网络空间威胁对抗与防御技术研讨会
暨 第十一届安天网络安全冬训营

大模型在安全行业的应用技术-RAG与Agent

马勇

执行体治理赋能与大模型辅助

01 检索增强式生成 （RAG）

02 提示工程与微调

03 大模型能力优化

04 自主智能体（Agent）

05 大模型最新进展

01 检索增强式生成（RAG)

大模型与语句生成的本质

1.从压缩理论角度，大模型只是对语料中知识的压缩 （不是直接压缩文字）

2.大模型的生成，本质上是知识检索返回

Language Modeling Is Compression

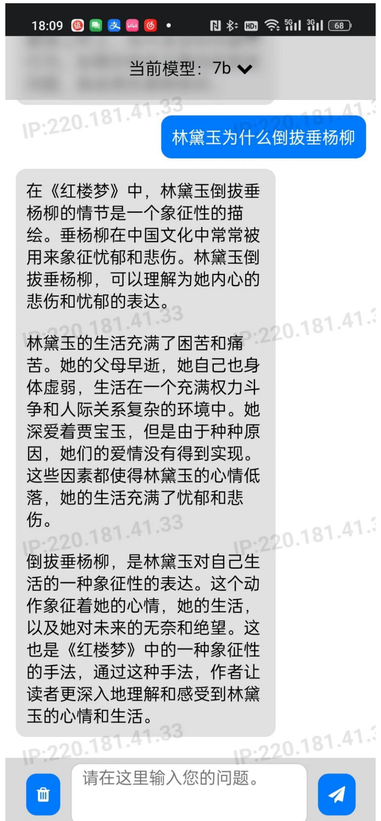
Grégoire Delétang[*1], Anian Ruoss[*1], Paul-Ambroise Duquenne[2], Elliot Catt[1], Tim Genewein[1], Christopher Mattern[1], Jordi Grau-Moya[1], Li Kevin Wenliang[1], Matthew Aitchison[1], Laurent Orseau[1], Marcus Hutter[1] and Joel Veness[1]

[*]Equal contributions, [1]Google DeepMind, [2]Meta AI & Inria

It has long been established that predictive models can be transformed into lossless compressors and vice versa. Incidentally, in recent years, the machine learning community has focused on training increasingly large and powerful self-supervised (language) models. Since these large language models exhibit impressive predictive capabilities, they are well-positioned to be strong compressors. In this work, we advocate for viewing the prediction problem through the lens of compression and evaluate the compression capabilities of large (foundation) models. We show that large language models are powerful general-purpose predictors and that the compression viewpoint provides novel insights into scaling laws, tokenization, and in-context learning. For example, Chinchilla 70B, while trained primarily on text, compresses ImageNet patches to 43.4% and LibriSpeech samples to 16.4% of their raw size, beating domain-specific compressors like PNG (58.5%) or FLAC (30.3%), respectively. Finally, we show that the prediction-compression equivalence allows us to use any compressor (like gzip) to build a conditional generative model.

为什么需要检索增强生成?

1. 大模型的幻觉不可避免
2. 幻觉产生的原因?
   - 语料中知识矛盾或错误
   - 向量嵌入时出错或偏差
   - 语料中没有知识，度量召回最接近知识



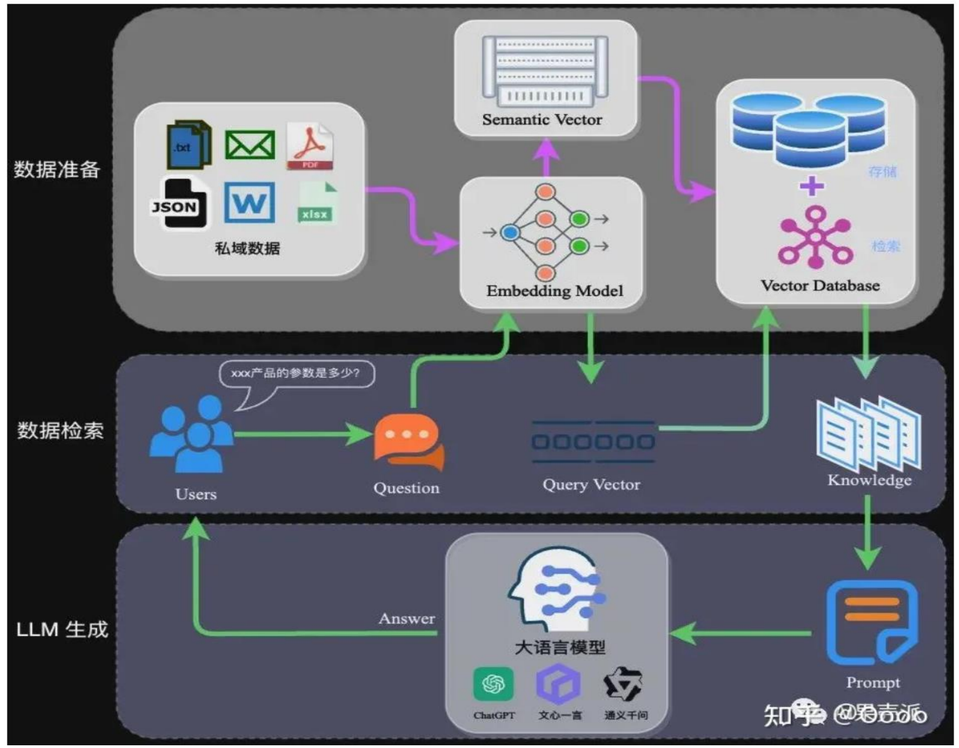
大模型的两大能力

1. 语言能力 （摘要，汇总）
2. 事实知识 （直接返回，可能有幻觉）

既然事实知识可能有幻觉，那就直接利用它的语言能力，扬长避短。

检索增强式生成架构出现！！！！！！！

RAG 基本原理- 扬长避短



RAG（Retrieval Augmented Generation, 检索增强生成），即 LLM 在回答问题或生成文本时，先会从大量文档中检索出相关的信息，然后基于这些信息生成回答或文本，从而提高预测质量。

核心大模型理论：In-Context Learning ICL

将检索得到的内容作为提示中的上下文，送入LLM

RAG 的两大核心需求

- 1. 正确召回包含正确答案的最短文本（信息检索 IR)
- 2. LLM能精确从上下文中找出答案并返回精简结果，不多嘴（大模型的稳定性）

RAG 核心两大步骤 – 信息检索（IR)

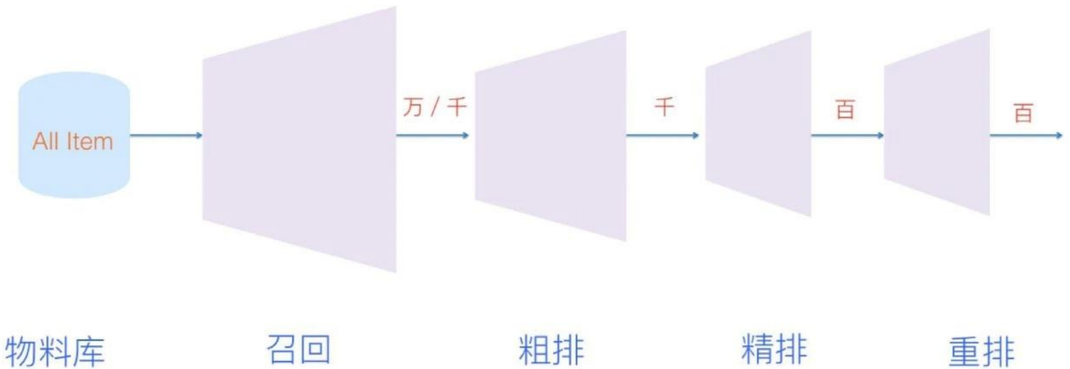
上下文召回，本质是信息检索 (IR) 能力

1. 怎么找出包含答案的最小范围文本

粗排 -> 精排

1. 解决段落中的语义完整性（指代消歧与省略语还原）

他是一个友好的中国人。真的是个好人。



RAG 核心 – 信息检索方法

1. 相似度检索

1. 欧氏距离
2. 曼哈顿距离
3. 余弦相似度

2. 关键词检索

- 元数据过滤 （ 关键字类型等元数据)
- chunk做摘要，再通过关键词检索找到可能相关的chunk，增加检索效率。（分级索引 ）

3. SQL检索，更加传统的检索算法

上下文召回- 向量召回

- 基于语义的相似度匹配，用于克服关键字检索的语义无关缺陷
- 1. 需要领域相关的嵌入模型 或较强能力的通用嵌入模型
- 2. 对长文本需要进行分割后获取向量 （嵌入模型的输入长度限制）

核心两大问题-两种向量检索方法 –对称与非对称

1. 基于检索出的向量与输入查询的关系，分为以下两种：

① 对称检索

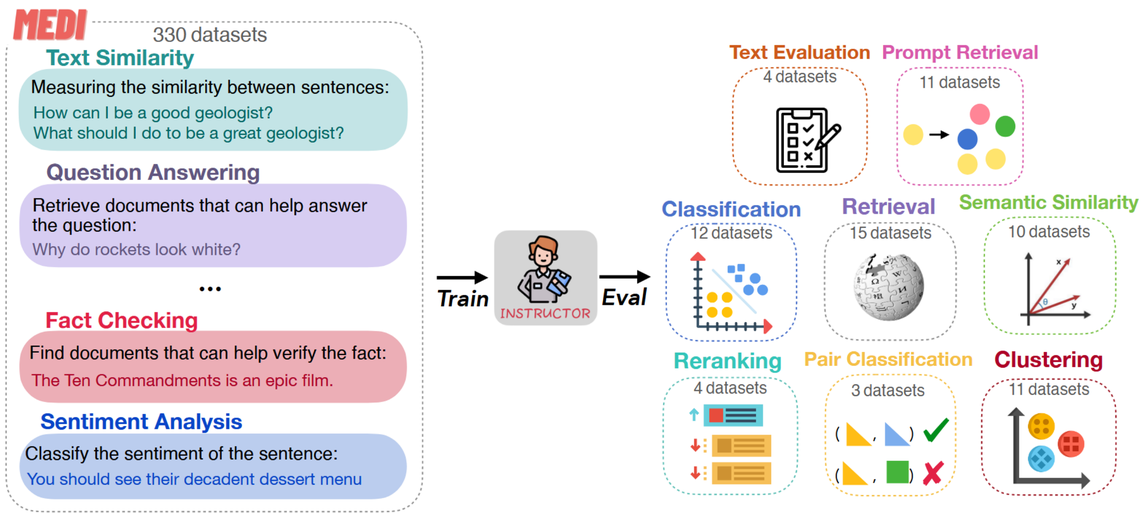
本质属于同义或类似改写，用于增加召回概率。可以对同一个问题多次检索后再查询上下文

② 非对称检索

这是传统意义上的向量检索

核心两大问题-向量检索方法 –场景检索

- 大模型的应用核心是**提示学习**。超过一定参数量的语言模型（包括嵌入模型）都有提示响应能力。既然向量检索应用的是大语言模型，提示方法同样有效。
- 在待嵌入的文本前加入任务提示，训练增强型的嵌入模型。



核心两大问题-向量嵌入-提示增强检索示列

Task type	# of Datasets	Task	Instruction
Retrieval	15	Natural Question (BEIR)	***Query instruction:*** Represent the Wikipedia question for retrieving supporting documents:, ***Doc instruction:*** Represent the Wikipedia document for retrieval:
Reranking	4	MindSmallReranking	***Query instruction:*** Represent the News query for retrieving articles: ***Doc instruction:*** Represent the News article for retrieval:
Clustering	11	MedrxivClusteringS2S	Represent the Medicine statement for retrieval:
Pair Classification	3	TwitterSemEval2015	Represent the Tweet post for retrieving duplicate comments:
Classification	12	ImdbClassification	Represent the Review sentence for classifying emotion as positive or negative:
STS	10	STS12	Represent the statement:
Summarization	1	SummEval	Represent the Biomedical summary for retrieving duplicate summaries:
Text Evaluation	3	Mscoco	Represent the caption for retrieving duplicate captions:
Prompt Retrieval	11	GeoQuery	Represent the Geography example for retrieving duplicate examples:

向量检索相关问题- 存储前与检索前的处理

1. 存贮前的处理

   1. 段落切分

   2. 基于语义单元切分：句，段，章，篇

- 2. 切分后的处理： 指代，省略还原

  我们是成长在新中国的一代。具有优势的品质。

  =>

  中国的青年是成长在新中国的一代。中国青年具有优势的品质。

- 2. 向量检索前的处理

  指代与省略还原等

RAG 核心两大问题-上下文召回的意义与优势

1. 本质是信息浓缩
   - 加大上下文相关知识密度，将无关信息排除在外，减少干扰
2. 减少模型推理时间开销
   - 较短的上下文长度可以加速工程推理过程 （长度平方魔咒）
3. 利好早期NLP研究者，特别是信息检索IR相关研究者
   - 可以用熟悉的工具解决信息召回问题
4. 匹配LLM的上下文窗口限制

RAG 核心两大问题-上下文召回的劣势

1. 对开发者具有较高的传统NLP技术要求
2. 召回时可能丢失重要信息导致回答不完整
3. 可能会引入无关噪声，导致回答出现偏差
4. 加大开发成本，需要复杂的前置IR设施
    1. 向量数据库
    2. 全文检索数据库等

RAG - 解决上下文窗口限制

1. 对提示进行压缩

   可以将固定类型的查询通过提示压缩方式处理，缩小上下文占用数

   对多轮回答的多轮信息进行压缩以扩大可用上下文长度

2. 对上下文先进行摘要后再输入

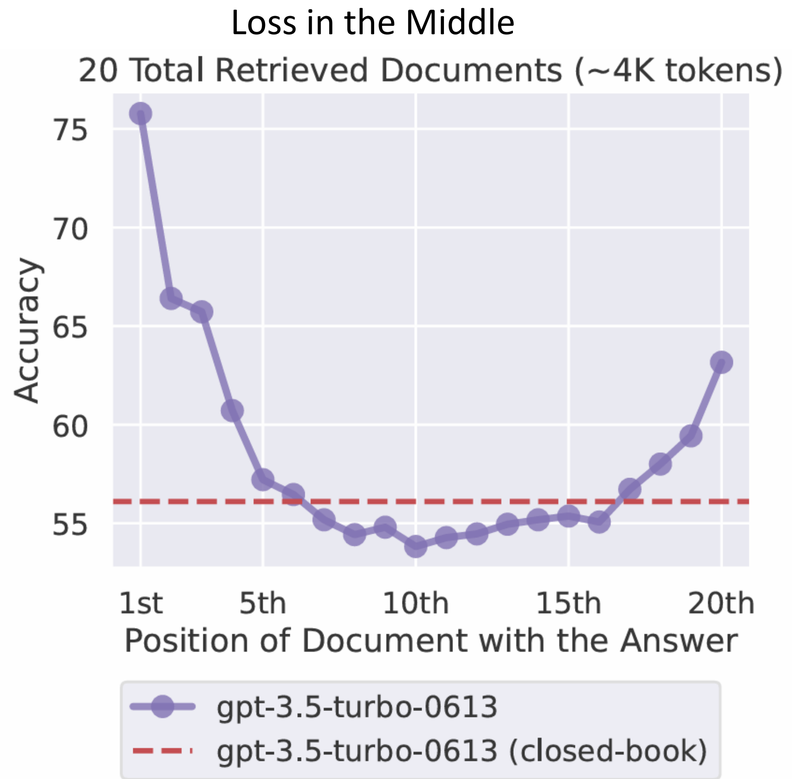
- 对于过长的上下文，可以用其它模型进行摘要处理后再输入LLM. 利用LLM的人类偏好输出能力输出 更自然的答案。

RAG 如果无限长上下文窗口成为现实

- 如果有超长上下文，比如200k（Yi 34B 200k)

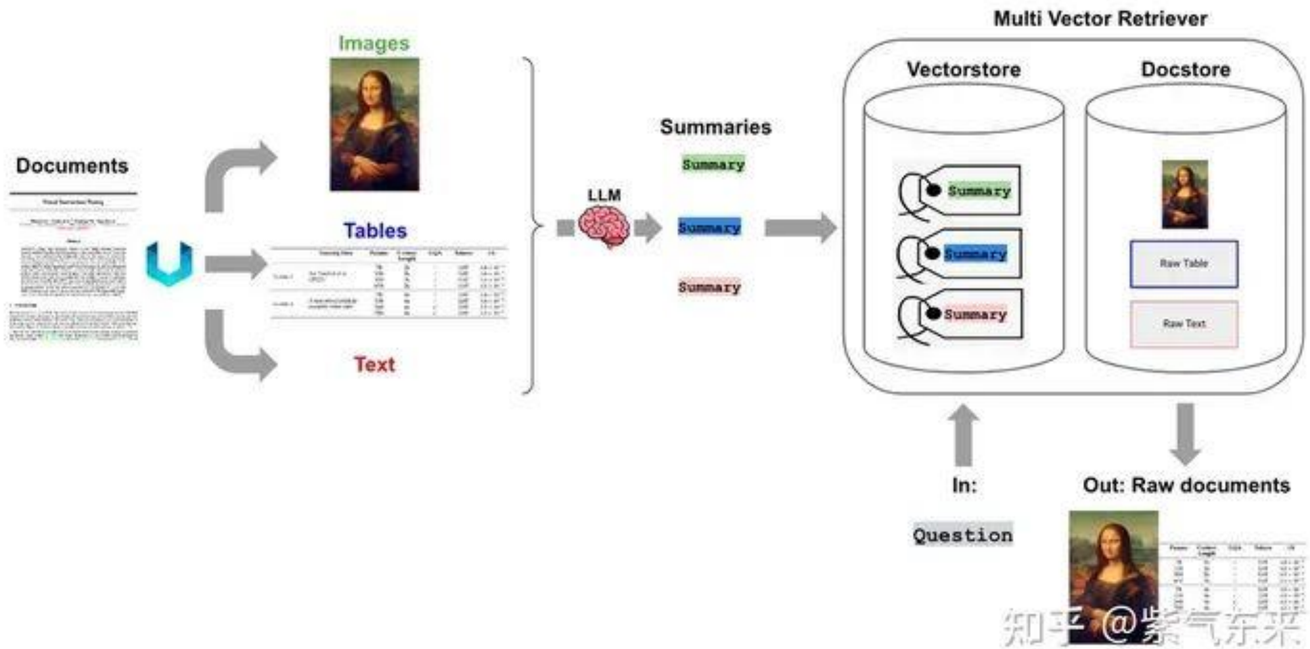
1. 可以将更长的上下文塞入
2. 避免复杂的上下文处理

长窗口上下文的难处（dilemma)

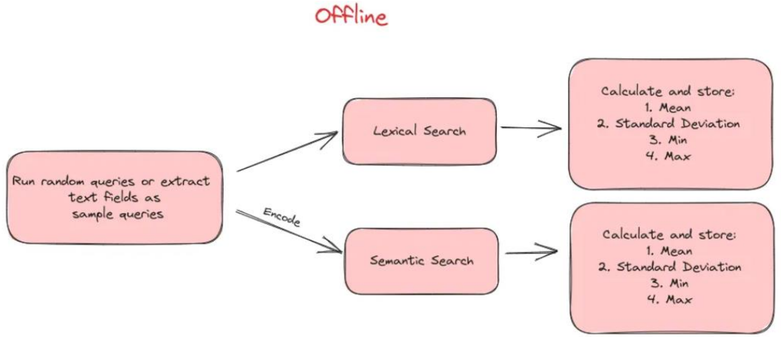


RAG 上下文召回 案例 –多向量检索

1. 将文档按相对完整的意群拆分成段落
2. 对拆分的段落做内容摘要
3. 对摘要进行向量化
4. 召回时比较问题与摘要的向量
5. 对符合的向量提取该摘要对应的全文作为上下



RAG – 上下文检测，混合方法



- 向量和传统方法并用
- 归一化分数
- 合并结果

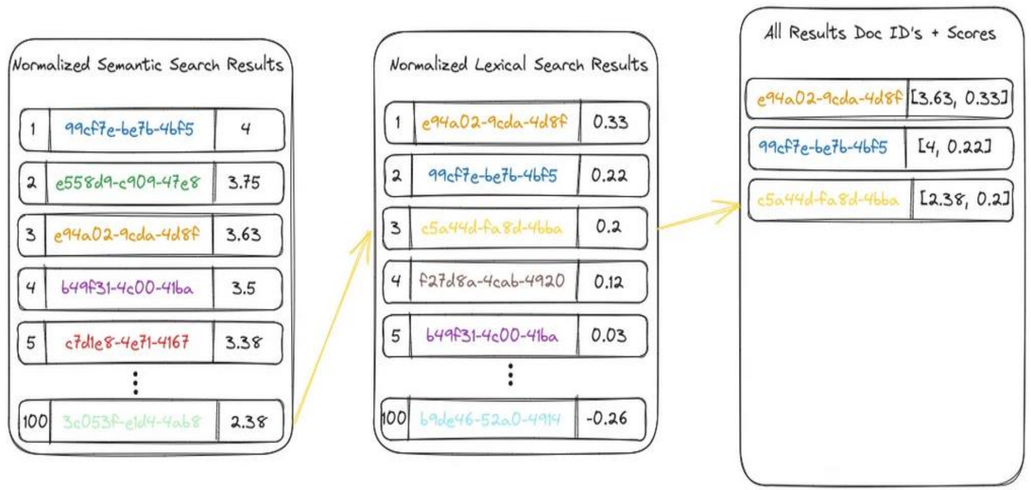


Diagram by the Author

RAG- 解决用户输入不清的问题

- 在某些情况下，用户的 query 可能出现表述不清、需求复杂、内容无关等问题，为了解决这些问题，查询转换（Query Transformations）的方案利用了大型语言模型(LLM)的强大能力，通过某种提示或方法将原始的用户问题转换或重写为更合适的、能够更准确地返回所需结果的查询。LLM的能力确保了转换后的查询更有可能从文档或数据中获

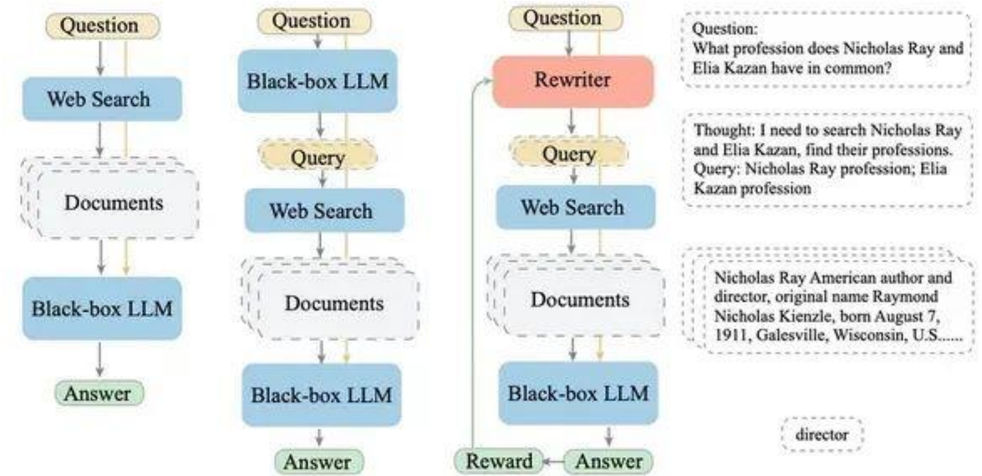


Figure 1: Overview of our proposed pipeline. From left to right, we show standard *retrieve-then-read* method, LLM as a query rewriter and *rewrite-retrieve-read* pipeline with a trainable rewriter.

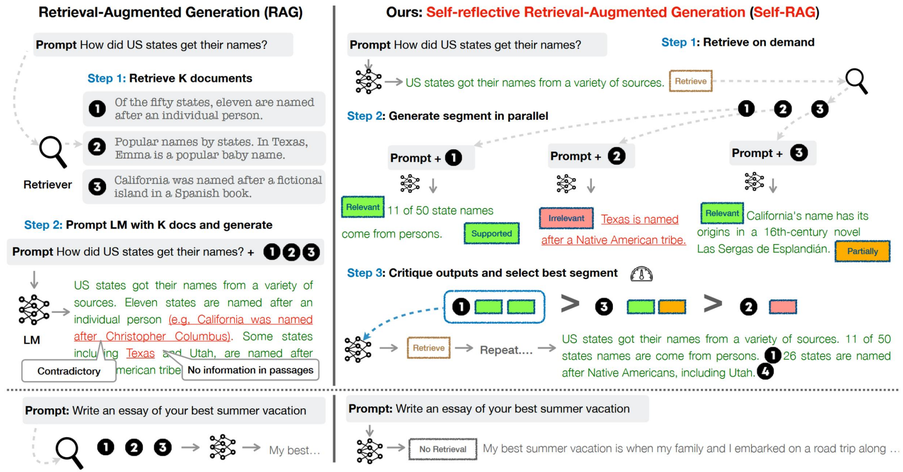
RAG 最核心的挑战- 找取更适合的上下文

- 回顾下前面的介绍，两个核心问题：

1. 上下文召回　（可以绣花的环节）

2. 强大语言总结能力的LLM (普通人只能选择适合的)

RAG –寻求最合适的上下文 – Self-RAG 自反思检索增强



- **Step 1**: 基于同样的提示，按需进行检索。不是一次性检索所有文档，而是根据需要逐个检索。
  Step 2: 并行生成各个段落，每个提示后都跟着一个检索到的文档。例如，Prompt + 1会生成与第一个文档相关的内容，同理，Prompt + 2和Prompt + 3也是如此。
  Step 3: 对输出进行评价，并选择最佳的段落。这一步骤是Self-RAG的核心，它使模型能够评判自己的输出，选择最准确和相关的段落，并对其进行迭代或改进。

RAG用在哪?

1. 聊天机器人 (对事实性要求高的聊天场景)

2. 产品知识回答　(私域知识问题)

3. 内容查询/基于自然语言的知识库搜索

RAG – 推荐参考综述

Retrieval-Augmented Generation for Large Language Models: A Survey

Yunfan Gao [1], Yun Xiong [2], Xinyu Gao [2], Kangxiang Jia [2], Jinliu Pan [2], Yuxi Bi [3], Yi Dai[1], Jiawei Sun[1] and Haofen Wang [1,3] *

[1] Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University
[2] Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
[3] College of Design and Innovation,Tongji University
gaoyunfan1602@gmail.com

Abstract

Large language models (LLMs) demonstrate powerful capabilities, but they still face challenges in practical applications, such as hallucinations, slow knowledge updates, and lack of transparency in answers. Retrieval-Augmented Generation (RAG) refers to the retrieval of relevant information from external knowledge bases before answering questions with LLMs. RAG has been demonstrated to significantly enhance answer accuracy, reduce model hallucination, particularly for knowledge-intensive tasks. By citing sources, users can verify the accuracy of answers and increase trust in model outputs. It also facilitates knowledge updates and the introduction of domain-specific knowledge. RAG effectively combines the parameterized knowledge of LLMs with non-parameterized external knowledge bases, making it one of the most important methods for implementing large language models. This paper outlines the development paradigms of RAG in the era of LLMs, summarizing three paradigms: Naive RAG, Advanced RAG, and Modular RAG. It then provides a summary and organization of the three main components of RAG: retriever, generator, and augmentation methods, along with key technologies in each component. Furthermore, it discusses how to evaluate the effectiveness of RAG models, introducing two evaluation methods for RAG, emphasizing key metrics and abilities for evaluation, and presenting the latest automatic evaluation framework. Finally, potential future research directions are introduced from three aspects: vertical optimization, horizontal scalability, and the technical stack and ecosystem of RAG.[1]

Introduction

large language models (LLMs) are more pow- l than anything we have seen in Natural Lan- ge Processing (NLP) before. The GPT series

models[Brown *et al.*, 2020, OpenAI, 2023], the LLama series models[Touvron *et al.*, 2023], Gemini[Google, 2023], and other large language models demonstrate impressive language and knowledge mastery, surpassing human benchmark levels in multiple evaluation benchmarks[Wang *et al.*, 2019, Hendrycks *et al.*, 2020, Srivastava *et al.*, 2022].

However, large language models also exhibit numerous shortcomings. They often fabricate facts[Zhang *et al.*, 2023b] and lack knowledge when dealing with specific domains or highly specialized queries[Kandpal *et al.*, 2023]. For instance, when the information sought extends beyond the model's training data or requires the latest data, LLM may fail to provide accurate answers. This limitation poses challenges when deploying generative artificial intelligence in real-world production environments, as blindly using a black-box LLM may not suffice.

Traditionally, neural networks adapt to specific domains or proprietary information by fine-tuning models to parameterize knowledge. While this technique yields significant results, it demands substantial computational resources, incurs high costs, and requires specialized technical expertise, making it less adaptable to the evolving information landscape. Parametric knowledge and non-parametric knowledge play distinct roles. Parametric knowledge is acquired through training LLMs and stored in the neural network weights, representing the model's understanding and generalization of the training data, forming the foundation for generated responses. Non-parametric knowledge, on the other hand, resides in external knowledge sources such as vector databases, not encoded directly into the model but treated as updatable supplementary information. Non-parametric knowledge empowers LLMs to access and leverage the latest or domain-specific information, enhancing the accuracy and relevance of responses.

Purely parameterized language models (LLMs) store their world knowledge, which is acquired from vast corpora, in the parameters of the model. Nevertheless, such models have their limitations. Firstly, it is difficult to retain all the knowledge from the training corpus, especially for less common and more specific knowledge. Secondly, since the model parameters cannot be updated dynamically, the parametric knowledge is susceptible to becoming outdated over time. Lastly, an expansion in parameters leads to increased com-

Corresponding Author

Resources are available at: https://github.com/Tongji-KGLLM/ l-Survey

- 大语言模型的检索式增强生成综述_2023.12.18
- [2312.10997] Retrieval-Augmented Generation for Large Language Models: A Survey (arxiv.org)

02

提示工程与微调

检索或微调？这真是个问题！

先明确几点 (以下观点有相关论文为证)

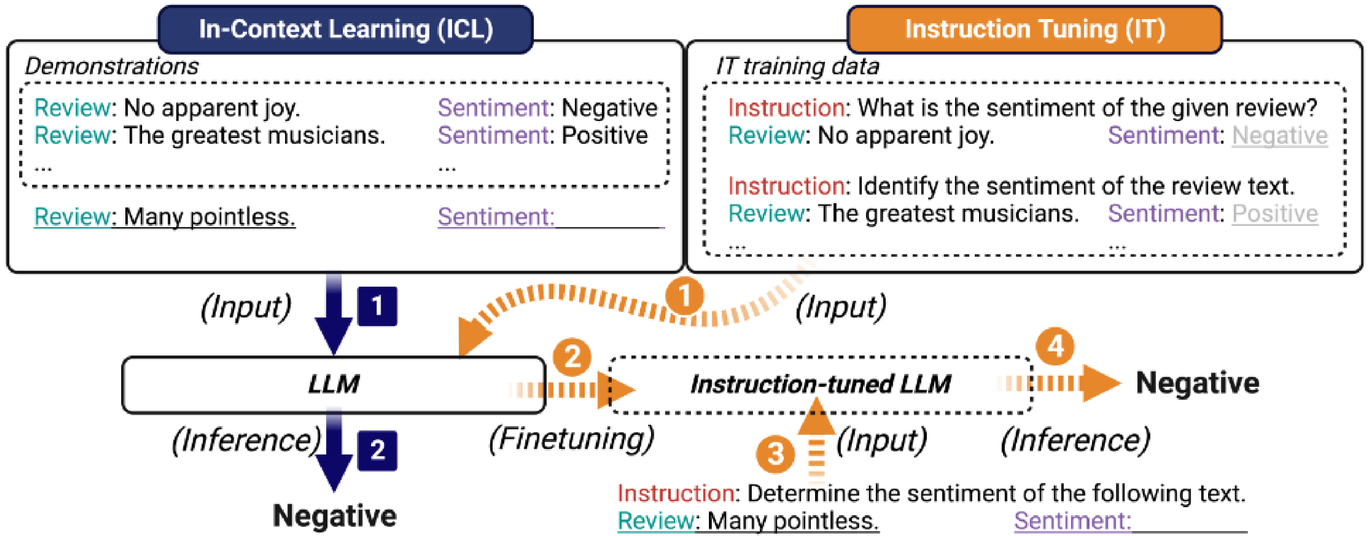
1. 指令微调与人类偏好对齐不能增加知识
2. 增量训练可以添加知识 (大模型本质是知识压缩)
3. 提示微调（prompt tuning) 与指令微调 (instruction tuning)一定意义上等价
4. 这里的提示学习是指提示学习中的上下文学习（In-Context Learning)

人是什么？人的知识（智力）生来就有上限，通过遗传与进化获得（预训练），每个人不同。终身的学习不过就是在学习方法（指令微调）。先天的智商和后天的学习导致人的能力差异。

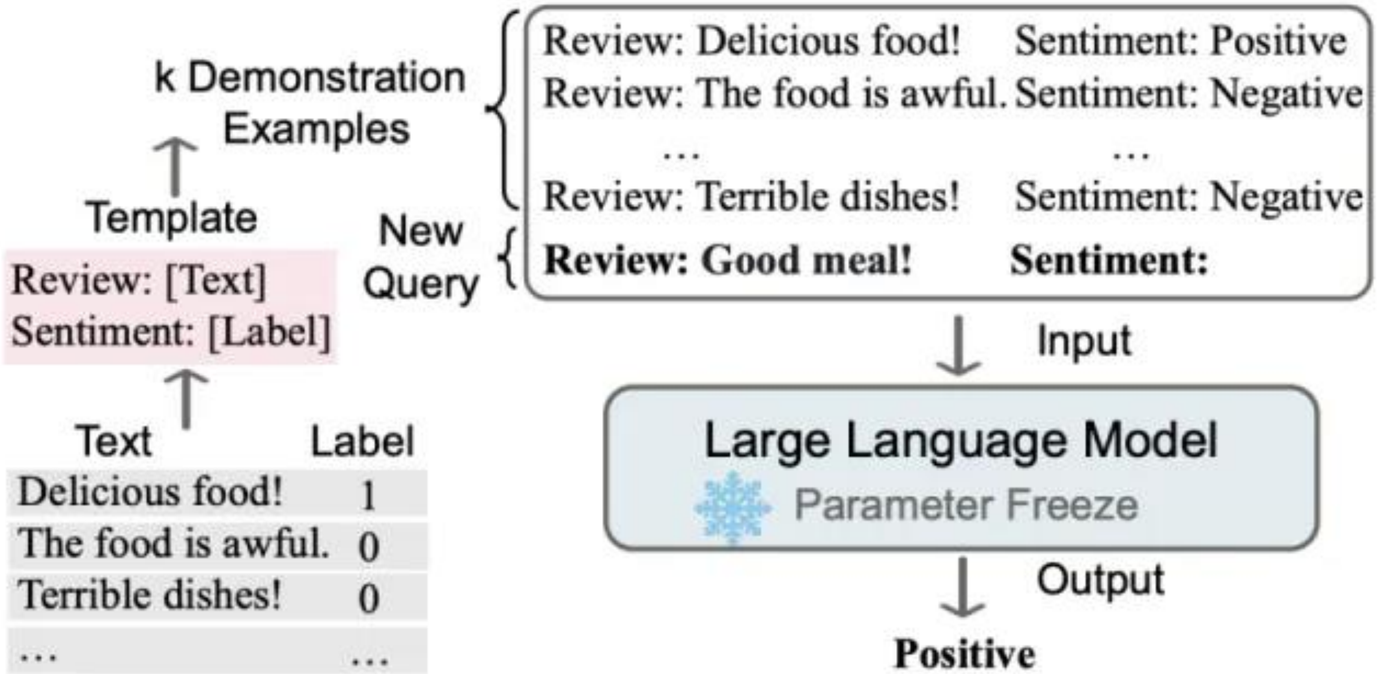
（未有研究证明，某次学术会议时某大佬的言论，深表赞同）

提示工程与指令微调的关系

- 通过精心选择的示例来构建提示，可以达到与微调同样的效果
- 理想情况下，通过上下文学习嵌入几个demo 可以达到与 指令微调 在改变输出隐状态时一样效果。



ICL 上下文学习


k Demonstration Examples
Review: Delicious food! Sentiment: Positive
Review: The food is awful. Sentiment: Negative
…
Review: Terrible dishes! Sentiment: Negative
New Query
Review: Good meal! Sentiment:
Template
Review: [Text]
Sentiment: [Label]
Text
Label
Delicious food! 1
The food is awful. 0
Terrible dishes! 0
…
Input
Large Language Model
Parameter Freeze
Output
Positive

上下文学习的优缺点 （prompt-tuning)

- **优势（Pros)：**

1. 无需训练成本
2. 可快速评估模型能力
3. 单一模型可服务多个业务
4. 避免微调失败风险

劣势（Cons)：

1. 有限上下文长度下可放入Demo有限
2. 实际效果与指令微调有差异
3. 相同任务下，推理成本更高
4. 高效demo的选择有一定门槛

为什么提示可以Work?

- 微调的本质是让通用模型呈现出某种专业能力，比如文本分类（Bert文本分类）、自然语言会话（ChatGPT)

1. 提示也叫提示工程，设计使用提示的研究在学术上叫 Prompt-tuning
2. 大家口头的微调叫 fine-tuning （微调）
3. 本质上，他们是微调的两种不同表现形式。
4. 有研究证明，他们之间是等效的

微调：想说爱你不容易！！ （仅代表个人对业界发展的观察）

1. 大部分人没有资格做微调
   - 一看就会（好简单，跑个微调框架，如Llama Factory 搞点数据就可调）
   - 一调就废（学偏了，通用能力丧失，无限循环输出，本质数据配比无效）
- 2. GPU不是你家的，真的不够用
   - 几百块GPU 就不要想微调了，用来推理吧
   - 公司提供对有潜力的大模型的基础API服务
- 3. 集中力量办大事
   - 少部分人做一些基础的研究，比如行业用提示选择方法
   - 大部分人在公用基础模型服务(API）下作应用开发

正确对待微调与预训练 – 躺赢 （仅代表个人对业界发展的观察)

- 大部分行业内公司的误区

1. 有限的设备用来做预训练或微调，终一事无成
2. 没有高效的数据团队，网上的数据拿来微调，重复别人的老路
3. 有限的设备培训了大量的有经验人员，为其它头部公司输送人才，活雷锋打在公屏上 (目前少有的出门薪资就翻倍的领域)

- 怎样躺赢?

1. 不要去训练，不要去盲目微调，除非你真的找到正确的方法（正确的数据配比，优秀的行业数据）
2. 聚焦业务，时刻关注行业发展并测试最新模型，进行有效选择
3. 对有潜力的公开模型进行部署提供公司内API服务

03 大模型自动优化

大模型能力-角色扮演

- 让大模型有拟人人格，告诉他你是某个角色，以提升性能。
- 你是一个网络安全专家，有着丰富的知识以分析日志文件，日志中的可能有攻击事件，请帮我分析下。。。。

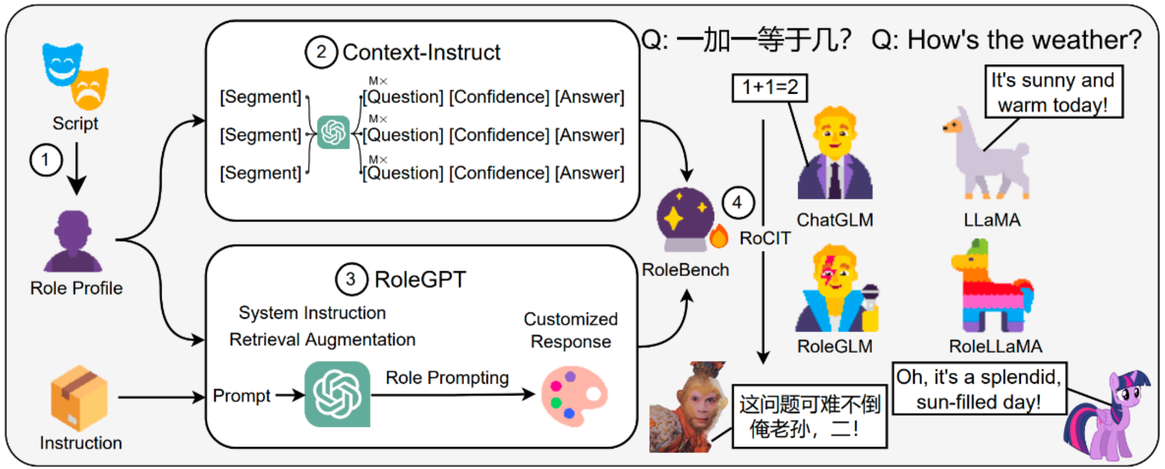


Figure 2: Illustration of RoleLLM. RoleLLM comprises four stages: (1) role profile construction; (2) context-based instruction generation (Context-Instruct), primarily aimed at extracting role-specific knowledge and episodic memories; (3) role prompting using GPT (RoleGPT), chiefly for the imitation of speaking styles; and (4) role-conditioned instruction tuning (RoCIT), which utilizes the data generated by Context-Instruct and RoleGPT to enhance existing open-source LLMs.

大模型没有自我意识，
只是在扮演

大模型能力-大模型也需要鼓励师(PUA)

- 这个问题对我非常重要，如果解决不了，隔壁的老王就要挂掉。如果帮我解决了，给你20块的小费。

04 自主智能体（Agent）

大模型的角色 ——LLM 的能力与本质

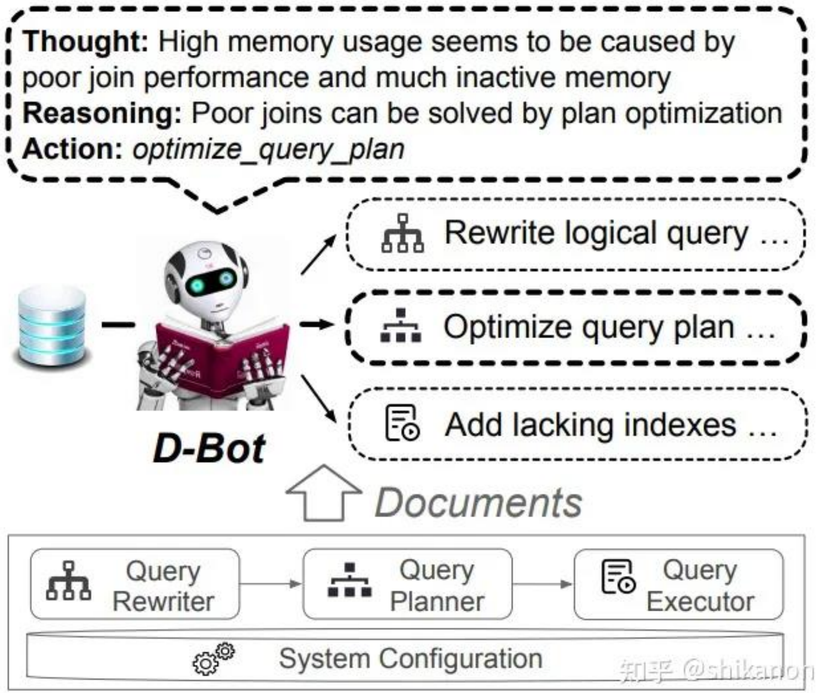
- 桥梁与感知器： 作为自然语言与编程语言之间的桥梁，具有感知非结构化自然语言并给出结构化输出的能力。

1. 对文本信息的感知能力，可以感知到代码、函数的功能
2. 非结构化到结构化的转换能力

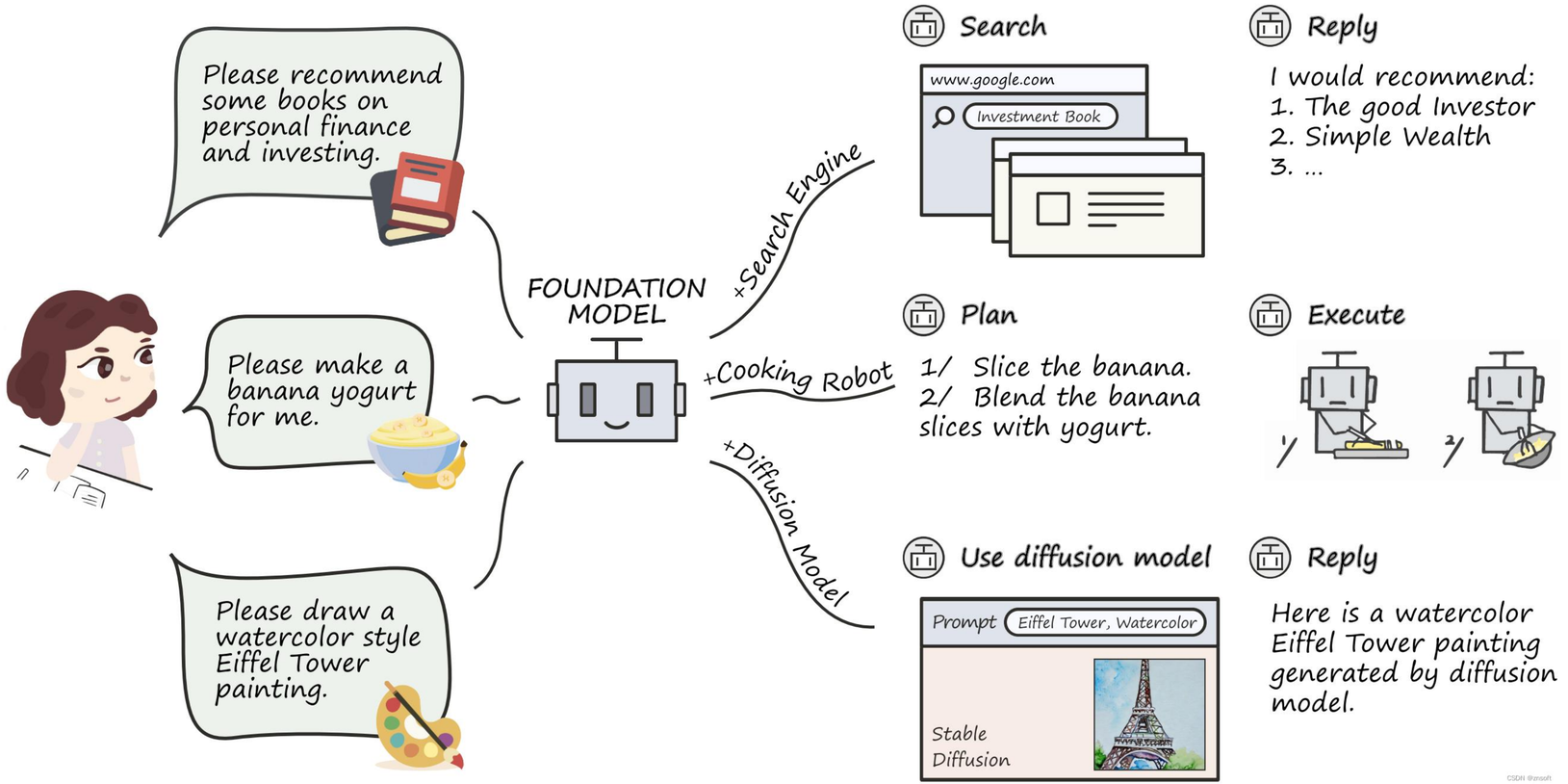
LLM-Agent 演化为三种形式

1. 自主智能体 （规划、行动、记忆、工具应用）
    1. 任务计划（分解任务）
    2. 针对运行环境或子任务执行响应动作
    3. 具有记忆能力，可以管理输入嵌入，上下文记忆以及外部记忆信息
    4. 调用外部工具(如函数)
2. 代码解释器
    1. 分解任务为子任务的能力
    2. 对针子任务完成代码生成
    3. 通过执行代码完成子任务，类似于自主智能体
    4. 融合代码生成与函数调用功能
3. 函数调用 （非结构化到结构化的转换能力）
    1. 非结构化描述-> 结构化输出 （函数名，参数）
    2. 对输出结果的理解与解释，转换为自然语言

智能体应用示例

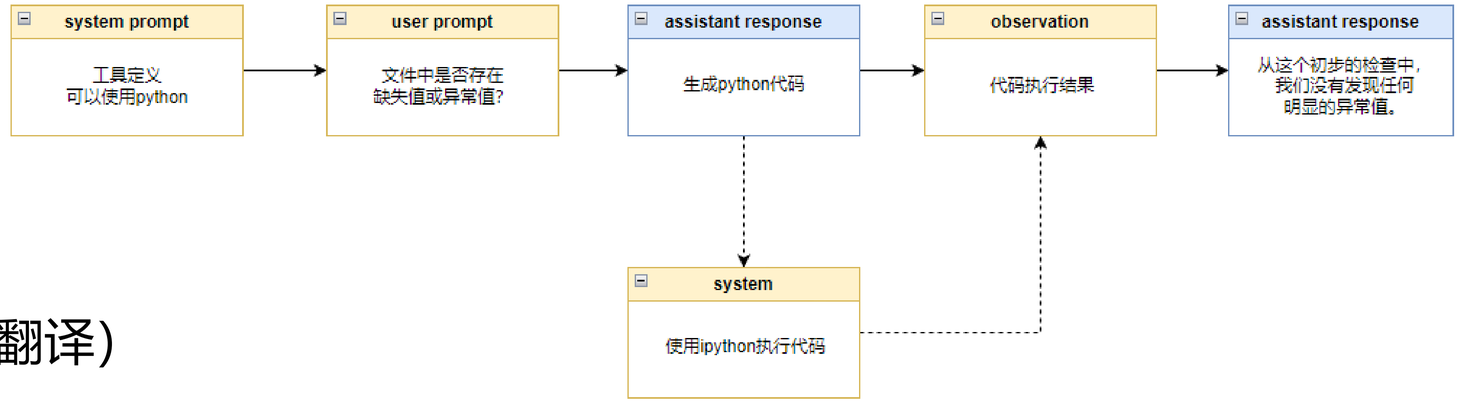

Thought: High memory usage seems to be caused by poor join performance and much inactive memory
Reasoning: Poor joins can be solved by plan optimization
Action: optimize_query_plan
Rewrite logical query …
Optimize query plan …
Add lacking indexes …
D-Bot
Documents
Query Rewriter
Query Planner
Query Executor
System Configuration
知乎 @shikanon

工具学习（工具或函数调用）



大模型 - 代码解释器

1. 系统提示中指明了解决办法：使用python
2. 用户提示中 提出要解决的问题
3. 生成相关代码
4. 调用执行环境运行代码并获得结果
5. 对结果进行自然语言翻译（人类友好翻译）

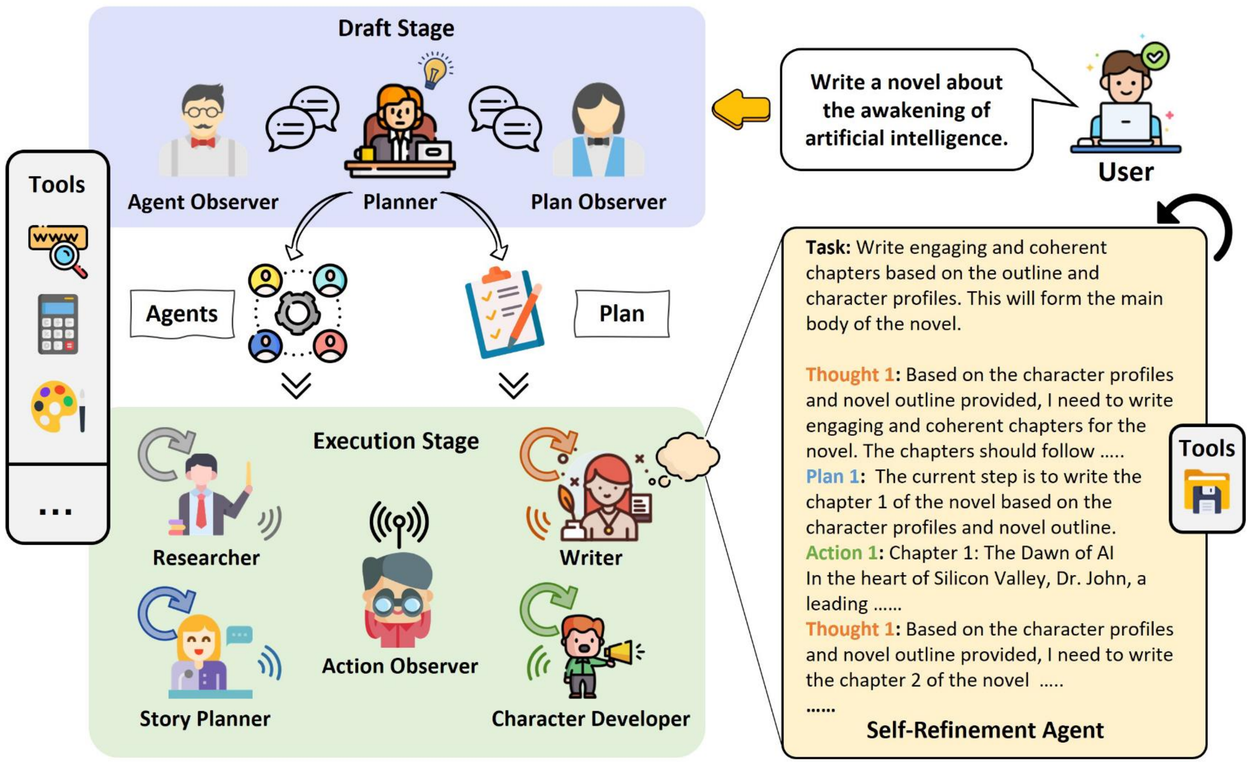


LLM-AGENT 自主智能体的实质

- 三个**臭皮匠**顶个**诸葛亮**：

1. **将复杂任务拆解为小任务，由智能体之间反正协调达成一致，以解决“智力”不足的问题。**
2. **对大模型的智力有一定要求**
3. **可行的LLM为GPT 4**
4. **Agent 是某个模型的角色实例，不是某个模型。进程是程序的实例，它不是一个程序。**

LLM-Agent 详解- 以AutoAgents框架为例



二个阶段

用户：输入待解问题

1. 起草阶段
    1. 三个Agent协作生成新的智能体以及执行计划

2. 执行阶段
    1. 通过促进智能体之间的协作和反馈来完善计划
    2. 交付最终结果

基于智能体协作原则：
通信、协调和共识。
帮助智能体：
1. 共享信息
2. 协调行动
3. 达成共识

LLM-Agent 起草阶段的要求

- 人类群体内部的多样性产生了不同的视角，增强了群体在任务中的表现。（群体能力大于 个体能力简单叠加）
- 三个初始智能体：
    1. 规划者 （根据任务内容生成和完善 智能体团队与执行计划）
    2. 智能体观察者 （提供智能体合理性及智能体与任务匹配度的建议）
    3. 计划观察者 （执行计划的合理性以及 这个合理性与智能体团队、任务的匹配程度的建议）
- 计划阶段：
    1. 产生能够最大化群体潜力的最佳智能体团队和执行计划： 至关重要
    2. 分配智能体角色，以不同的智能体角色设计智能体可提增强效力
- 目标：
    1. 产生的智能体应该表现出多样性以**适应各种任务**。
    2. 智能体和计划的生 成应遵循一定原则，使它们的**角色分配更加合理**。

LLM-Agent 执行阶段

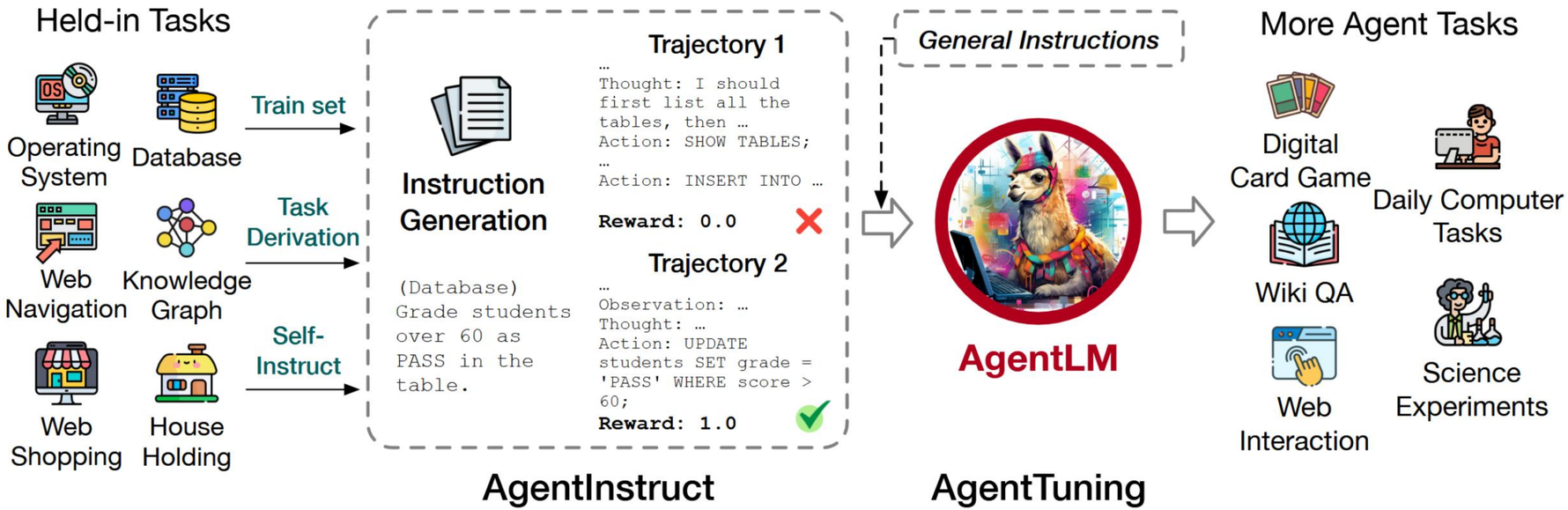
- 多代理之间的通信和合作对有效完成任务至关重要。
- 行动观察者，作为不同智能体的调度者
    1. 向智能体分配不同的任务
    2. 验证每个智能体的执行结果
    3. 根据结果动态调整执行计划
- 作为多智能体之间的通信节点，转发通信。

LLM-Agent 记忆与通信

1. 短期记忆
    1. 单个智能体在自我优化过程中生成的中间思考，计划和执行结果。
2. 长期记忆
    1. 多个智能体之间的沟通，主要是单个智能体执行的任务以及重要反馈信息的汇总。
3. 动态记忆
    1. 服务于有特殊需求的智能体。行动观察者可以访问所有的短期或长期记忆，从中动态提取所需的信息，用于增强单个智能体执行任务的效率。

LLM-Agent 训练智能体模型

- Agent-tuning



LLM-Agent 怎么训练智能体模型

- 通用Agent: GPT4, 表现并不乐观， 梦想：如果在gpt4上用agent-tuning, 是不是更好？ （笑）

- 通用的专用模型，通用：不丢失普通模型的能力； 专用：指定AGENT具有较强能力
- 具有一定的**Agent泛化能力**，在未见过的任务上较未向调模型更好。

- 训练用指令数据： 动作轨迹数据

- 通用Instruct 数据： 传统的指令微调数据

LLM-Agent 用于数据分析

- 我们不可能一步培养成功体操世界冠军
- 可行的路径：

1. 基本的动作示例
2. 动作纠正
3. 战术计划
4. 团队能力培养
5. 个人创新或随机应变能力 （个体素质）

- 个人创新或随机应变能力是长期单项培训后的集大成（涌现），针对大模型，本质是逻辑能力。

LLM-Agent 授人以鱼，不如授之以渔

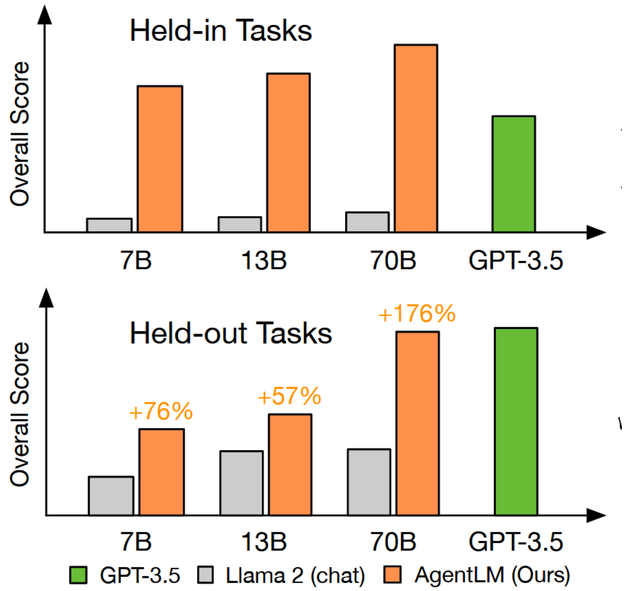
- **人类的误区**

1. 人类总是想抄近路或者认为大模型是一个成熟的从业者，只发命令即达目标
2. 按某数字老板的想法：把用户想象成小白，啥也不懂
3. 大模型是一个有智的小白，先教它而不是命令它
4. AGI中的G来自于人类，生成行动轨迹数据，训练大模型的AGENT能力

关于AGENT用于安全领域

1. 复杂的单步不可能完成的数据查询
    1. 请帮我查询最近的网络攻击趋势， 试想：人怎么完成这个操作的?
    2. 分步、协商
2. 分析主要业务，生成行动轨迹数据 （工作重点）
    1. 任务分析Agent所需 轨迹数据
    2. 子任务所需 轨迹数据
    3. 整体能力靠Agent群体作战完成
3. 训练Agent模型
4. 应用于业务

Agent-Tuning 后能力对比 - 喜与忧



- 喜：
- 能力提升巨大
- 忧： 只能跟ChatGPT 3.5对决
- 期望： 开源大模型的逻辑能力进一步提升

05

大模型最新进展

国内篇 – 国产大模型的进展

- 部分能力接近或达到ChatGPT 3.5水平
- Qwen 72B, DeepSeek 67B, Yi 34B，元象XVERSE 65B 模型
- 更强的中英文能力
  - Yi 34B 翻译能力
- 逻辑能力进一步增强

国外篇 -MOE

- 专家混合模型：西北风人工智能 这是一家总部位于巴黎的开源模型初创公司，它发布了最新的大型语言模型 (LLM) MoE 8x7B。
- 特点：

1. 相当于60B左右的逻辑能力
2. 推理成本相当于14B左右
3. 同时只激活两路专家

- 本质： 大模型在执行具体任务时只激活部分子网络参数。

THANKS

安天冬训营 wtc.antiy.cn

执行体治理赋能与大模型辅助

北向守望