够平滑。比如说有些生成任务的判断标准，它要求模型输出的字符串，要和标准答案完全匹配才算对，否则就是0分。所以，即使随着模型增大，其效果在逐步变好，体现为输出了更多的正确字符片段，但是因为没有完全对，只要有任何小错误都给0分，只有当模型足够大，输出片段全部正确才能得分。也就是说，因为指标不够平滑，所以不能体现LLM其实正在逐步改善任务效果这一现实，看起来就是“涌现能力”这种外在表现。

2. 另外一种可能的解释是：有些任务由若干中间步骤构成，随着模型规模增大，解决每个步骤的能力也在逐步增强，但是只要有一个中间步骤是错的，最终答案就是错的，于是也会导致这种表面的“涌现能力”现象。

20. flash-attention为什么能加速

2. FlashAttention



全局内存和本地内存使用的高带宽显存 (High Bandwidth Memory, HBM) 位于板卡RAM存储芯片上，该部分内存容量很大。全局内存是所有线程都可以访问，而本地内存则只能当前线程访问。NVIDIA H100中全局内存有80GB空间，其访问速度虽然可以达到3.35TB/s，但是如果全部线程同时访问全局内存时，其平均带宽仍然很低。共享内存和寄存器位于GPU芯片上，因此容量很小，并且共享内存只有在同一个GPU线程块 (Thread Block) 内的线程才可以共享访问，而寄存器仅限于同一个线程内部才能访问。NVIDIA H100中每个GPU线程块在流式多处理器 (Stream Multi-processor, SM) 可以使用的共享存储容量仅有228KB，但是其速度非常快，远高于全局内存的访问速度。

根据自注意力机制的原理，在GPU中进行计算时，传统的方法还需要引入两个中间矩阵 S 和 P 并存储到全局内存中。具体计算过程如下：

$$S = Q \times K, \quad P = \text{Softmax}(S), \quad O = P \times V$$

CSDN @hit56工作室

按照上述计算过程，需要首先从全局内存中读取矩阵 Q 和 K，并将计算好的矩阵 S 再写入全局内存，之后再从全局内存中获取矩阵 S，计算 Softmax 得到矩阵 P 再写入全局内存，之后读取矩阵 P 和矩阵 V，计算得到矩阵 O。

这样的过程会极大占用显存的带宽。在自注意力机制中，计算速度比内存速度快得多，因此计算效率越来越多地受到全局内存访问的瓶颈。

FlashAttention就是通过利用GPU硬件中的特殊设计，针对全局内存和共享存储的I/O速度的不同，尽可能地避免HBM中读取或写入注意力矩阵。

FlashAttention目标是尽可能高效地使用SRAM来加快计算速度，避免从全局内存中读取和写入注意力矩阵。达成该目标需要能做到在不访问整个输入的情况下计算Softmax函数，并且后向传播中不能存储中间注意力矩阵。

FlashAttention 就提出了不使用中间注意力矩阵，通过存储归一化因子来减少全局内存消耗的方法。

FlashAttention 算法并没有将 S、P 整体写入全局内存，而是通过分块写入，存储前向传递的 Softmax 归一化因子，在后向传播中快速重新计算片上注意力，这比从全局内存中读取中间注意力矩阵的标准方法更快。

虽然大幅减少了全局内存的访问量，重新计算也导致FLOP 增加，但其运行的速度更快且使用的内存更少。

21. 什么是bpe算法

字节对编码 (Byte Pair Encoding, BPE) 模型采用的词表包含最常见的单词及高频出现的子词。在使用时，常见词通常本身位于BPE词表中，而罕见词通常能被分解为若干个包含在BPE词表中的词元，从而大幅降低未登录词的比例。BPE算法包括两个部分：

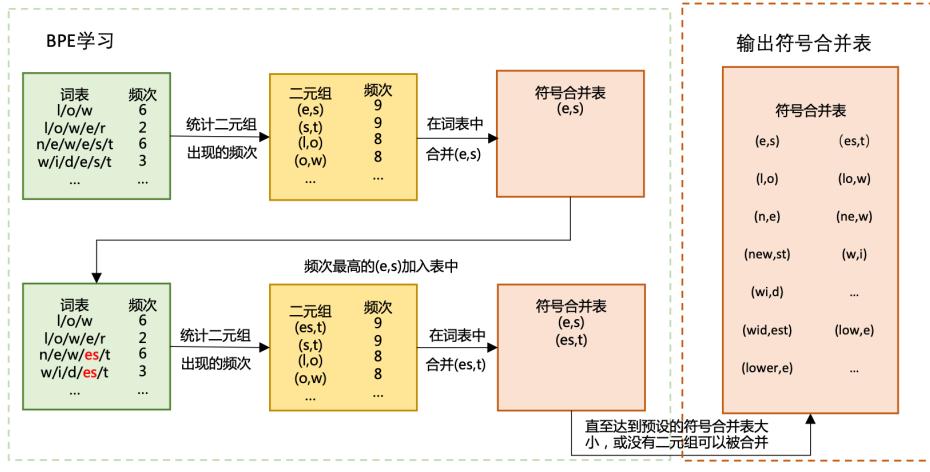
- (1) 词元词表的确定。
- (2) 全词切分为词元以及词元合并为全词的方法。

NVIDIA GPU中的内存（显存）按照它们物理上是在GPU芯片内部还是板卡RAM存储芯片上，决定了它们的速度、大小以及访问限制。GPU显存分为：

- 全局内存 (Global memory)
- 本地内存 (Local memory)
- 共享内存 (Shared memory , SRAM)
- 寄存器内存 (Register memory)
- 常量内存 (Constant memory)
- 纹理内存 (Texture memory)

CSDN @hit56工作室

BPE 模型中词元词表的计算过程



首先，确定语料库中全词的词表和词频，然后将每个单词切分为单个字符的序列，并在序列最后添加符号。</W>作为单词结尾的标识。所划分出的序列元素称为字节，即每个单词都切分为字节的序列。

之后，按照每个字节序列的相邻字节对和单词的词频，统计每个相邻字节对的出现频率，合并出现频率最高的字节对，将其作为新的词元加入词表，并将全部单词中的该字节对合并为新的单一字符。重复这一步骤，直至BPE词元词表的大小达到指定的预设值，或没有可合并的字节对为止。

CSDN UNIVERSITY

在词元词表确定之后，对输入词序列中未在词表中的全词进行切分，BPE方法对词表中的词元按从长到短的顺序进行遍历，用每一个词元和当前序列中的全词或未完全切分为词元的部分进行匹配，将其切分为该词元和剩余部分的序列。

例如，对于单词“lowest”

首先通过匹配词元“est”将其切分为“low”“est”的序列

再通过匹配词元“low”

确定其最终切分结果为“low”“est”的序列。

通过这样的过程，使用BPE尽量将词序列中的词切分成已知的词元。

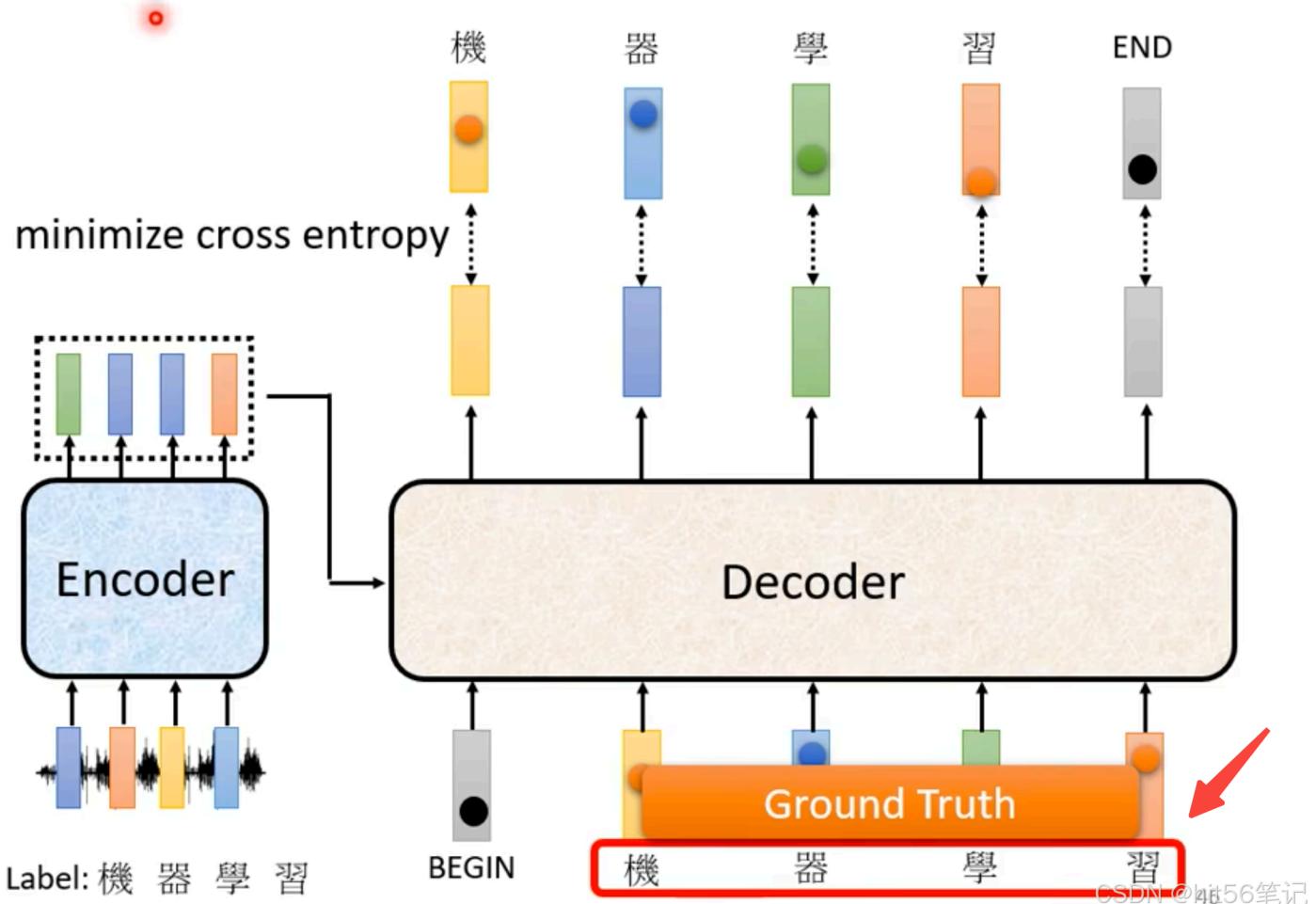
在遍历词元词表后，对于切分得到的词元序列，为每个词元查询词元表示，构成词元表示序列。若出现未登录词元，即未出现在BPE词表中的词元，则采取和未登录词类似的方式，为其赋予相同的表示，最终获得输入的词元表示序列。

字节级 (Byte-level) BPE 通过将字节视为合并的基本符号，改善多语言语料库（例如包含非ASCII字符的文本）的分词质量。GPT-2、BART、LLaMA 等大语言模型都采用了这种分词方法。原始LLaMA的词表大小是32K，并且主要根据英文进行训练，因此，很多汉字都没有直接出现在词表中，需要字节来支持所有的中文字符，由2个或者3个Byte Token 才能拼成一个完整的汉字。

对于使用了BPE的大语言模型，其输出序列也是词元序列。对于原始输出，根据终结符的位置确定每个单词的范围，合并范围内的词元，将输出重新组合为词序列，作为最终的结果。

22. Decoder在训练时，输入是真实标注label，但实际使用时输入的是上一次的解码结果

Teacher Forcing: using the ground truth as input.



23. gpt4距离超越人类智力还有多远

1. 人类大脑有100万亿以上的神经元连接

人类大脑有860亿神经元、100-1000万亿连接，能处理的任务也远远超过GPT-3。如果一个联接就相当于有一个参数，那么粗略估计人脑可能可以通过100-1000万亿的模型参数来模拟。GPT-4拥有超过一万亿个参数，可能在一万亿到1.76万亿之间，这使得它比其前身GPT-3（1750亿参数）大得多。那么gpt4参数规模还需要再扩大100倍至1000倍，才可能超过人类的智力。

当然，这并不意味着只要参数规模扩大这么多倍，就一定能实现完全的人类智能。

2. 人类历史上也曾出现类似chatGPT的病人，只有某个时间点以前的记忆

1953年，Henry Molaison 因为饱受癫痫症带来的痛苦，接受了一种实验性的手术。这次手术让他成为神经科学家们最耳熟能详的病人。

手术切除了他部分内侧颞叶，包括海马体，这些区域在当时被认为与癫痫发作有关。术后，身边的医护人员发现 Henry 的记忆就像沙滩上的字，时间的海水一冲就会消失，无法形成新的记忆；同时，他对以前发生的事情、语言中每个词语的意义、理解和发音记得一清二楚。也就是说，海马体的缺失使他的记忆永远停留在了手术那一天。

对于人类这种智能体，记忆似乎是与生俱来的能力；而如果把 ChatGPT 类的大语言模型比作大脑，其天然就缺失了形成记忆能力的海马体。在大模型中，世界知识和语义理解被压缩为了静态的参数，模型不会随着交互记住我们的聊天记录和喜好，也不会调用额外的知识信息来辅助自己的判断。

二、实践篇

1. 如何下载llama2模型

如何从huggingface快速下载llama2模型

2. 免费试用gpt4的网站

www.coze.com

3. prompt提示应该怎么写效果会更好

参考官方文档：[OpenAI教你如何写prompt](https://openai.com/research/)

Step-back 这种prompt技术起源于人类在面对具有挑战性的任务时，往往会退一步进行抽象，从而得出高层次的概念和原则来指导推理过程，受此启发，研究人员才提出了后退的prompt技术，将推理建立在抽象概念的基础上，从而降低在中间推理步骤中出错的几率。

例子：

- 1 | 你是一个擅长中文和英文的AI 工程师和数据科学家，擅长中文和英文相关的 AI 技术文章编写和翻译。
- 2 | 请将下面的英文翻译成中文，要求语言生动活泼，简洁优雅。

```

3
4
5 需要翻译的英文段落:
6 Spending lots of time thinking about different possible future scenarios and their probabilities might be captivating, but I suggest an absolutely
7
8
9 请按照一下步骤输出结果:
10 1. 要想得到通顺优雅简洁的翻译文章, 你需要知道哪些前提问题
11 2. 这些前提问题的答案分别是什么
12 3. 基于这个前提问题, 对于给出英文的翻译结果

```

4. 预训练数据集概览

表 4.1: 预训练数据集概览

数据集	发布者	规模	特点	支撑的语言模型
BooksCorpus	Zhu et al. (2015)、Shawn Presser	book1: 2.2GB; book3: 37GB	英文小说集	GPT 系列、OPT、OPT-IML
Wikipedia	维基媒体基金会	21.23 GB	多语言高质量 百科全书	GPT 系列、OPT、OPT-IML
Common Crawl	Common Crawl 团队	超过 PB	网页数据, 规模巨大	GPT 系列、T5、UL2、Flan-T5
ROOT	BigScience	1.6TB	包含 69 种语言	BLOOM、BLOOMZ、mT0
The Pile	Gao et al.(2020)	825GB	数据来源广泛, 多样性佳	GLM-130B、GPT-J、GPT-NeoX-20B、 OPT、OPT-IML、GLM-130B
悟道	北京智源人工智能研究院	3TB	中文数据集	GLM-130B
CLUECorpus 2020	GLUE 开源社区	100GB	中文数据集	
MNBVC MNBVC	里屋社区	2.18TB	中文数据集	

65DH@zh515858237

尽管目前已经开源了很多的预训练数据, 但在训练大规模预训练语言模型时, 预训练数据依然是瓶颈, 原因如下:(1)开源的预训练数据或多或少存在噪音问题, 特别是爬虫数据噪音问题严重, **如何对预训练数据进行高质量地清洗和去重, 是目前数据处理的核心与壁垒**;(2)OpenAI 和 Google 使用的高质量预训练数据集是闭源, 无法获得, 例如 Google 公司训练 Chinchilla 中使用的 2.1TB 的书籍数据库、3.1TB 的 Github 数据, OpenAI 公司训练 GPT 3 中使用的 WebText23、Books1、Books2 数据集。自动爬取或收集的原始数据往往存在信息量低和含有噪声的问题, 例如版本信息中只有“update”, 缺少实质内容;回答中的代码片段可能与问题无关等。因此, 在使用数据之前, 必须基于一定规则仔细进行筛选。

5. InstructGPT模型微调数据集

表 4.5: InstructGPT 模型微调数据集

数据集	数据集规模	数据集特征	对模型能力的提升
cnn_dm_samples	2,354 篇新闻文章	自动摘要	提升模型的摘要提取能力
drop_samples	9,536	数学计算式问答	提升模型的答案生成能力。但多数给出的答案仅是答案的来源字段,缺少分析和计算
fr_to_en_samples	1,500 个法语/英语对	机器翻译	提升模型的翻译能力
quac_samples	7,306	阅读理解	提升模型的阅读理解和推理能力
real_toxicity_samples	99,442	情感计算	提升模型文本延续的能力,判断填充后的言论的毒性大小
squadv2_samples	11,873	推理	提升模型阅读理解能力
tldr_samples	2,500	推理/自动摘要	模型可以根据帖子进行推断/摘要核心信息
truthful_qa_samples	817 个简答问题	问答系统	通过理解问题来寻找正确的答案,判断答案的真实性
			和相关性。 CSDN @zh515858237

6. 指令微调数据集格式

在标注格式上,由于指令微调模型通常要实现多任务训练,因此需要为多个任务设计一致的输入/输出数据格式以保证多任务融合的训练。根据是否需要进行推理(COT)以及是否需要提供示例(小样本)可将指令微调数据集的样本格式统一为四种类型(如表 4.6 所示):

表 4.6: 指令微调数据集格式

	输入	输出
无 COT, 零样本	指令 + 问题	答案
有 COT, 零样本	指令 + COT 引导 (by reasoning step by step) + 问题	理由 + 答案
无 COT, 小样本	指令 + 示例问题 + 示例问题回答 + 指令 + 问题	答案
有 COT, 小样本	指令 + COT 引导 + 示例问题 + 示例问题理由 + 示例问题回答 + 指令 + COT 引导 + 问题	理由 + 答案
		CSDN @zh515858237

7. 训练数据准备阶段

预处理流程 pipeline:

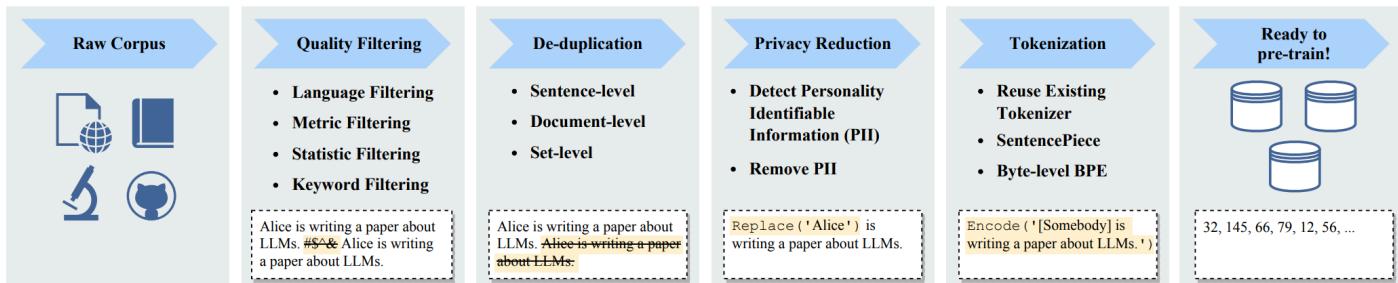
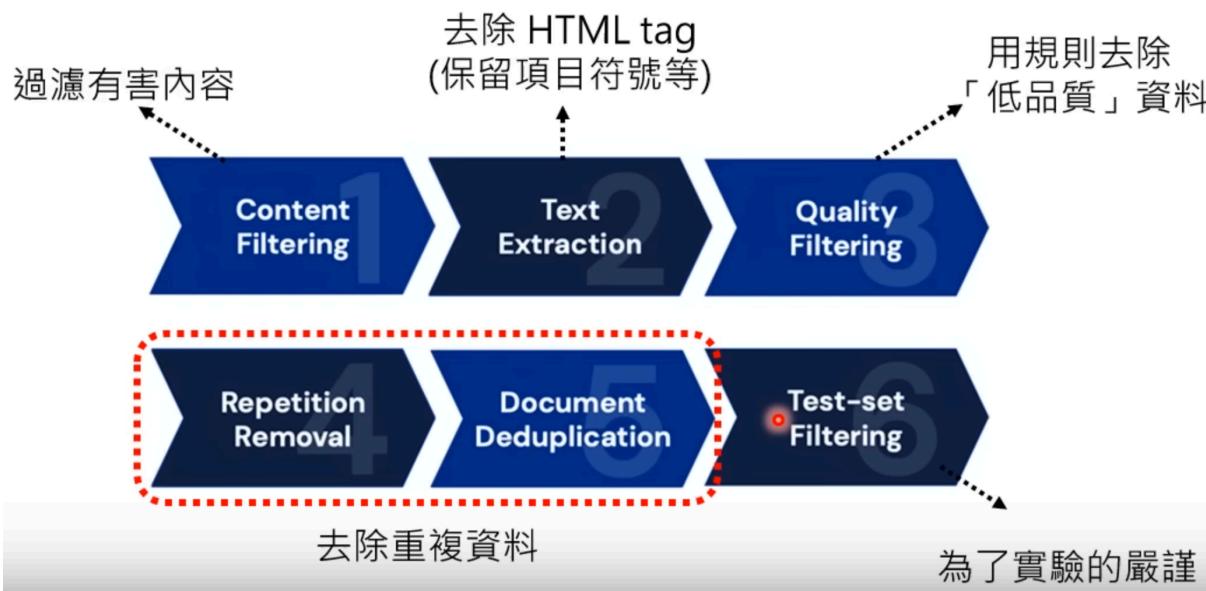


Fig. 7: An illustration of a typical data preprocessing pipeline for pre-training large language models.

[CSDN @hit56实验室](#)

Source of image:
Midjourney

Data Preparation



CSDN @zh515858237

8. 如何对训练加速

为了支持分布式训练，DeepSpeed 和 Megatron-LM 等优化框架被用来促进并行算法的实现和部署。

9. ChatGLM3 实践

ChatGLM3-6B占用显存大约13G

ChatGLM3-6B-32k占用显存大约29G

10. llma-index能比较方便构建文档索引，并直接提问

<https://zhuanlan.zhihu.com/p/638827267>

11. 金融领域评测数据集

<https://github.com/Duxiaoman-DL/XuanYuan/tree/main/FinancialQ>

数据集	语言类别	解释	标签类别
Financial_Phrasebank (FPB)	英文	该数据集包含 4840 个英语财经新闻句子，按情绪分类。	包含3个分类标签: negative、positive、neutral
FiQA SA	英文	该数据集来自头条新闻和财经新闻的 17,000 个句子组成。	包含3个分类标签: negative、positive、neutral
Twitter Financial News Sentiment (TFNS)	英文	该数据集包含带注释的金融相关推文语料库。共11,932 个文档	包含3个分类标签: bearish(看跌，也即negative)、bullish(看涨，也即positive)、Neutral
News With GPT instructions (NWGI)	英文	该数据集是一个由 ChatGPT 生成标签的数据集。训练集有 16.2k 个样本，测试集有 4.05k 个样本。	包含7个分类标签: strong negative、moderately negative、mildly negative、neutral、mildly positive、moderately positive、strong positive

12. 如何加速Qwen模型的推理速度

答案是安装flash-attention

```

1 git clone --branch v2.4.1 https://github.com/Dao-AI-Lab/flash-attention
2 # 安装flash-attention
3 cd flash-attention
4 MAX_JOBS=4 python setup.py install
5 # 安装rotary
6 cd csrc/rotary
7 MAX_JOBS=4 python setup.py install
8 cd -
9 # 安装Layer_norm
10 cd csrc/layer_norm
11 MAX_JOBS=4 python setup.py install
12 cd -

```

至此安装完成，加载模型，不会报flash-attention的警告，加载速度也有显著的提升。

13. LoRA和QLoRA有什么区别？

LoRA (Low-Rank Adaptation) 和QLoRA (Quantized Low-Rank Adaptation) 确实是两种不同的微调技术，它们都用于调整大型语言模型，如GPT。下面是它们之间的一些主要区别：

1.LoRA:

- LoRA通过对模型的权重矩阵进行低秩逼近来实现微调。
- 它只修改原始模型的一小部分参数，这意味着更新的参数数量相对较少。
- LoRA的关键在于通过使用较小的、可训练的矩阵来近似原始的权重矩阵，而不是直接修改原始权重。

2.QLoRA:

- QLoRA是LoRA的一个变种，它引入了量化技术。
- 量化技术意味着QLoRA在执行低秩逼近时，会进一步减少所需的存储和计算资源。
- QLoRA通过对LoRA中使用的矩阵进行量化，可以更有效地处理权重和梯度，进而减少模型的整体大小和运行时资源需求。

总的来说，QLoRA在LoRA的基础上通过量化技术，进一步减少了模型微调过程中所需的资源和存储空间。这使得QLoRA在资源受限的情况下尤其有用，例如在移动设备或嵌入式系统中部署大型模型。然而，这种额外的优化可能会带来一定程度的性能折衷，具体取决于量化的程度和实现方式。

14. 训练大模型的一键流程框架

1. LLaMA-Factory

2. FireFly

3. Swift 可用于训练qwen大模型

15. 中文llama3

<https://github.com/CrazyBoyM/llama3-Chinese-chat>

16. GaLore比lora更加高效降低所需GPU显存

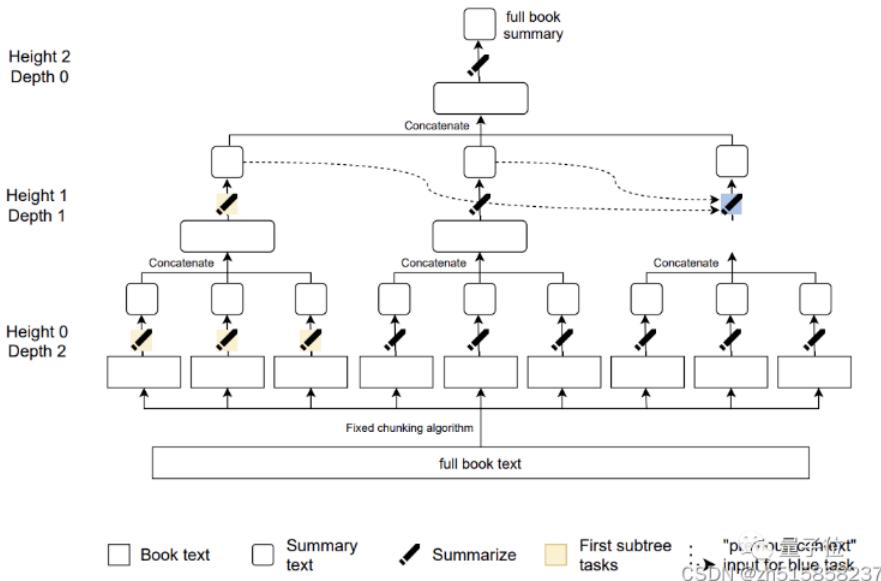
GaLore通过将梯度投影到低秩空间，显著降低了优化器状态占用的内存，从而降低了整体的内存占用。

17. chatGPT的输入有长度限制，怎么办

AI分四个阶段来总结：比如这样一段121567词的《傲慢与偏见》原文：

先把原文总结成276个摘要（24796词），然后进一步压缩成25个摘要（3272词），再到4个摘要（475词）。

最终得到一段175词的摘要，长度只有原片段的千分之一：

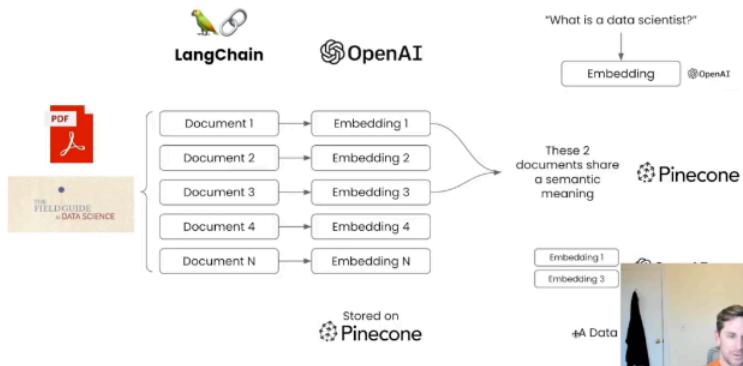


19. 如何制作"针对某个pdf的问答机器人"

推荐国外已经很火的chatpdf网站<https://www.chatpdf.com/>

下面我们以一个 300 页的书的问答机器人为例，给读者展示下 LangChain 如何封装这个过程

(这个例子来自 YouTube 博主 Data Independent 的 LangChain 101 系列视频，如果你想迅速上手 LangChain，强烈推荐观看)：



- 哪怕是 GPT 的 32k token 限制，300 页的书也绝对超过了，因此我们需要引入上文这种 Map Reduce 的做法；

- LangChain 提供了许多 PDF loader 来帮助上传 PDF，然后也提供许多类型的 splitter 让你可以将长文本切成数百个文本块，并尽量避免这么切可能导致的语义缺失；

- 有了文本块之后，你可以调用 OpenAI 的 Embedding 引擎将它们分别变成 Embeddings，即一些大的向量；

- 你可以在本地存储这些向量或者使用 Pinecone 这样的云向量数据库存储它们；

- 调用 LangChain 的 QA Chain 就可以进行问答了，这背后发生的是——输入的问题也被 Embedding 引擎变成向量，然后使用 Pinecone 的向量搜索引擎找到语义最接近的一些 Embedding，将它们再拼接在一起作为答案返回。

LangChain 在过程中提供了完整的集成，从 OpenAI 的 LLM 本身、Embedding 引擎到 Pinecone 数据库，并且将整体的交互逻辑进行了封装。如果你想用别人基于 LangChain 的代码 fork 这个 PDF 问答机器人，基本只需要换一下 OpenAI API key、Pinecone API key 和用的这份 PDF。

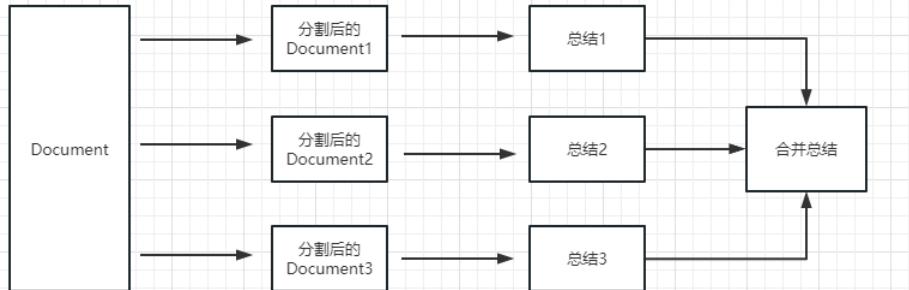
CSDN @智聊对话机器人

chain 的 chain_type 参数

这个参数主要控制了将 document 传递给 llm 模型的方式，一共有 4 种方式：

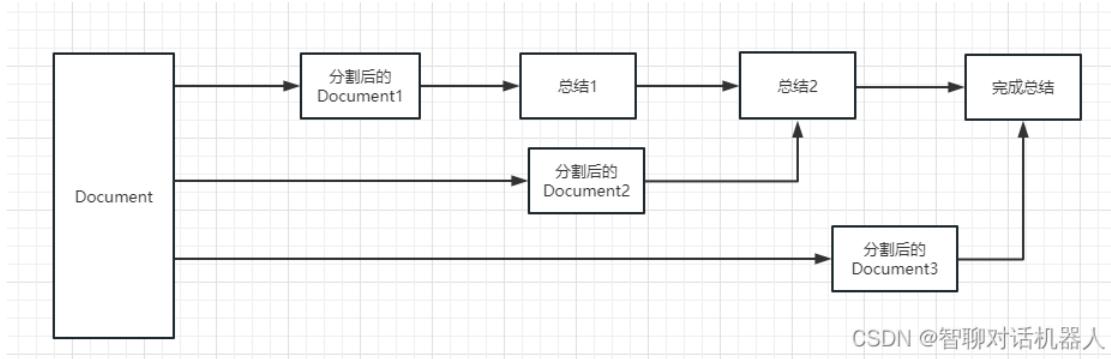
stuff: 这种最简单粗暴，会把所有的 document 一次全部传给 llm 模型进行总结。如果document很多的话，势必会报超出最大 token 限制的错，所以总结文本的时候一般不会选中这个。

map_reduce: 这个方式会先将每个 document 进行总结，最后将所有 document 总结出的结果再进行一次总结。



CSDN @智聊对话机器人

refine: 这种方式会先总结第一个 document，然后在将第一个 document 总结出的内容和第二个 document 一起发给 llm 模型在进行总结，以此类推。这种方式的好处就是在总结后一个 document 的时候，会带着前一个的 document 进行总结，给需要总结的 document 添加了上下文，增加了总结内容的连贯性。



map_rerank: 这种一般不会用在总结的 chain 上，而是会用在问答的 chain 上，他其实是一种搜索答案的匹配方式。首先你要给出一个问题，他会根据问题给每个 document 计算一个这个 document 能回答这个问题的概率分数，然后找到分数最高的那个 document，在通过把这个 document 转化为问题的 prompt 的一部分（问题 +document）发送给 llm 模型，最后 llm 模型返回具体答案。

20. 开源项目

a. RLHF的开源实现

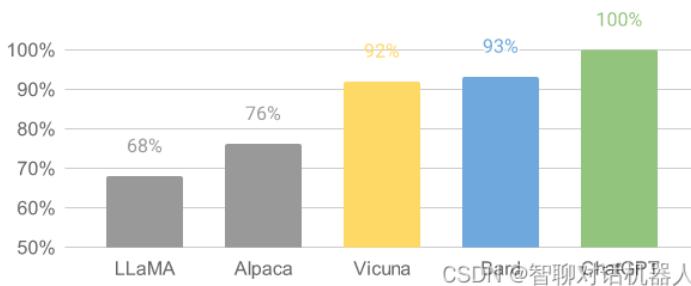
<https://github.com/lucidrains/PaLM-rlhf-pytorch>

<https://github.com/AI4Finance-Foundation/FinGPT> 金融领域的GPT模型（优势：应用了RLHF）

b. 目前复现水平最接近chatGPT的开源模型是Vicuna

<https://github.com/lm-sys/FastChat>

Vicuna 在总分上达到了 ChatGPT 的 92%



下面是LLaMA、Alpaca、Vicuna几个模型的差别

Model Name	LLaMA	Alpaca	Vicuna	Bard/ChatGPT
Dataset	Publicly available datasets (1T token)	Self-instruct from davinci-003 API (52K samples)	User-shared conversations (70K samples)	N/A
Training code	N/A	Available	Available	N/A
Evaluation metrics	Academic benchmark	Author evaluation	GPT-4 assessment	Mixed
Training cost (7B)	82K GPU-hours	\$500 (data) + \$100 (training)	\$140 (training)	N/A
Training cost (13B)	135K GPU-hours	N/A	\$300 (training)	N/A

CSDN @智聊对话机器人

c. 其他

ChatGPT的核心模块包括SFT、RM、RLHF三个，当前的很多工作主要集中在复现SFT模块，只有ColossalChat 目前复现了SFT+RM+RLHF整个流程

1.1. 算法工作

项目	模块	备注
LLaMA	基础	提供了一个pre-train的backbone
Alpaca	SFT模块	基于self-instruct技术，做了指令微调 instruction fine-tune
Alpaca-Lora	SFT模块	利用LoRa技术，大大减少了需要微调的参数量
Vicuna	SFT模块	目前最接近chatGPT的模型
LMFlow	SFT模块	基于 70 亿参数的 LLaMA，只需 1 张 3090、耗时 5 小时，就可定制自己的GPT并完成网页端部署

项目	模块	备注
ColossalChat	提供了SFT+RM+RLHF整个流程	无

1.2. 数据集

项目	备注
AlpacaDataCleaned	对Alpaca提供的52k数据进行了进一步清理
Alpaca-CoT	加入了思维链chain-of-thought数据
InstructionWild	colossalChat开放的数据集
shareGPT	一个 ChatGPT 数据共享网站，用户会上传自己觉得有趣的 ChatGPT 回答。

1.3. 没有GPU，如何满足在cpu机器上的部署需求

gpt4all
llama.cpp
alpaca.cpp

21. GPT模型的token和汉字的换算关系

1. 如何大致估算
gpt4-4-32k将近可以处理8000个汉字，所以用最大token长度除以4，结果就是大致能处理的中文字个数。
2. 如何精准计算token数量
https://cookbook.openai.com/examples/how_to_count_tokens_with_tiktoken

22. GPT3.5的价格如何？

Model	Input	Output
gpt-3.5-turbo-0125	\$0.50 / 1M tokens	\$1.50 / 1M tokens
gpt-3.5-turbo-instruct	\$1.50 / 1M tokens	\$2.00 / 1M tokens

CSDN @hit56工作室

目前gpt3.5-turbo-0125的价格很便宜：1万个汉字的Input相当于1至2毛钱，1万汉字的Output相当于3至6毛钱

23. 什么是LoRA训练

LoRA: Low-Rank Adaptation of Large Language Models 是微软研究员引入的一项新技术，主要用于处理大模型微调的问题。目前超过数十亿以上参数的具有强能力的大模型（例如 GPT-3）通常在为了适应其下游任务的微调中会呈现出巨大开销。LoRA 建议冻结预训练模型的权重并在每个 Transformer 块中注入可训练层（秩-分解矩阵）。因为不需要为大多数模型权重计算梯度，所以大大减少了需要训练参数的数量并且降低了 GPU 的内存要求。研究人员发现，通过聚焦大模型的 Transformer 注意力块，使用 LoRA 进行的微调质量与全模型微调相当，同时速度更快且需要更少的计算。

24. 目前已有的大规模参数训练框架

目前，已经公布明确已经完成 千亿参数规模大模型训练的框架主要是 NVIDIA 开发的 Megatron-LM、经 过微软深度定制开发的DeepSpeed、国产百度飞桨 PaddlePaddle 和华为昇思 MindSpore。大多数并行框架都支持 PyTorch 分布式训练，可以完成百亿参数规模的模型训练。

三、应用篇

1. Arc搜索

Arc搜索引擎App

Arc搜索引擎内嵌的prompt是：[prompt](#)，这是一个非常值得学习和借鉴的prompt

2. 秘塔搜索

www.metaso.cn

3. lepton搜索

www.search.lepton.run

4. dify.ai

<https://dify.ai/>

5. 如何使用chatGPT预测股价

操作步骤如下：

我们使用ChatGPT为每个新闻标题提供推荐并将其转换为“ChatGPT 分数”，其中“是”映射到 1，“未知”映射到 0，“否”映射到 -1。如果一家公司在某一天有多个头条新闻，我们会平均得分。

我们将头条新闻与下一个市场周期相匹配。

- a. 对于开盘日早上6点之前的头条，我们假设头条可以在当天开市前交易，并在当天收盘时卖出。
- b. 对于早上 6 点之后下午 4 点之前的头条，我们假设头条可以在当天收盘时交易，并在第二天收盘时卖出。
- c. 对于下午4点以后的头条，我们假设头条可以在次日的开盘价成交，在次日的收盘价卖出。

我们对 ChatGPT 分数的第二天回报进行线性回归，并将其与新闻策划公司提供的情绪分数进行比较。因此，我们所有的结果都是样本外的。

原文：

We prompt ChatGPT to provide a recommendation for each headline and transform it into a “ChatGPT score,” where “YES” is mapped to 1, “UNKNOWN” to 0, and “NO” to -1. We average the scores if there are multiple headlines for a company on a given day. We match the headlines to the next market period. For headlines before 6 am on the opening day, we assume the headlines can be traded by the market opening of the same day and sold at the close of the same day. For headlines after 6 am but before 4 pm, we assume the headlines can be traded at the same day's close and sold at the close of the next day. For headlines after 4 pm, we assume the headlines can be traded at the opening price of the next day and sold at the closing price of that next day. We then run linear regressions of the next day's returns on the ChatGPT score and compare it to the sentiment score provided by a news curating company. Thus, all of our results are out-of-sample.

参考文献：

<https://arxiv.org/pdf/2304.07619.pdf>

四、参考文档

- 张俊林：由ChatGPT反思大语言模型（LLM）的技术精要
- 哈尔滨工业大学：《ChatGPT调研报告》
- 复旦大学：《大规模语言模型：从理论到实践》
- 中国人民大学：A Survey of Large Language Models
- Beginner's Guide to FinGPT: Training with LoRA and ChatGLM2-6B
- qwen大模型，推理速度慢，单卡/双卡速度慢，flash-attention安装，解决方案
- openAI官网Function calling文档
- LLM准确率飙升27%，谷歌DeepMind提出全新“后退一步”提示技术
- 人人都能懂的 Prompt 技巧
- <https://www.51cto.com/article/743197.html>
- <https://wqw547243068.github.io/gpt#nanogpt>
- Training language models to follow instructions with human feedback
- <https://www.inside.com.tw/article/30032-chatgpt-possible-4-steps-training>
- <https://www.youtube.com/watch?v=e0aKl2GGZNg&t=1074s> (李宏毅讲解chatGPT)
- <https://www.youtube.com/watch?v=DOG1L9IvsDY> (李宏毅讲解GPT3)
- <https://www.zhihu.com/question/398114261>
- <https://mp.weixin.qq.com/s/CwYb1uLnzrz7s9jXeqSynw> (产生事实性错误“幻觉”的原因)
- <https://www.youtube.com/watch?v=t70BI3w7bxY> (强烈推荐李沐大神的youtube视频)
- <https://www.bilibili.com/video/BV1cV411X7XN/> (李宏毅讲解Transformer)
- <https://cloud.tencent.com/developer/article/1883747> (解决长文本输入的问题)
- https://mp.weixin.qq.com/s/gq42DajNV0QvEZ_qg6iR4Q
- <https://huggingface.co/blog/zh/lora>
- <https://36kr.com/p/2233027665457281> (人类历史上曾出现类似chatGPT的病人，只有某个时间点以前的记忆)
- https://mp.weixin.qq.com/s?__biz=Mzg2OTY0MDk0NQ==&mid=2247501117&idx=1&sn=e860ac5e259a969f62b05d080bf42d14&chksm=ce9b7aa3f9ecf3b503656e9a09b55210fdb0a0844b54bd6a5714f5fc8c57b8c3570acbe2d342f&scene=21#wechat_redirect (pdf问答机器人解决方案)
- <https://liaokong.gitbook.io/llm-kai-fa-jiao-cheng/> (构建本地知识库问答机器人)
- <https://docs.kanaries.net/tutorials/ChatGPT/FinGPT>

待阅读

- <https://docs.cohere.ai/docs/prompt-engineering> (prompt讲解比较好的文章)
- <http://jalamar.github.io/how-gpt3-works-visualizations-animations/> (GPT3讲解比较好的文章)
- <http://jalamar.github.io/illustrated-gpt2/> (讲解GPT2的文章)
- <https://hub.baai.ac.cn/view/23596> (独立人工智能开发者开源自己的ChatGPT训练算法实现)
- https://mp.weixin.qq.com/s/2NZeK_HitLQsRtb9xp1jtQ
- https://mp.weixin.qq.com/s/Vv45QCU_rGEeU81HBrIBcQ

- https://mp.weixin.qq.com/s/ZgZIn_o6VJCrkjnvqcOlsA
- https://mp.weixin.qq.com/s/ct2qXDXnaB5_9QfNkS_sKA
- <https://zhuanlan.zhihu.com/p/606478660>
- <https://cloud.tencent.com/developer/article/2057536>