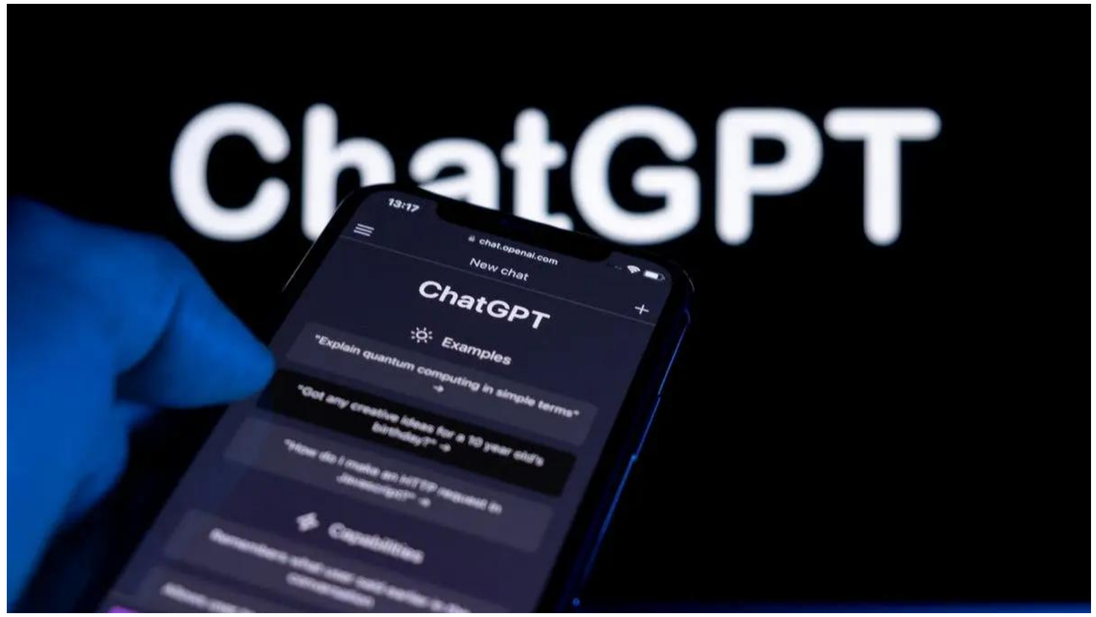
Agent新技术新实践

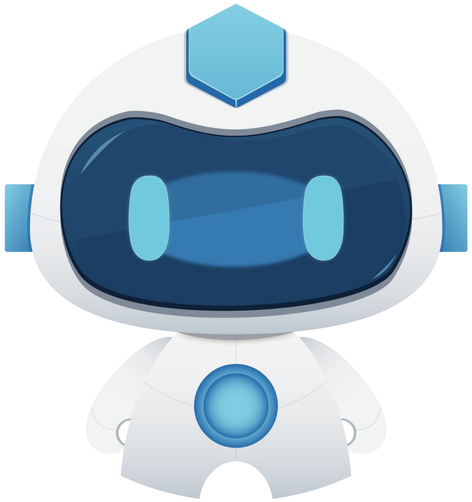
王元 CEng

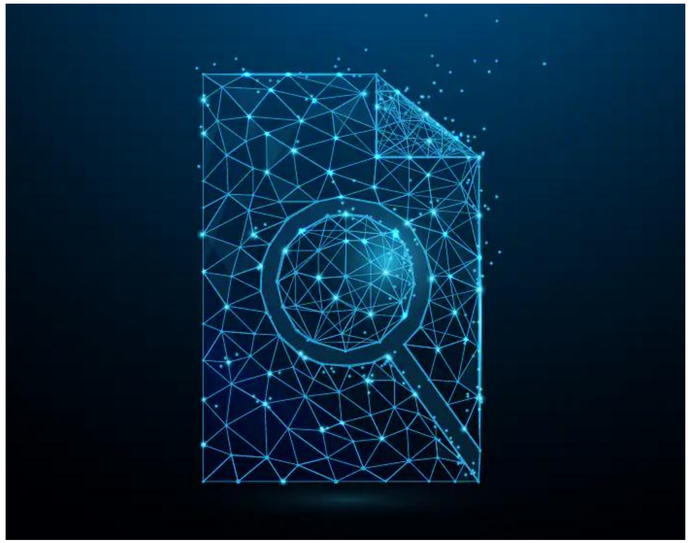
LLM出圈了



- ChatGPT技术内核：生成式AI
- 本次分享，AI Agent时代下的新方法和新实践

聊技术前，先捋捋场景



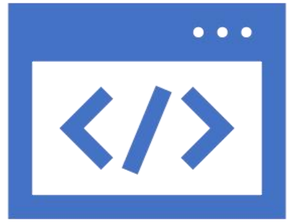


- 私域问答机器人
- 无代码数据分析与报表
- 文档智能/企业内网搜索

UI -> 交互式；后台 ->多轮
UI -> 交互式；后台 ->多轮
UI ->线性执行；后台 ->单/多轮

新的技术栈

私域数据



多模态

测试问题

语义缓存

私域数据

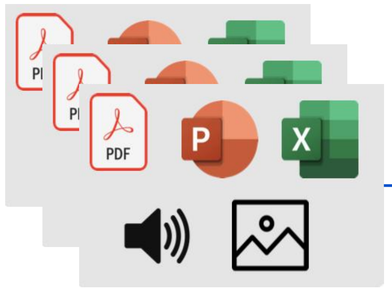
问题：如何接入私域数据？

- 私域数据常常大于LLM原生的context length
- 全量放入context可能不是最优方式

步骤：

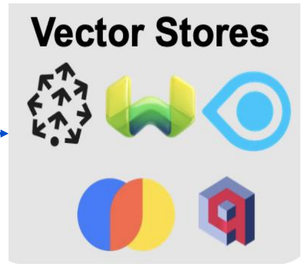
- 私域数据的分割，存入向量数据库；
- 向量数据库召回，召回结果作为LLM的context；

私域数据–分割



M个文件

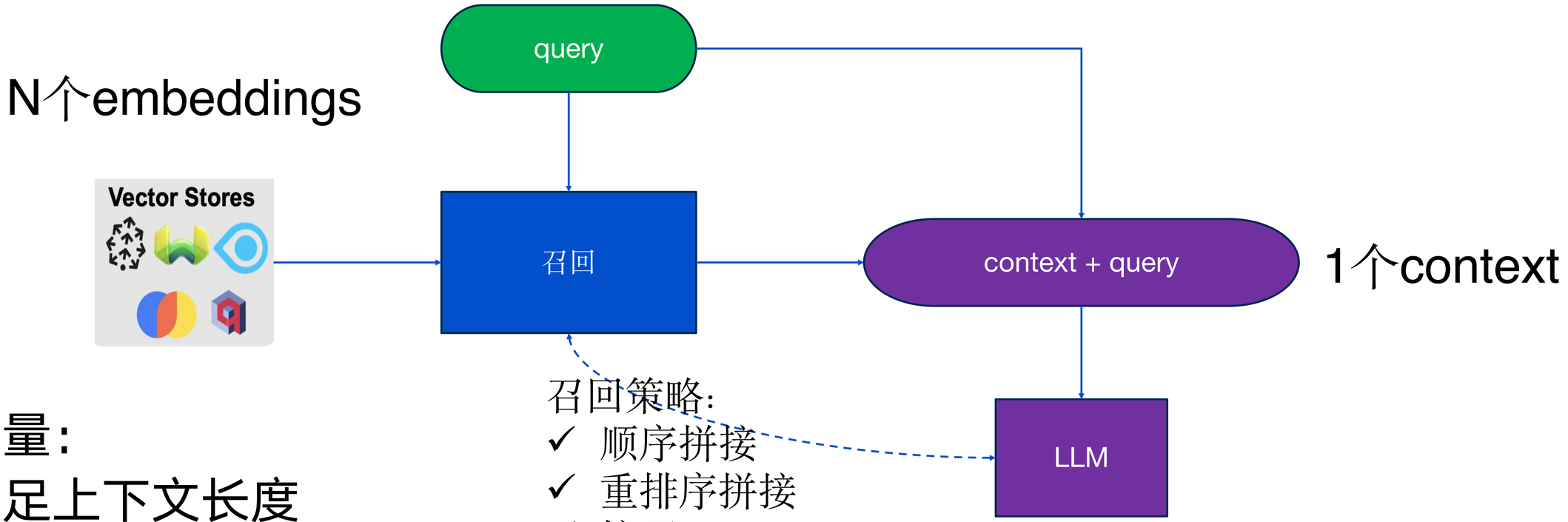
分割策略:
- ✓ 分隔符
- ✓ 均匀分割
- ✓ 树结构



N个embeddings

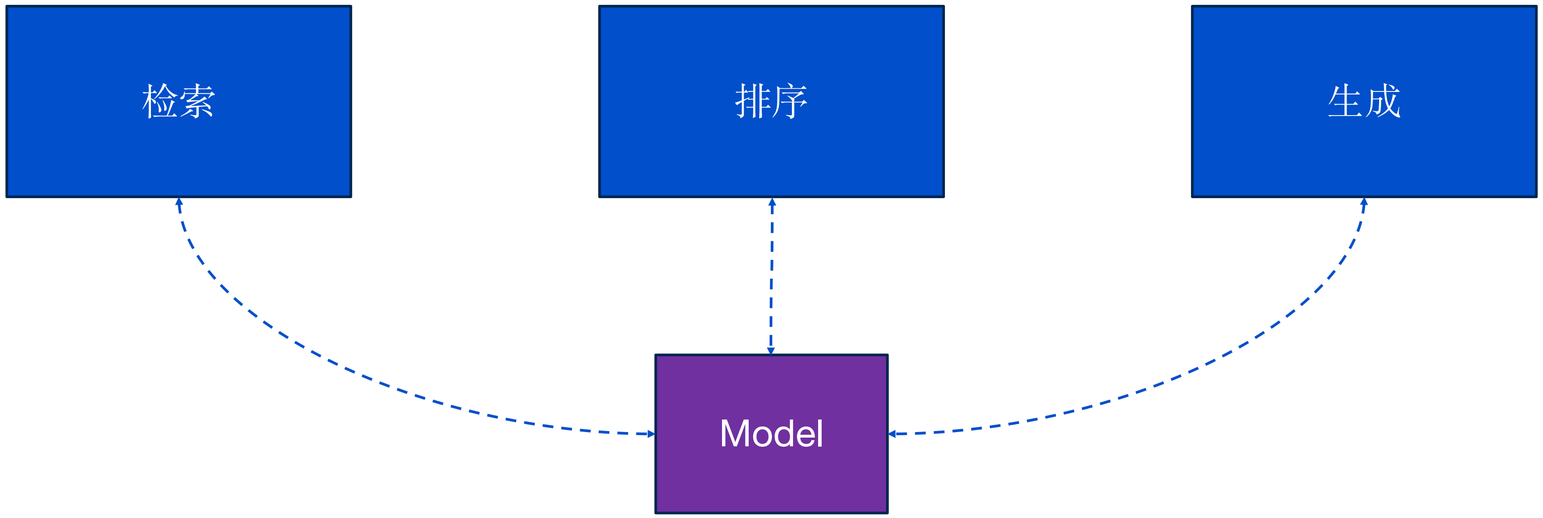
- 考量:
- 分割颗粒度
- 召回准确率和速度
- 维护成本

私域数据–召回



- 考量：
- 满足上下文长度
- 信息不丢失
- 语义流畅

私域数据–召回



私域数据–评估

传统检索指标：

- MRR
- NDCG等

端到端：

- 答案与标准答案的相似度

LLM - 无需标注（RAGAS）：

- **上下文与问题的相关性**
- 答案与上下文的相关性
- 答案与问题的相关性

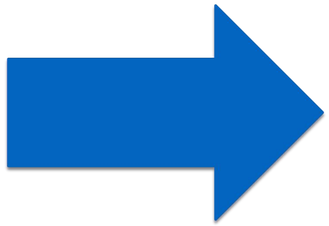
私域数据–经验

- 召回方法，分割大小，top-K是影响性能的主要超参
- 向量空间的语义搜索，有时会有不靠谱的情况
- 结合传统文字BM25搜索会带来一些帮助
- 召回数据排序，decoder-only比encoder-decoder更敏感
- 把query同时放在召回数据前和后，有助于LLM回答
- 召回可以有更多的考量维度，不仅限于语义相似度

角色框架

给LLM加上方法论：

- 了解背景
- 任务拆分，逐个击破
- 自省
- 行动（感知和改变环境）



基础模块：

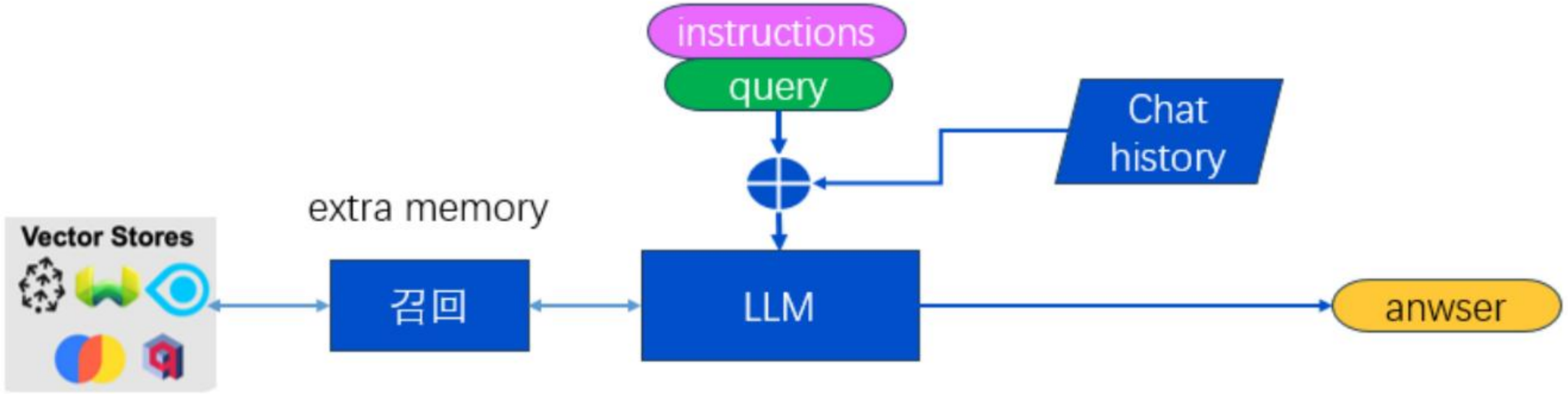
- 角色设定
- 规划模块
- 内存模块
- 动作模块

角色框架–优势

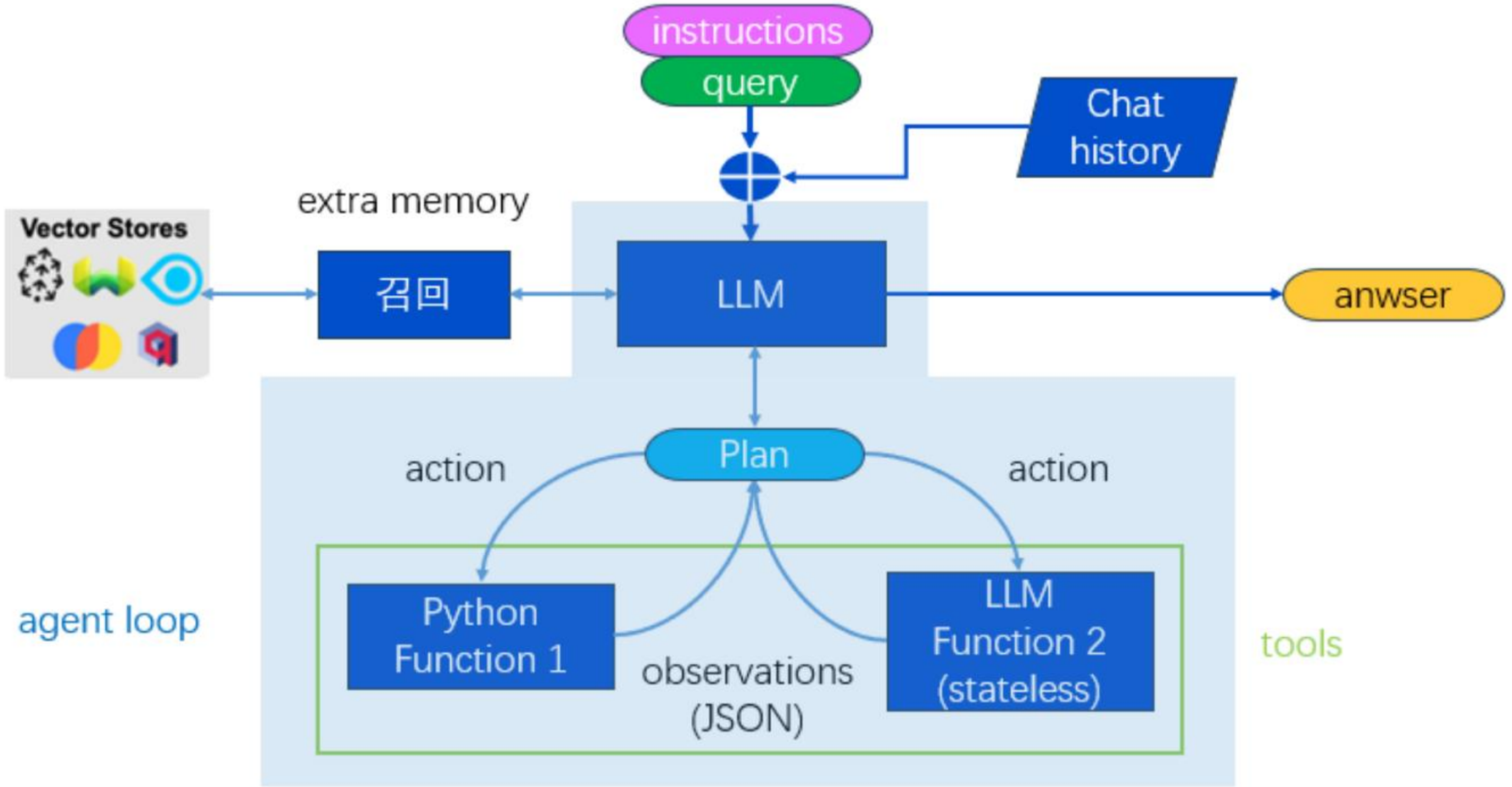
Agent机制的好处：提升LLM处理复杂任务的能力，扩大了应用范围

- 将LLM从无状态变为有状态；
- 缓解context length有限的外部框架；
- 赋予LLM自主调用外部工具的能力；
- 使LLM获得拆解任务的思维，将复杂问题分拆逐个解决；
- 简化手写FSM；
- 多个agent相互合作成为可能，即群体智慧（multi-agent）

角色框架


instructions
query
Chat history
extra memory
Vector Stores
召回
LLM
anwser

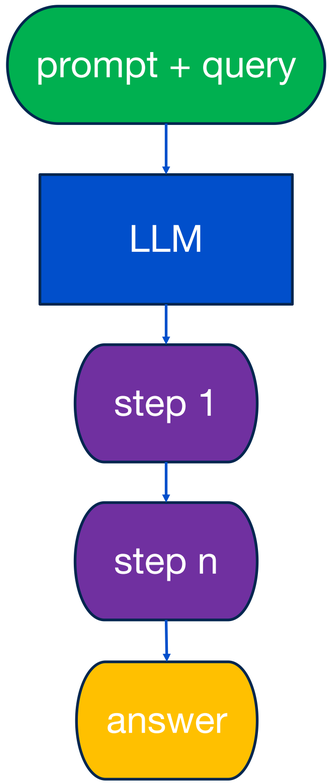
角色框架


instructions
query
Chat history
extra memory
Vector Stores
召回
LLM
anwser
action
Plan
action
agent loop
Python Function 1
observations (JSON)
LLM Function 2 (stateless)
tools

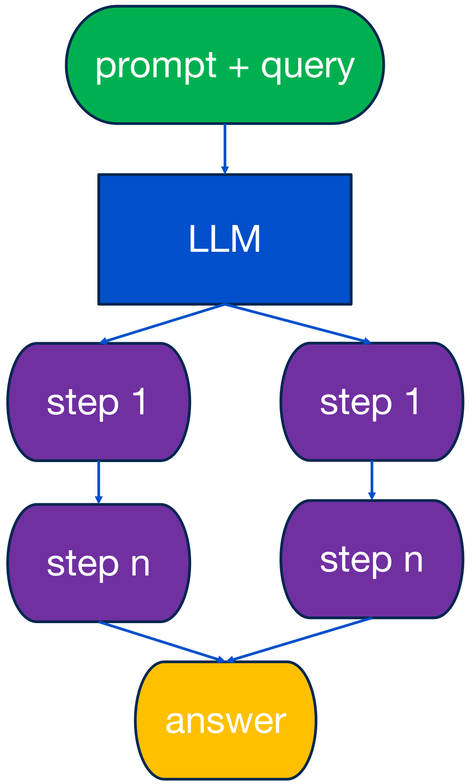
角色框架–规划

开环系统：i步的执行结果**不影响**i+1步的规划

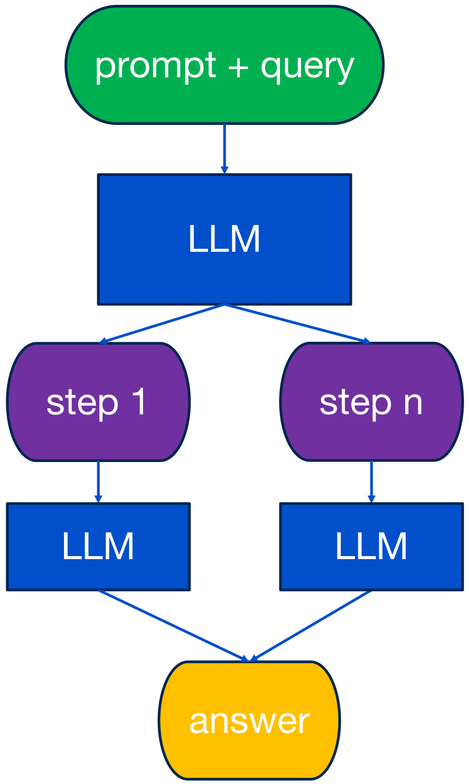
COT



COT-SC



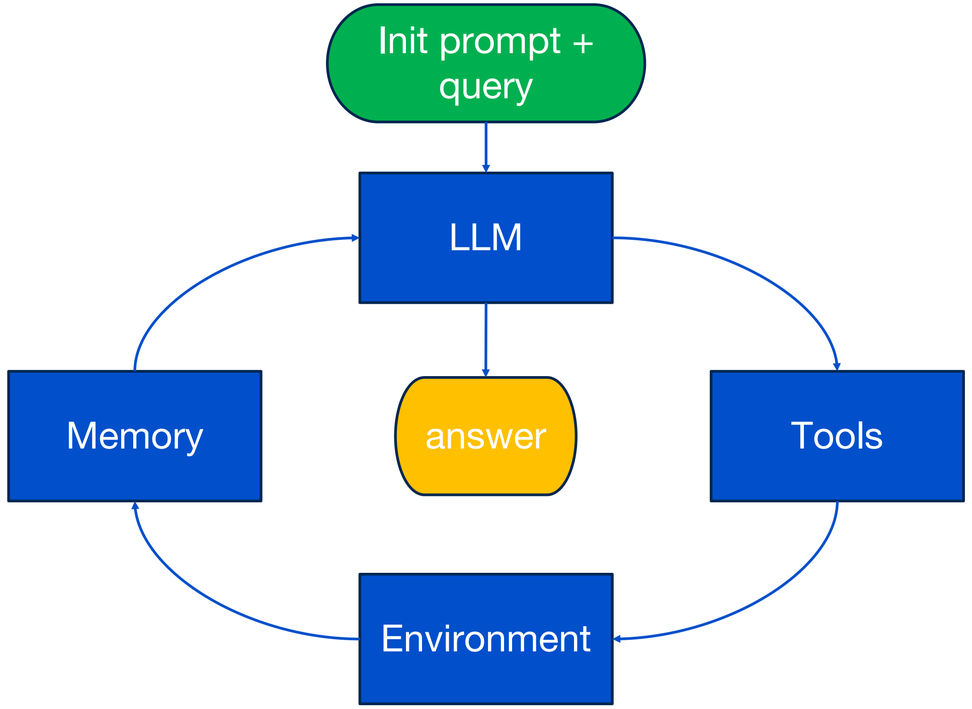
ReWOO



角色框架–规划

闭环系统：

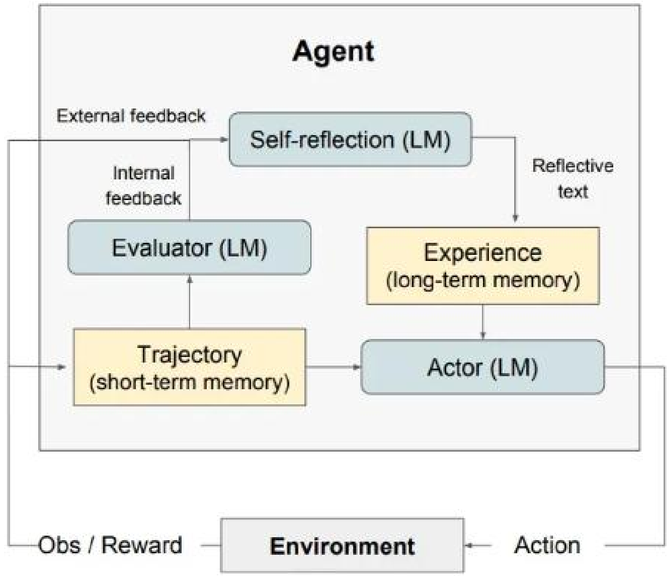
- i步的执行**影响**i+1步的规划
- 多轮调用LLM
- 使用外部工具（tools）
- 内存使用



角色框架–规划

一些代表性的闭环系统：

- Self-Ask：COT + follow-up Q + tools
- ReAct：局部plan – 执行 – 观察
- Plan-n-Solve：全局plan + ReAct
- Reflextion：群体智能雏形(3个LLM)



角色框架–内存

内存种类：短期记忆（上下文），长期记忆（向量数据库）
内存形式：向量，字符串（text/json/list），数据库
内存召回指标：相关性，时效性，重要性
内存写入：append，去重，覆写（overflow或者简化）

角色框架–行动

函数类型：人类函数，LLM函数
函数选择: 类似RAG检索
函数入参提取：string -> json
函数返回值：string <-> json/xml

角色框架-性能评估

主观评估：

- 人类打分；
- 图灵测试；

数据评估：

- 端到端任务指标，KIE准确率，回答GT相似度；
- 经典数据集：Alfworld，HotpotQA，FEVER，HumanEval
- benchmark：AgentBench，ToolBench等multi-task集合

系统评估：

- 报错次数
- 平均LLM calls
- 迭代轮次（耗时）

角色框架–经验

一些尝试：
ReAct
Plan-n-Execute
Self-Ask

一些思考：
Plan-n-Execute > ReAct > Self-Ask
工具选择有时会错 – 工具list最好不要固定一个顺序
中间结果生成有时会错 – GPT4 turbo会改善
现实情况：Agent和手写FSM混合使用

角色框架–挑战

商业：

- Agent多次调用LLM，除了能力边界提升，成本也提高了
- Tools的使用会带来额外成本

技术：

- 系统鲁棒性
- 系统时延

安全：

- Agent自主创建子目标并执行，可能会有潜在安全风险
- Agent机制会激发AI产生自我意识吗？

角色框架–推荐

关键论文：

- COT （https://arxiv.org/abs/2201.11903）
- ReAct （https://arxiv.org/abs/2210.03629）
- Reflextion（https://arxiv.org/abs/2303.11366）
- Agent综述（https://arxiv.org/abs/2308.11432）

工程代码：

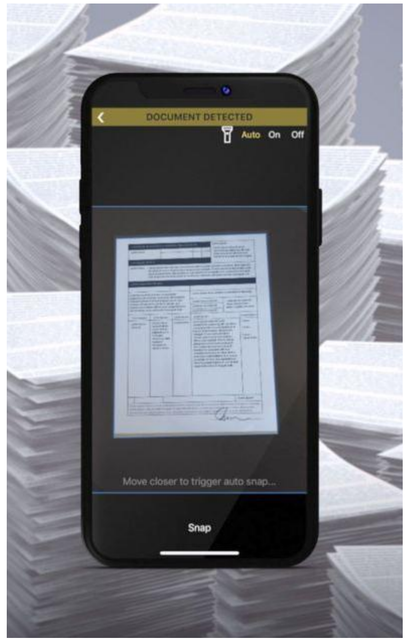
ReAct：https://github.com/ysymyth/ReAct

Reflextion：https://github.com/noahshinn024/reflexion/tree/main

Langchain：https://github.com/langchain-ai/langchain

Marvin：https://github.com/PrefectHQ/marvin

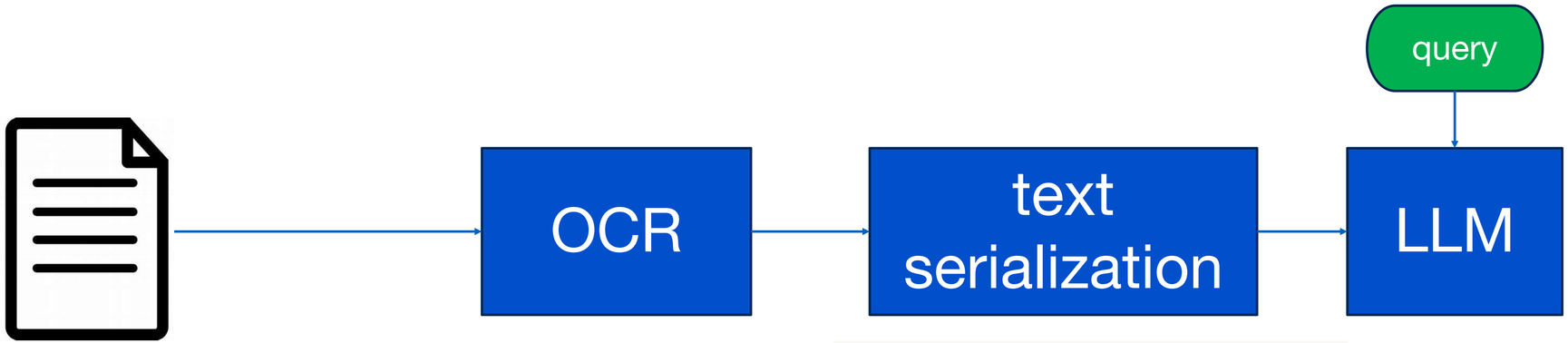
多模态



场景不同：自然图片 ->文字密集型文档

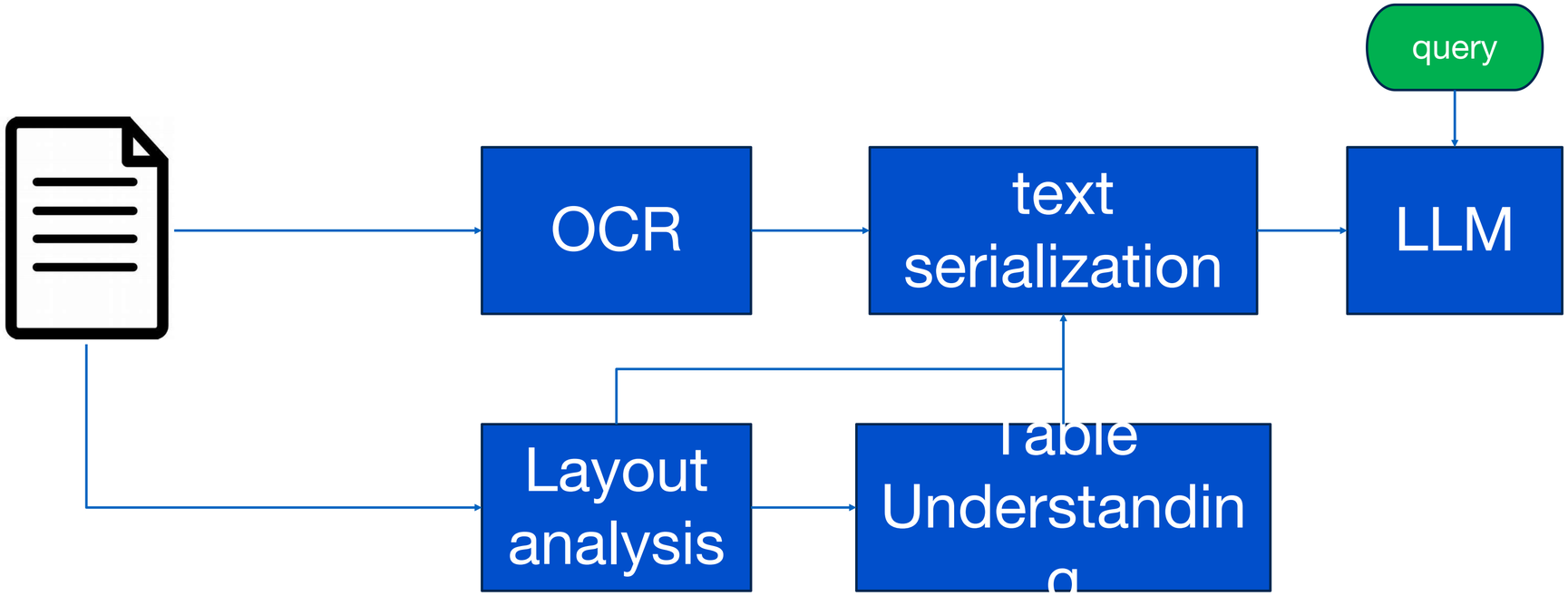
问题：对于多模态LLM，OCR是否需要？

多模态–baseline



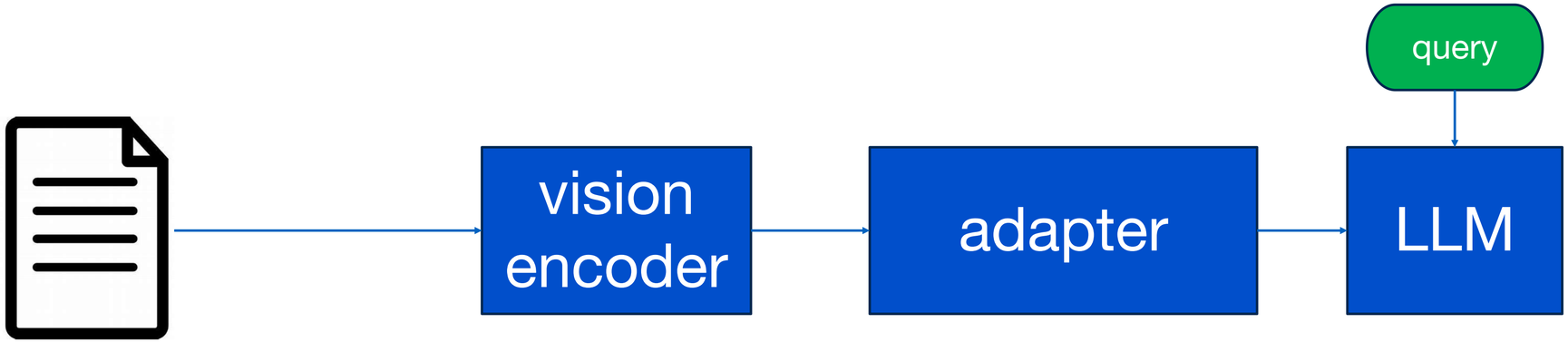
难点 – text序列化非常难以泛化，错误的排序直接影响LLM做QA或KIE

多模态–baseline



Baseline on steroids: 模型越堆越多，系统越搞越复杂，泛化性依旧成问题

多模态-方案1



微调形式：高效微调

挑战：

- 开源LLM/MLLM适用，闭源不适用（没API）
- 分类任务效果好，KIE效果不好
- 预训练的vision encoder多数分辨率低，且不是multi-scale
- 微调不是端到端的，是2-stage微调

多模态-方案1

Method	Regular			Irregular						Occluded		Others				Avg.
	IIIT5K 3000	SVT 647	IC13 857	IC15 1811	SVTP 645	CT80 288	COCO 9896	CTW 1572	TT 2201	HOST 2416	WOST 2416	WordArt 1511	IAM 3000	ReCTS 3000	CAR-A 3784	
BLIP-2 $OPT_{6.7b}$	76.63	80.22	82.96	69.35	73.33	76.04	48.68	61.70	63.52	57.00	68.00	**74.26**	38.00	0	6.21	58.40
BLIP-2 $FlanT5_{XXL}$	76.60	83.77	86.35	70.84	73.80	80.90	50.10	64.50	65.74	57.16	68.34	73.79	40.50	0	17.73	60.68
OpenFlamingo	68.20	74.19	74.10	63.61	73.49	67.71	45.52	53.94	57.84	48.18	60.55	60.62	45.53	0	3.57	53.14
LLaVA	64.10	67.70	70.71	58.97	62.95	61.11	41.71	50.89	52.43	47.39	55.26	62.61	50.40	0	1.40	49.84
MiniGPT4	48.00	50.39	48.89	42.19	50.39	57.29	26.25	41.86	40.57	34.52	41.06	51.42	28.90	0	1.69	37.56
mPLUG-Owl	74.43	77.74	82.15	65.21	72.71	81.94	50.42	68.64	68.11	47.81	60.60	72.73	42.53	0	40.20	60.35
Supervised-SOTA	**96.63**	**93.04**	**96.73**	**85.70**	**89.30**	**89.93**	**64.42**	**78.57**	**80.13**	**73.10**	**81.58**	72.49	**91.24**	**94.77**	**95.53**	**85.54**

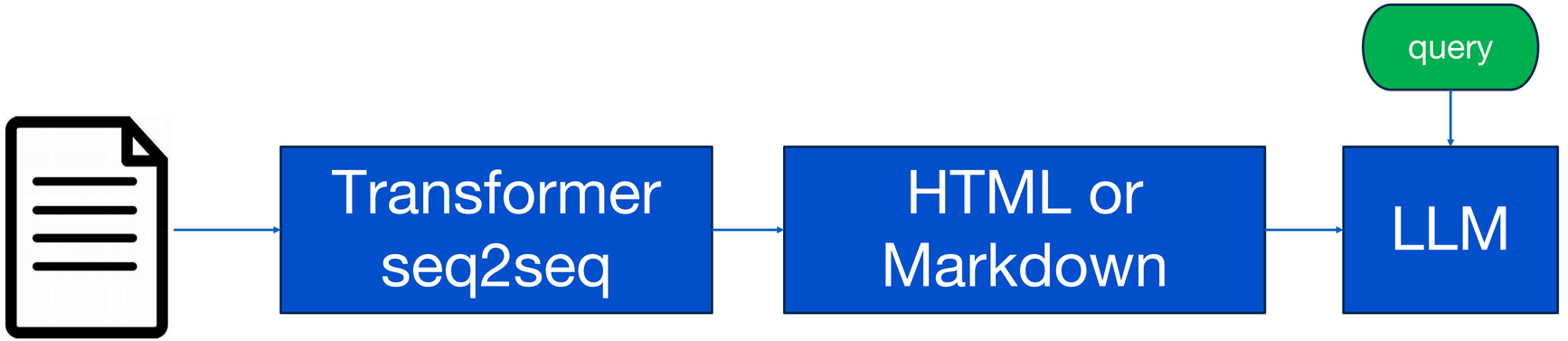
Method	VQA								KIE			HME100K	Avg.
	STVQA 5000	OCRVQA 5000	TextVQA 5000	DocVQA 5349	InfoVQA 2801	ChartQA 1250	ESTVQA(En) 5000	ESTVQA(Ch) 5000	FUNSD 588	SROIE 2503	POIE 6321	5000	
BLIP-2 $OPT_{6.7b}$	13.36	10.58	21.18	0.82	8.82	7.44	27.02	0.08	0.00	0.00	0.02	0.00	9.41
BLIP-2 $FlanT5_{XXL}$	21.70	30.74	32.18	4.86	10.17	7.20	42.46	0.04	1.19	0.20	2.52	0.04	16.19
OpenFlamingo	19.32	27.82	29.08	5.05	14.99	9.12	28.20	0.26	0.85	0.12	2.12	0.00	13.02
LLaVA	22.08	11.36	28.86	4.49	13.78	7.28	33.48	0.16	1.02	0.12	2.09	0.04	12.00
MiniGPT4	14.02	11.52	18.72	2.97	13.32	4.32	28.36	0.10	1.19	0.04	1.31	0.00	9.59
mPLUG-Owl	29.26	28.62	40.28	6.88	16.46	9.52	**49.68**	0.44	1.02	0.64	3.26	0.18	18.44
Supervised-SOTA	**69.60**	**68.10**	**73.67**	**90.16**	**36.82**	**70.5**	43.26†	**43.26**†	**93.12**	**98.70**	**79.54**	**64.29**	**72.75**

Source: https://arxiv.org/abs/2305.07895

KIE任务，MLLM效果普遍差强人意

期待更多开源高分辨率多模态原生大模型的出现！

多模态-方案2



- 解决了text序列化问题
- 开源和闭源LLM都适用
- 对于LLM，是in-context learning，LLM本身不微调
- 预训练Seq2seq模型分辨率高，模型小，只需微调Seq2Seq模型
- 缺点：GT标注较难获取，有标注成本

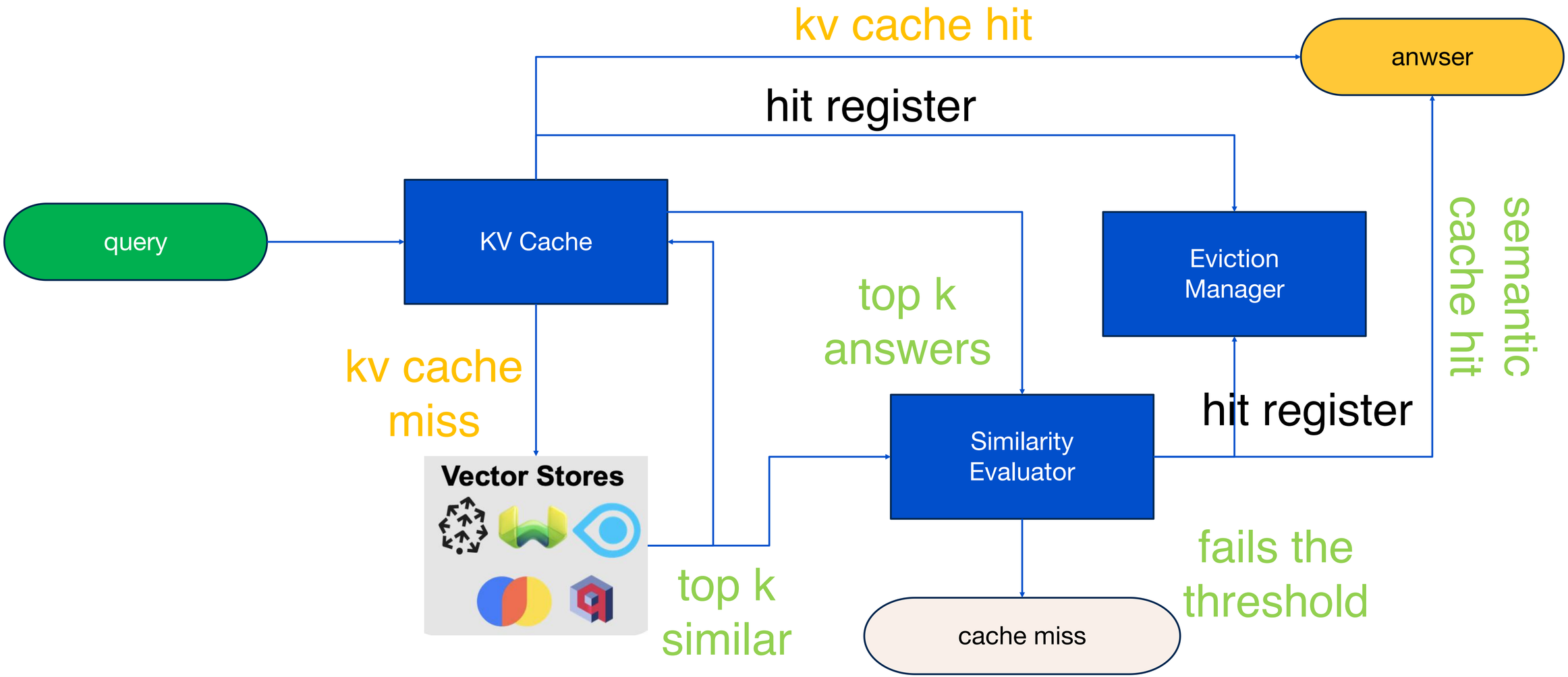
语义缓存

LLM服务很贵，以GPT4为例：

- GPT4接口调用成本很高，千万美元/年
- 接口平均响应时间为3-6秒

语义缓存技术可以有效缓解上述挑战

语义缓存


kv cache hit
anwser
hit register
query
KV Cache
Eviction Manager
semantic cache hit
top k answers
kv cache miss
hit register
Vector Stores
Similarity Evaluator
top k similar
fails the threshold
cache miss

语义缓存

注意事项：

- Hit ratio vs search accuracy
- 高速实现attribute filtering， 推荐支持hybrid search的向量数据库
- 缓存一致性
- 并行化支持 – eviction manager的进程安全
- 持久化和故障恢复

统一缓存设计

- Key定义
- 表结构合理划分
- Eviction manager交互接口

推荐：GPTCache

测试问题

LLM应用让测试变得更有挑战

测试数据

- On-topic similarity
- Off-topic rejection
- Moderation
- Prompt injection
- Hallucination

模型打分

- 人工收集gold responses
- 模型自动打分 – 需要研发专门的打分模型，OR
- LLM as Evaluator – 大模型prompting来打分

THANKS

软件正在重新定义世界

Software Is Redefining The World