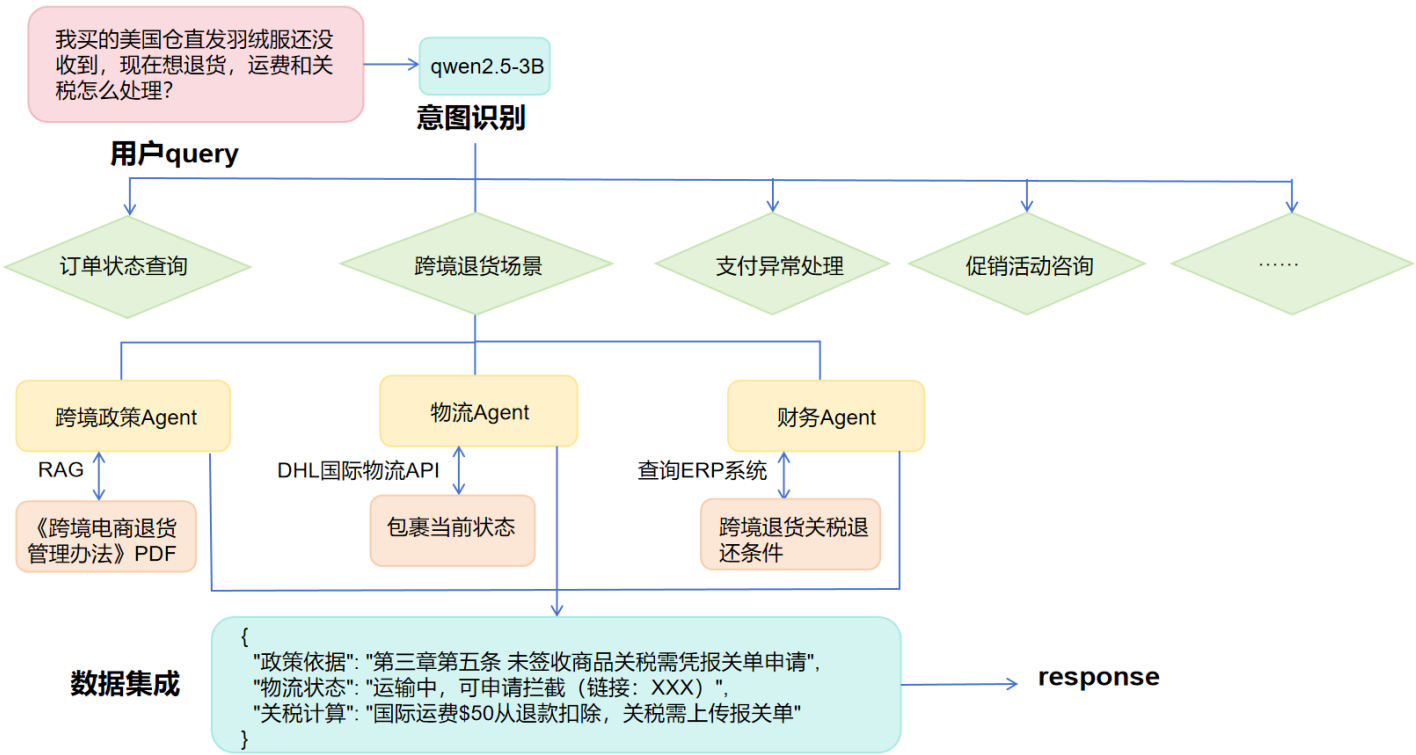
Agentic RAG案例分析



✨ 上图案例的回复：

“您的美国仓直发羽绒服目前仍处于运输中（物流单号：US20240715-001，预计 3 天后抵达您所在城市）。根据《跨境电商退货管理办法》第三章第五条规定，未签收商品可申请物流拦截退货，国际段运费（$50）将从退款中扣除，关税部分需您在退货时提供原始报关单照片（上传入口：订单详情页> 退货申请 > 关税材料）。

如需立即处理，可点击此处申请物流拦截 <操作链接>，拦截成功后系统将自动生成退货单（无需签收商品）。退款预计在仓库确认商品状态无误后 15 个工作日内原路返回，关税退还流程同步启动（预计额外 3 个工作日）。

如需人工复核或修改地址，可直接回复 “转接客服”，当前客服专员已获取您的订单详情，将优先为您处理。”

这是一个电商场景的案例。

用户query：“美国仓羽绒服未签收，退货关税怎么处理？”

意图识别模块判定为“跨境未签收退货”场景。

Router Agent分配任务：

- 给**跨境政策Agent**：检索《跨境电商退货管理办法》第5条（关税退还条件）。
- 给**物流Agent**：调用DHL API获取包裹状态（运输中，可拦截）。
- 给**财务Agent**：查询ERP系统中“未签收退货的关税计算规则”。
- **不做**：不直接处理政策文本内容、物流JSON数据或财务公式计算，仅决定“谁该做什么”。

数据集成：三个Agent返回的数据格式不同（政策PDF段落、物流API的JSON、财务系统的SQL结果），由独立的**数据集成模块**统一为：

```
代码块
1  {
2    "政策依据": "第三章第五条 未签收商品关税需凭报关单申请",
3    "物流状态": "运输中，可申请拦截（链接：XXX）",
4    "关税计算": "国际运费$50从退款扣除，关税需上传报关单"
5  }
```

LLM合成：根据上述结构化数据，LLM生成自然语言response，Router Agent不参与内容生成，仅确保数据按正确顺序输入LLM。

在本案例中，Agentic RAG的**优点**在于：用户获得的不再是 “碎片化政策条款”，而是 “包含具体操作路径的解决方案”；系统不再是 “被动响应查询”，而是 “主动诊断场景并提供最优路径”。

不过Agentic RAG并非尽善尽美，依据最近阅读的论文：Why Do Multi-Agent LLM Systems Fail? 分析一下本案例可能会存在的实际**问题**：

1. **不遵守任务规范**

- **问题**：Agent未严格遵循用户查询的具体要求，导致解决方案偏离核心需求。
- 若用户明确要求“提供未签收商品的关税退还流程”，但跨境政策Agent错误引用“已签收商品退货政策”，导致回复中包含“需先签收再申请”的错误步骤，违反用户对“未签收场景”的任务规范。

2. **不遵守角色规范**

- **问题**：Agent越权或混淆角色职责，破坏分工协作逻辑。
- 财务Agent本应负责运费计算，却越权调用物流API获取包裹状态，导致物流Agent的职责被架空，流程混乱（如财务Agent错误判断“包裹已签收”，错误计算关税）。

3. **步骤重复**

- **问题**：Agent重复执行已完成的任务，浪费计算资源并延长响应时间。
- **案例映射**：物流Agent已返回“包裹运输中，可拦截”，但协调Agent未记录状态，再次触发物流Agent重复查询同一包裹状态，导致响应延迟10秒以上。

4. **对话历史丢失**

- **问题**：Agent在多轮对话中丢失关键上下文，导致逻辑断裂。
- **案例映射**：用户补充“退货时需要保留原包装吗？”，但Agent因丢失前期“未签收拦截”的对话历史，错误引用“已签收退货需保留包装”的政策，导致回复矛盾。

……

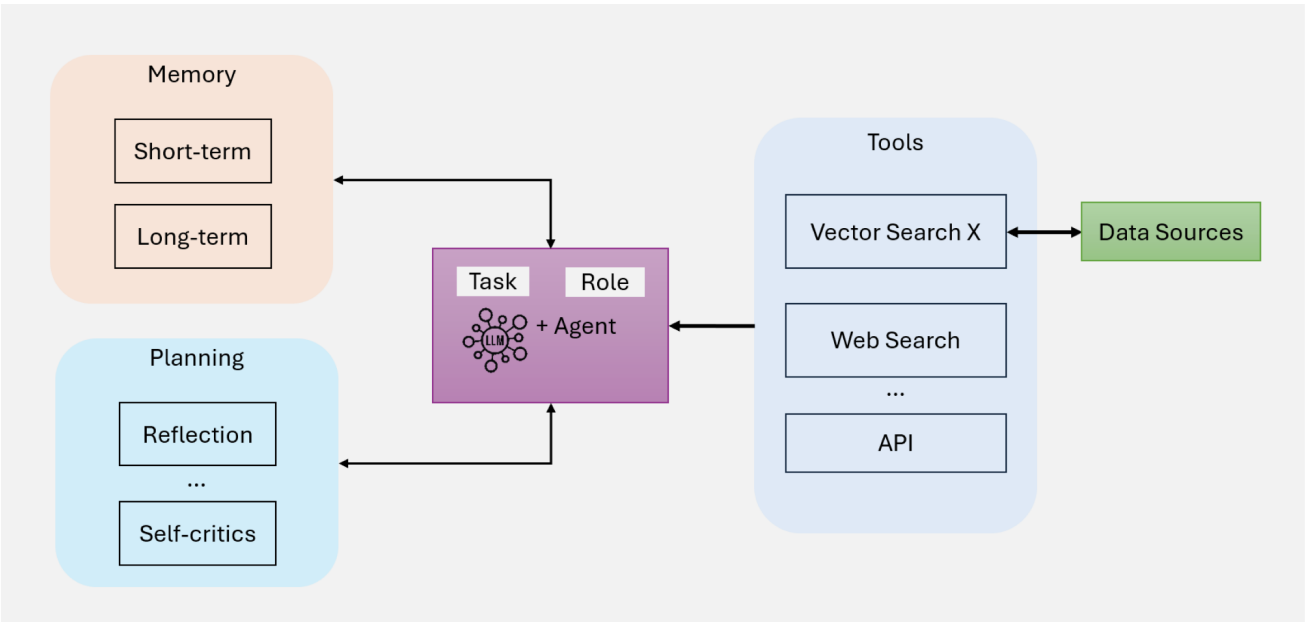
诸如此类的问题仅通过改进Prompt（如更明确的角色定义）只能部分缓解，但无法根治，需要一些结构性策略（如标准化通信协议、强化验证机制）。

Agentic 样式

Agent通常由四部分组成：

- **LLM**：作为Agent的主要推理引擎和对话接口，负责解释用户查询，生成响应，并保持连贯性
- **记忆（短期和长期）**：在交互过程中捕捉上下文和相关数据。短期记忆跟踪即时对话状态，而长期记忆存储积累的知识和Agent经验
- **规划（反思和自我批评）**：通过反思、查询路由或自我批评指导Agent的迭代推理过程，确保有效地拆分复杂任务
- **工具（向量搜索、网络搜索、API 等）**：扩展Agent的能力，使其不仅限于文本生成，还能够访问外部资源、实时数据或专门的计算

> 更多关于Agent的知识见4 Agent篇



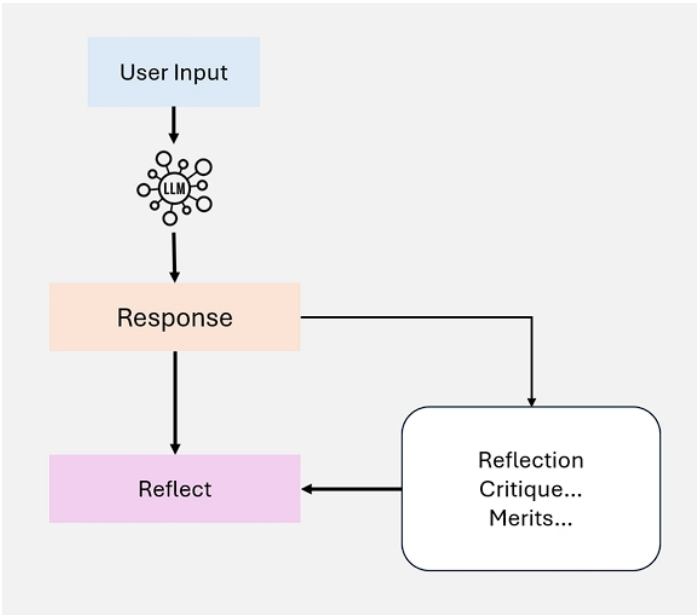
Reflection

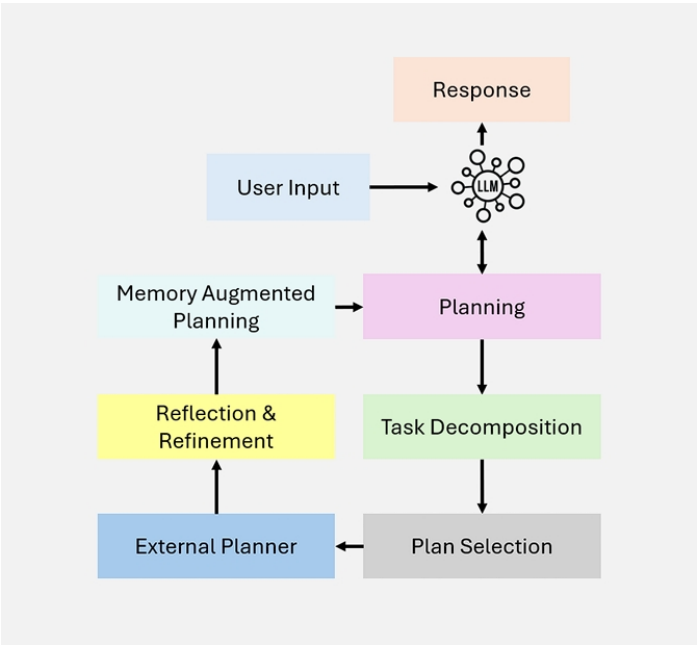
Agent迭代地评估和改进其输出，以识别和解决错误、不一致性和改进空间，提高在代码生成、文本生成和问题回答等任务中的性能。在实际应用中，反思涉及促使Agent对其输出进行正确性、风格和效率的批判，并将这些反馈纳入后续的迭代中。外部工具如单元测试或网络搜索，可以进一步增强这个过程，验证结果并突出差距。

通过不断地迭代精炼，可以提高多步骤推理任务的准确性。在多Agent系统中，反思可以涉及不同的角色，例如一个Agent生成输出，而另一个Agent对其进行批判，促进协作改进（类似强化学习的Actor-Critic算法）。例如，在法律研究中，Agent可以通过重新评估检索到的案例法来迭代地改进回答，确保准确性和全面性。

Planning

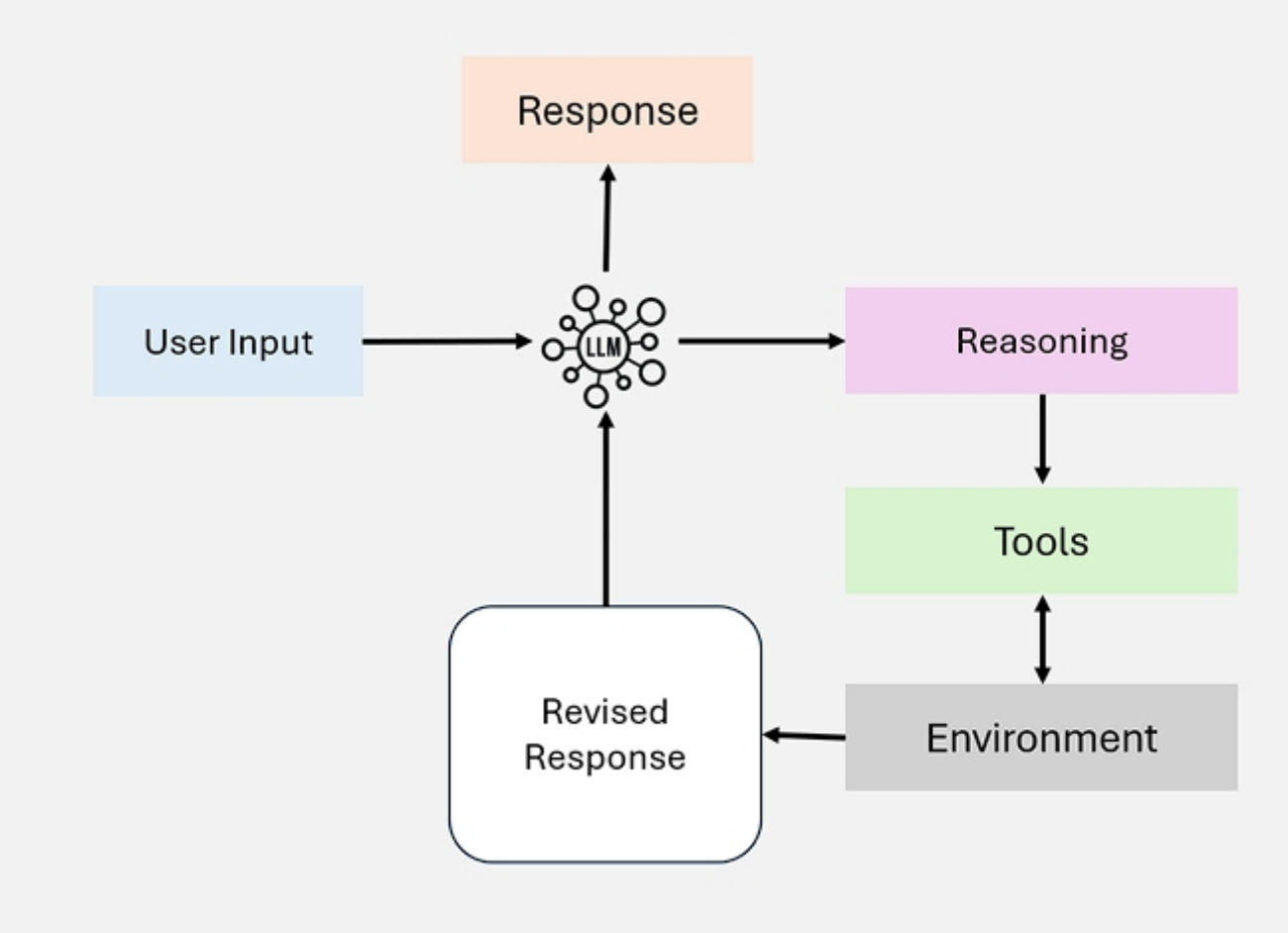
规划使Agent能够自主地将复杂任务分解为更小、可管理的子任务，创建结构化的工作流程和任务序列，高效地解决问题。目标在于通过分解任务促进多步骤推理，通过优化任务优先级减少计算开销。例如，一个财务分析系统规划数据检索任务，以评估风险并提供建议。与反思等确定性工作流程相比，规划可能产生较不可预测的结果。





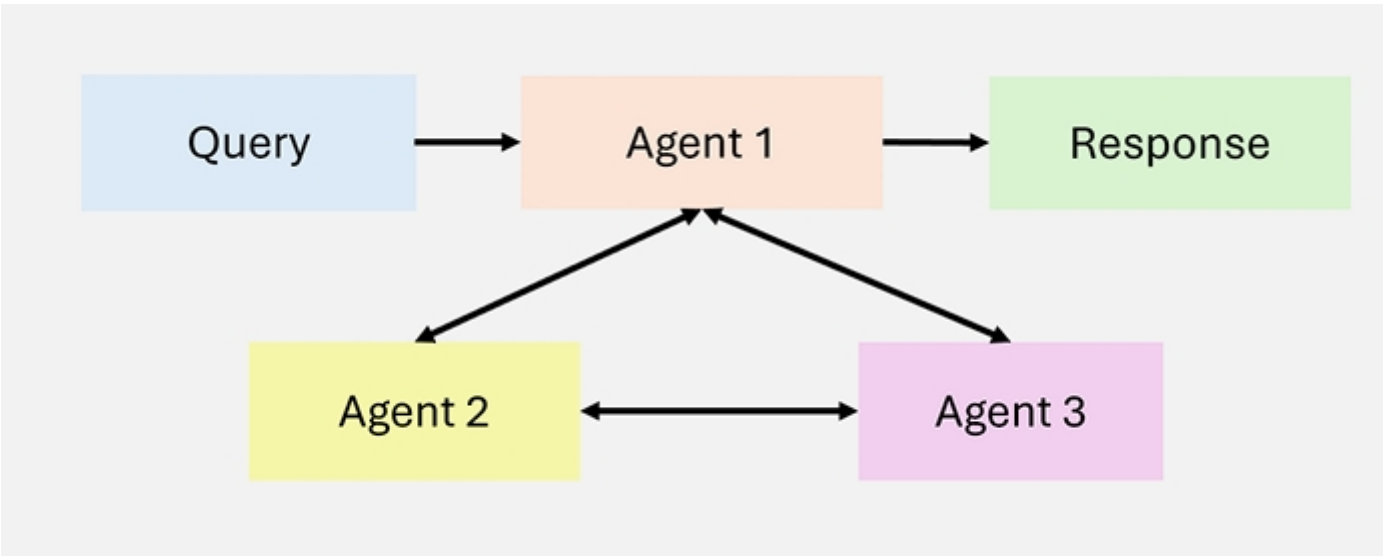
Tool Use

Agent与外部工具、API和知识库互动来扩展其能力。目标在于将系统功能扩展到预训练知识之外，通过集成外部资源实现特定领域的应用。通过将工具动态集成到工作流程中，Agent可以适应复杂任务并提供更准确和与上下文相关的输出。例如，法务助理Agent人从合同数据库中检索条款，并应用特定领域的规则进行合规性分析。



Multi-Agent

多个Agent协同工作以解决复杂任务，Agent之间进行通信和共享中间结果，确保整体工作流程高效和连贯。通过将子任务分配给专门的Agent，这种模式提高了复杂工作流程的可扩展性和适应性。每个Agent都有自己的记忆和工作流程，可以包括使用工具、反思或规划，实现动态和协作的问题解决。例如，在软件开发中，不同Agent分别负责前端、后端、测试、算法。



提升Agentic性能的方法

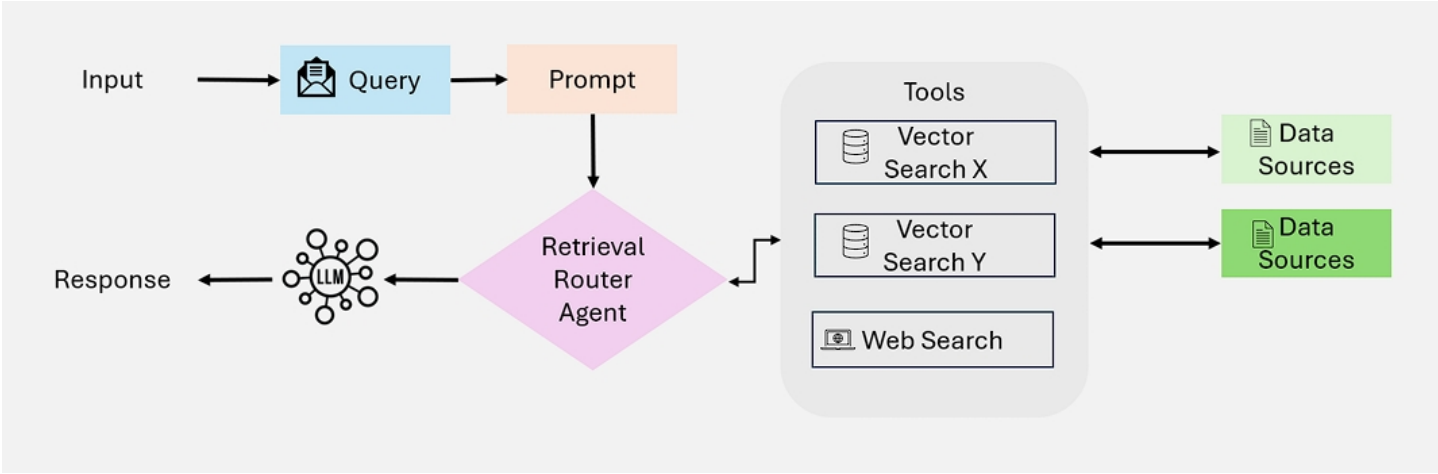
- **Prompt Chaining**：将复杂任务分解为多个步骤，每个步骤都建立在前一个步骤的基础上。这种结构化方法通过在继续前进之前简化每个子任务来提高准确性。然而，由于顺序处理，它可能会增加延迟。该方法适用于逐步推理，每个子任务都对最终输出有贡献的场景，例如数学推理。
- **Routing**：对输入进行分类，并将其引导到适当的专门提示或处理过程。这种方法确保不同的查询或任务被单独处理，提高了效率和响应质量。适合不同类型的输入需要不同处理策略的场景，确保每个类别的最佳性能，例如智能客服。
- **Parallelization**：并行化将一个任务分解为同时运行的独立的进程，从而减少延迟并提高吞吐量。它可以分为分段（独立子任务）和投票（多个输出以提高准确性）两种类型，例如内容审核。
- **Orchestrator-Workers**：利用中央协调模型动态地将任务分解为子任务，分配给专门的工作模型，并编译结果。与并行化不同，它能够适应不同的输入复杂性。
- **Evaluator-Optimizer**：评估器-优化器工作流程通过生成初始输出并根据评估模型的反馈进行改进，迭代地提高内容质量。例如Actor-Critic算法。

Agentic RAG分类

Single-Agent Agentic RAG

Single-Agent Agentic RAG 作为一个集中的决策系统，负责管理信息的检索、路由和整合，适用于有限工具或数据源的设置，可扩展性有限，对于多步推理或大型数据集表现较差。

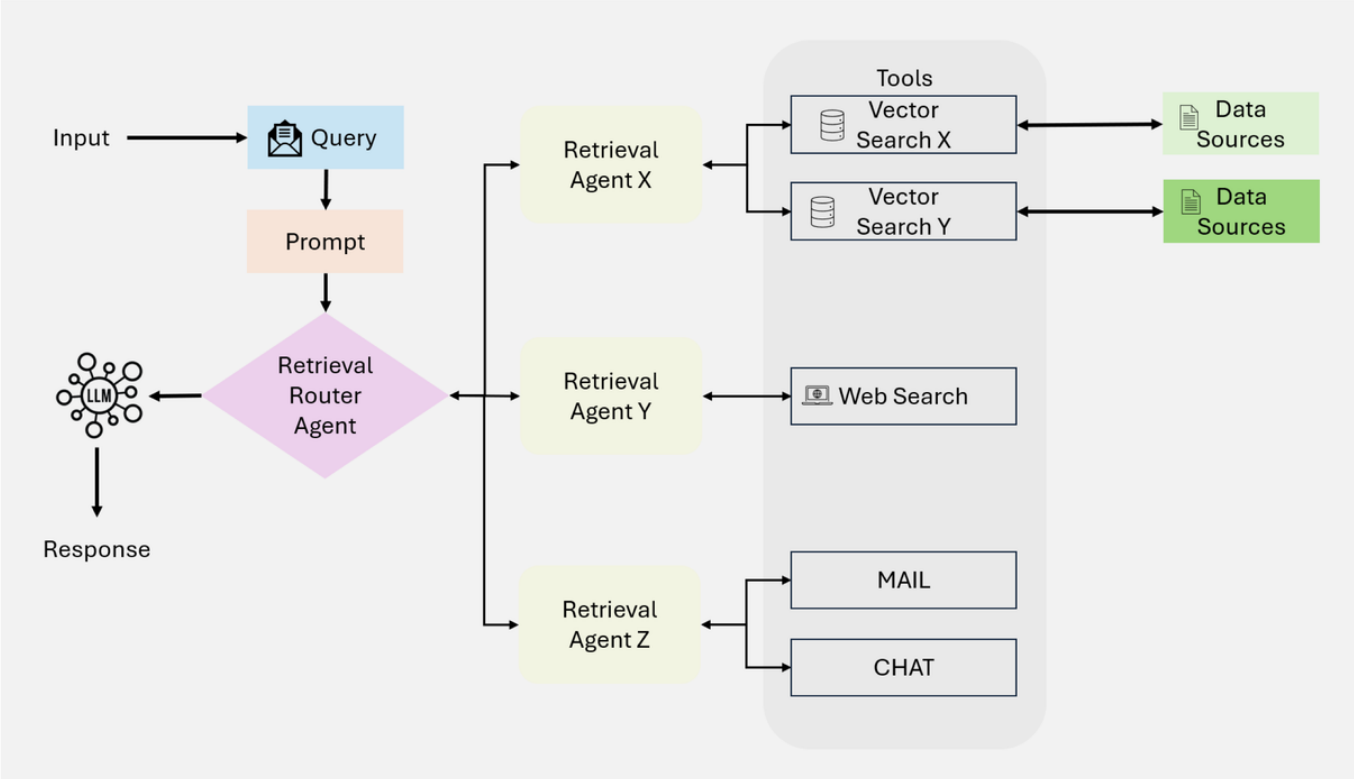
举例：《Search-o1: Agentic Search-Enhanced Large Reasoning Models》



Multi-Agent Agentic RAG

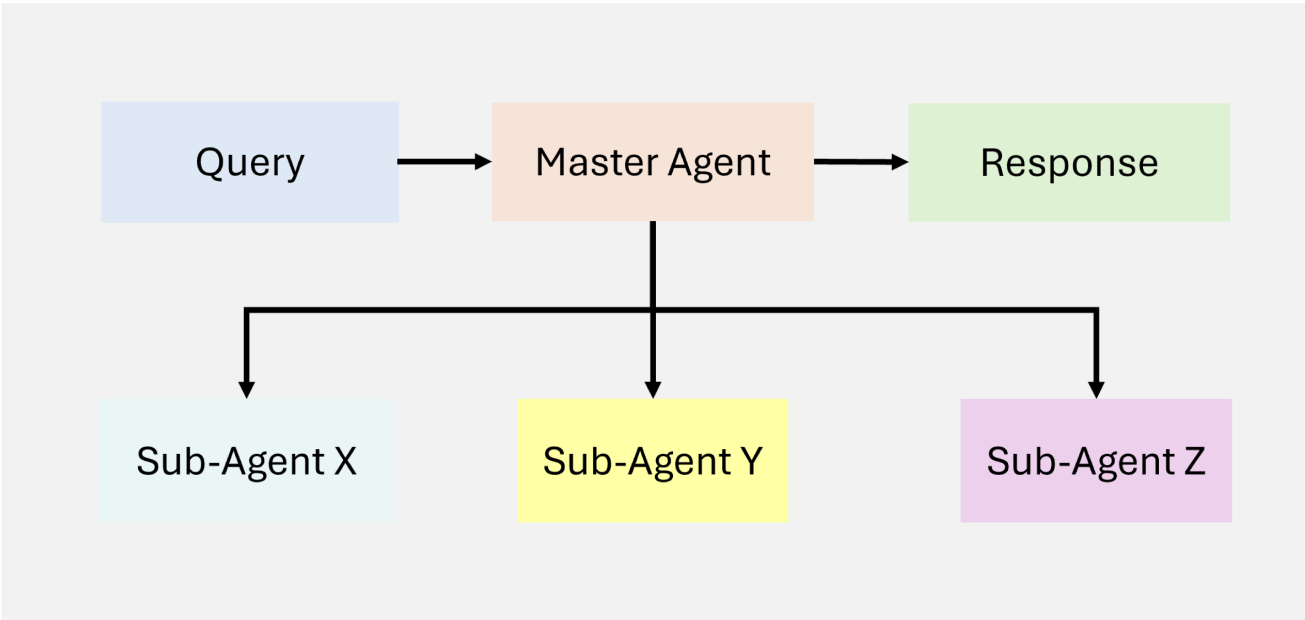
Multi-Agent Agentic RAG 通过利用多个专门的Agent来处理复杂的工作流程和多样化的查询类型。该系统不再依赖于单个Agent来管理所有任务（推理、检索和响应生成），而是将责任分配给多个Agent，每个Agent针对特定的角色或数据源进行了优化。对于分布式、多步骤任务的性能更好，增加模块化和可扩展性。

举例：《Agentic Retrieval-Augmented Generation for Time Series Analysis》



分层 Agentic RAG

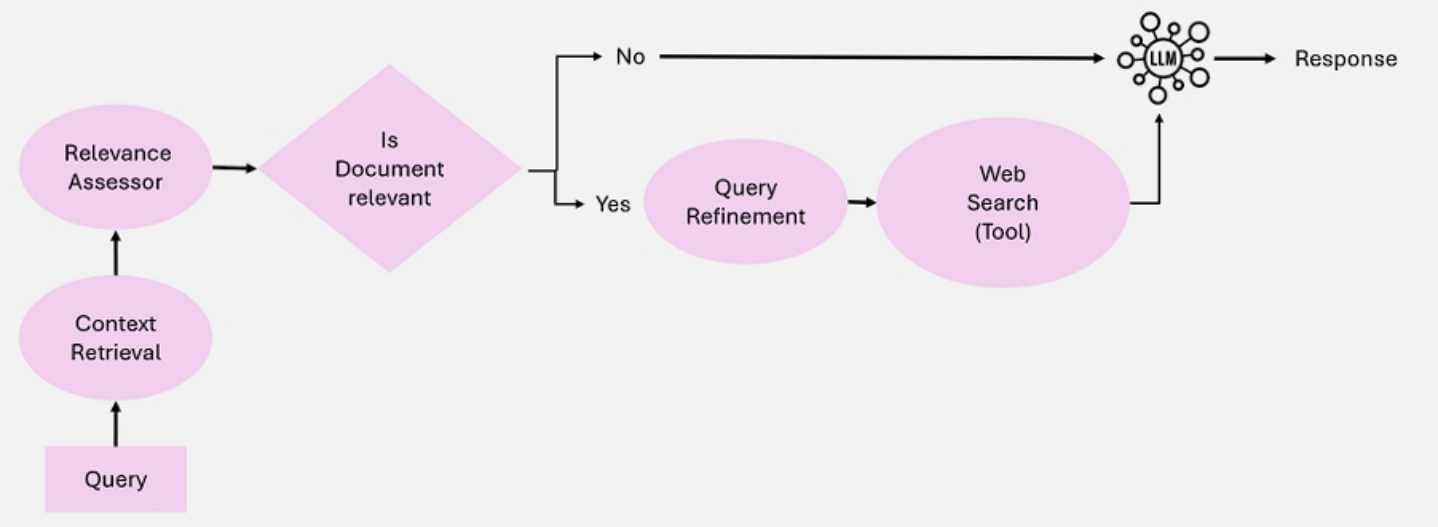
分层Agentic RAG系统采用结构化的多层次方法进行信息检索和处理，Agent按层次结构组织，高级Agent监督和指导低级Agent。这种结构实现了多级决策，确保查询由最合适的资源处理。



Agentic Corrective RAG

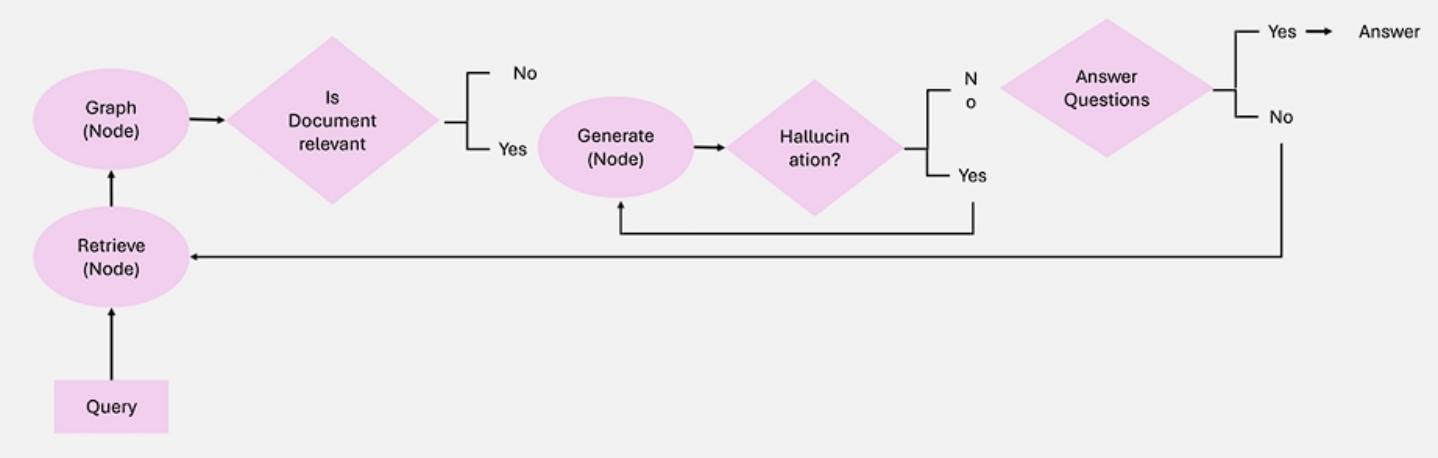
Corrective RAG 引入了自我纠正检索结果的机制，迭代地改进上下文文档和响应，最小化错误并最大化相关性。

举例：《Agentic AI-Driven Technical Troubleshooting for Enterprise Systems: A Novel Weighted Retrieval-Augmented Generation Paradigm》《Corrective RAG (CRAG)》



Adaptive Agentic RAG

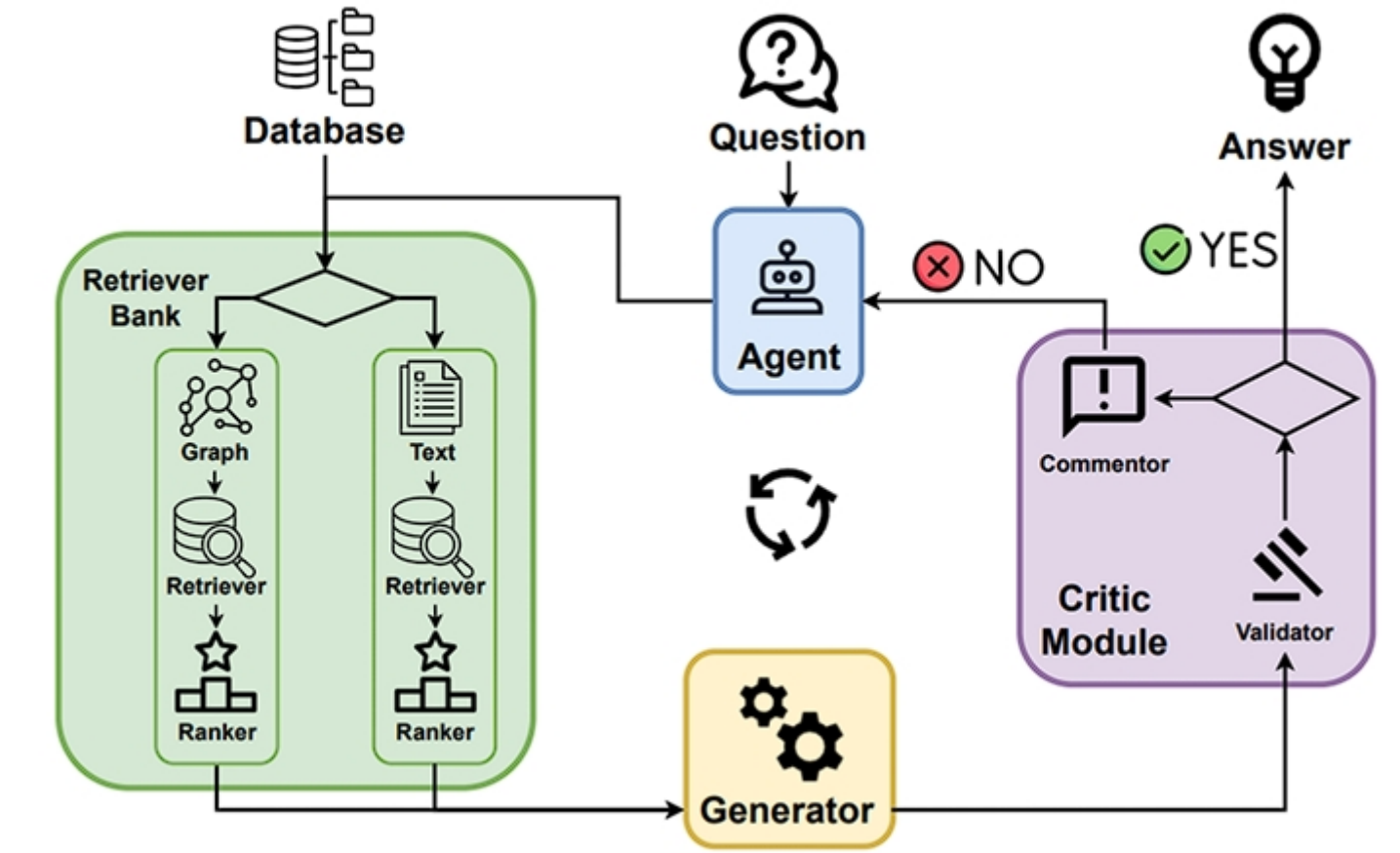
其思想在于根据任务需求动态调整检索策略和工作流程。工作流程为：Agent评估查询及其上下文->基于可用数据和用户需求实时调整检索策略->使用动态工作流程合成响应。



Graph-Based Agentic RAG

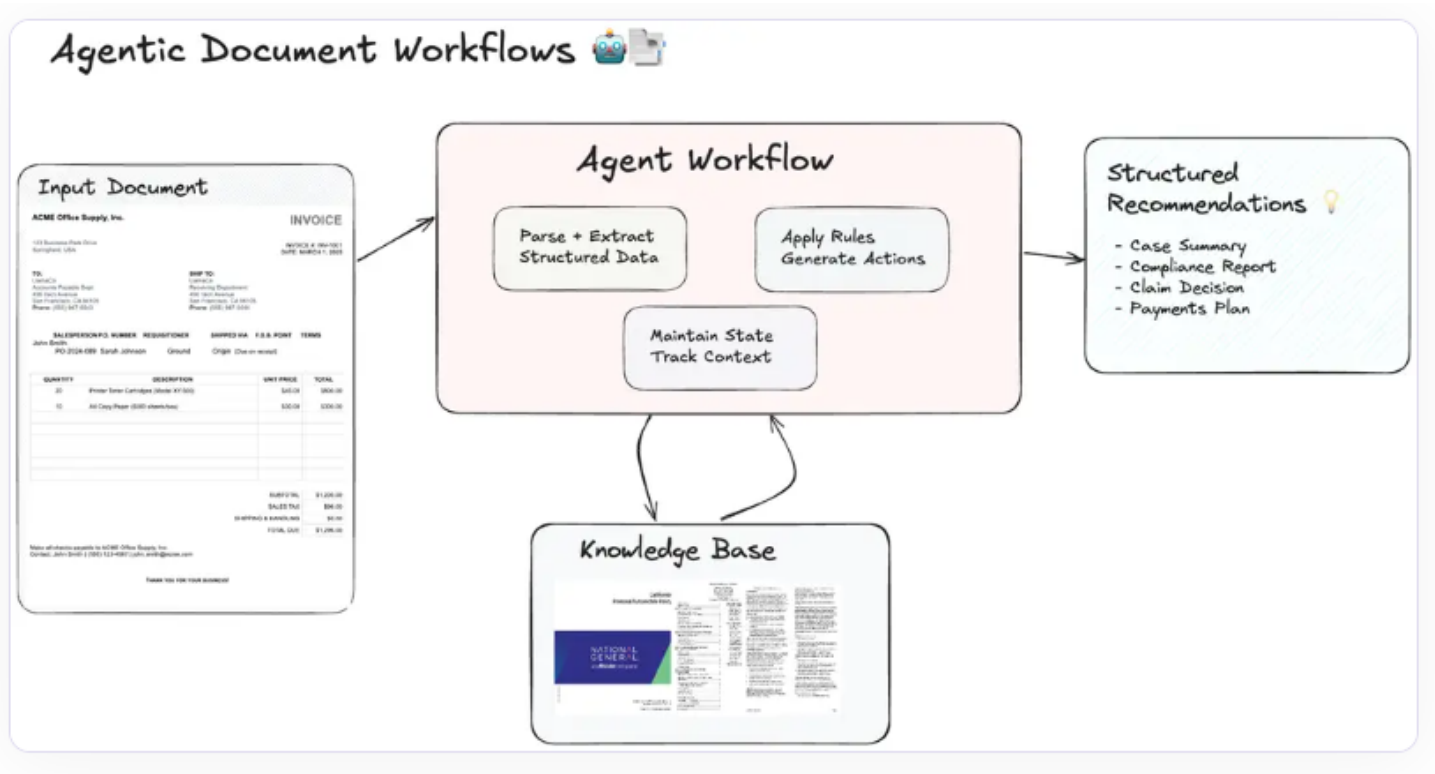
将图知识库与非结构化文档检索相结合。通过结合结构化和非结构化数据源，利用图知识库和反馈循环动态分配任务给专业Agent，提高了 RAG 系统的推理和检索准确性。

例子：《Agent-G: An Agentic Framework for Graph Retrieval Augmented Generation》



Agentic文档工作流

Agentic Document Workflows 通过实现端到端的知识工作自动化，扩展了传统的 RAG 范式。这些工作流程协调以文档为中心的复杂过程，集成了文档解析、检索、推理和结构化输出，并与智能Agent结合。



Agentic RAG 工具和数据集

这里只介绍我使用过的工具，更多工具请参考原论文

- LangChain 和 LangGraph
- LLamaIndex
- Hugging Face Transformers 和 Qdrant

RAG评估数据集：

Category	Task Type	Datasets and References
QA	Single-hop QA	Natural Questions (NQ) [65], TriviaQA [66], SQuAD [67], Web Questions (WebQ) [68], PopQA [69], MS MARCO [56]
	Multi-hop QA	HotpotQA [60], 2WikiMultiHopQA [59], MuSiQue [58]
	Long-form QA	ELI5 [70], NarrativeQA (NQA) [71], ASQA [72], QMSum [73]
	Domain-specific QA	Qasper [74], COVID-QA [75], CMB/MMCU Medical [76]
	Multi-choice QA	QuALITY [77], ARC (No reference available), CommonsenseQA [78]
Graph-based QA	Graph QA	GraphQA [79]
	Event Argument Extraction	WikiEvent [80], RAMS [81]
Dialog	Open-domain Dialog	Wizard of Wikipedia (WoW) [82]
	Personalized Dialog	KBP [83], DuleMon [84]
	Task-oriented Dialog	CamRest [85]
Recommendation	Personalized Content	Amazon Datasets (Toys, Sports, Beauty) [86]
Reasoning	Commonsense Reasoning	HellaSwag [87], CommonsenseQA [78]
	CoT Reasoning	CoT Reasoning [88]
	Complex Reasoning	CSQA [89]
Others	Language Understanding	MMLU (No reference available), WikiText-103 [65]
	Fact Checking/Verification	FEVER [90], PubHealth [91]
	Strategy QA	StrategyQA [92]
Summarization	Text Summarization	WikiASP [93], XSum [94]
	Long-form Summarization	NarrativeQA (NQA) [71], QMSum [73]
Text Generation	Biography	Biography Dataset (No reference available)
Text Classification	Sentiment Analysis	SST-2 [95]
	General Classification	VioLens[96], TREC [57]
Code Search	Programming Search	CodeSearchNet [97]
Robustness	Retrieval Robustness	NoMIRACL [98]
	Language Modeling Robustness	WikiText-103 [99]
Math	Math Reasoning	GSM8K [100]
Machine Translation	Translation Tasks	JRC-Acquis [101]

Agentic RAG实战项目详见《https://github.com/asinghcsu/AgenticRAG-Survey?tab=readme-ov-file#implementation-of-rag-agentic-taxonomy-techniques-and-tools》