公开课：《Agentic RAG 原理与实战》

第一部分：原理

1. RAG 背景

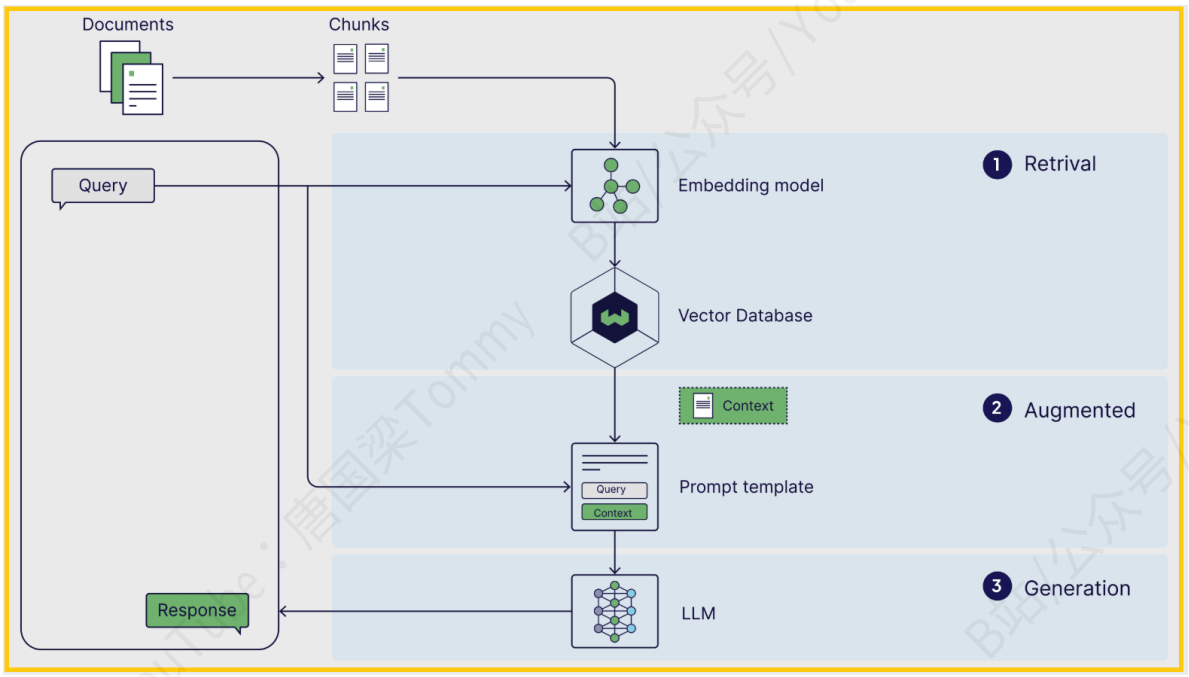
1.1 LLM 面临的挑战

局限	产生原因	典型表现
时效性不足	预训练语料静态，更新不及时	无法回答最近 72 小时新闻、最新法规
行业深度不够	通用语料缺乏垂直领域知识	医疗、法律等专业概念回答不准确
事实幻觉	模型生成依赖语言概率而非事实检索	引用不存在的论文、伪造数据
推理链缺失	黑盒生成，缺乏可解释性与因果链	难以支撑推理过程，合规审核困难

> **核心痛点**：预训练 LLM 本质是「静态知识库 + 语言模式」，一旦问题超出语料覆盖范围，模型极易**凭空编造内容**。

1.2 RAG的作用

RAG（Retrieval-Augmented Generation）通过引入实时检索机制，有效弥补了 LLM 静态知识的局限，增强模型对上下文的理解与时效性响应能力。



RAG 的核心机制包括：

- **Retrieval (R) 检索：** 从外部源、数据库或知识库中搜索相关数据。目标是收集特定、准确和相关的信息，以支持或增强AI对特定主题或查询的理解。
- **Augmentation (A) 增强：** 在此阶段，检索到的数据被添加到提示（prompt）上下文中。这意味着信息被集成或结合到提供给AI的输入中，有效地丰富其知识库，以实现更好的推理和上下文感知响应。
- **Generation (G) 生成：** 最后，LLM 使用增强的上下文生成输出。

借助这一流程，RAG 架构能够显著提升生成内容的**准确性**、**可控性与现实关联度**，为复杂任务提供更可靠的知识支持。

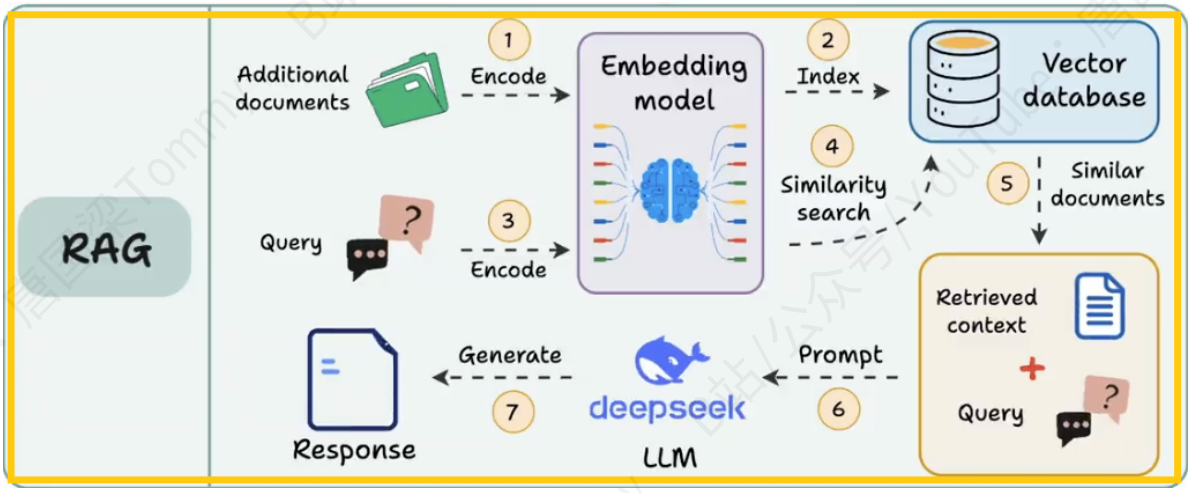
1.3 LLM vs RAG 比较

以下是 RAG和 LLM在多个方面的差异：

特征	LLM（大模型）	LLM With RAG
准确性	容易生成未经证实或幻觉内容，不依赖可靠来源	响应基于外部来源的可验证引用
时效性	依赖静态的预训练数据，可能过时或与当前事件无关	通过整合来自外部源的实时、最新信息增强静态预训练数据
上下文清晰度	经常难以解释歧义查询，导致模糊或不完整的答案	检索特定上下文的信息，提高响应的清晰度
定制化	无法访问或利用用户特定数据集或私有源，导致通用响应	集成公共和私有数据集，实现高度定制化和相关的输出
搜索范围	限于预训练知识库；无法扩展到新信息或外部信息	能够在多个数据库或在线源上进行广泛的、按需搜索
可靠性	由于依赖静态和预生成知识，错误的可能性高	通过实时交叉引用多个受信任的来源确保可靠性
用例	适用于通用任务，但对于动态或数据密集型应用效果较差	适用于需要实时更新、研究或自定义数据集成的任务
透明度	提供的信息没有明确的参考或引用，难以验证	提供引用或链接到来源，确保透明度和可信度

2. RAG 工作原理与挑战

RAG系统架构分为两个主要部分：1️⃣ **数据索引** 2️⃣ **搜索与生成**



2.1 数据索引

这部分侧重于在检索过程中准备和管理知识库。

- **步骤1：加载**：系统获取不同类型的数据（例如，文本文件、PDF、URL和JSON文件）。数据

可以来自不同的来源，确保知识库的全面性。

- **步骤2：分割**：数据被分成更小、有意义的块（chunks）。这一步确保检索高效工作，允许系统获取文档中精确相关的部分，而不是检索整个文件。
- **步骤3：嵌入：**数据的每个块使用嵌入模型转换为向量表示。这些嵌入捕获文本的语义含义，使系统能够执行基于相似度的搜索。
- **步骤4：存储**：嵌入和相应的数据存储在向量数据库中。向量数据库针对快速准确的相似度搜索进行了优化，这对检索至关重要。

2.2 搜索与生成

这描述了结合检索和生成来产生答案的整个过程。

- **步骤1：问题输入**：用户提供查询或问题。系统首先分析此问题以了解上下文和意图。
- **步骤2：检索**：系统查询已索引的知识库（检索系统）以收集最相关的文档或信息片段。这些文档作为生成答案的支持证据或上下文。
- **步骤3：提示创建：**检索到的文档被构建成一个用于LLM的提示。提示包括原始问题和检索到的信息，指导LLM生成一个上下文感知的响应。
- **步骤4：生成**：LLM处理提示，利用其生成能力创建连贯精确的响应。响应结合了检索到的文档中的见解和LLM的预训练知识。
- **步骤5：答案输出**：最终答案呈现给用户，融合了检索到的知识和LLM的生成能力。

2.3 RAG面临的挑战

尽管RAG在克服LLM局限性方面有效，尤其适用于基础任务（如问答场景），但它在复杂用例中仍然面临挑战。

挑战类别	具体挑战	解决方案
1️⃣ 检索与生成融合	- 检索结果不准确或评分不高导致生成错误 - 多文档融合难，影响上下文利用 - LLM提取答案受上下文噪声影响	- 优化检索算法，提升相关性 - 设计有效的融合策略 - 采用Self-RAG等自反机制提升生成准确性
2️⃣ 文档切分粒度	- 切块过粗导致语义丢失 - 切块过细带来噪声和上下文不完整	- 基于语义和语言模型辅助分块 - 通过测试调整最佳分块粒度
3️⃣ 系统扩展性与成本	- API调用成本高 - 向量数据库延迟和吞吐瓶颈 - 维护成本高	- 精简索引，删除冗余 - 分层索引检索减少计算 - 结合本地模型与云端API分配资源
4️⃣ 隐私与数据安全	- 私有数据传输风险 - 向量数据库存储敏感信息安全问题	- 本地处理或加密存储敏感数据 - 设计访问控制和隐私保护机制
5️⃣ 检索-生成语义不对称	- 查询与文档语义不匹配 - LLM对矛盾或无关信息鲁棒性差	- 使用假设文档嵌入（HyDE）提升语义对称性 - 利用LLM优化查询 - 引入自然语言推理过滤上下文
7️⃣ 垂直领域适应性	- 通用模型在专业领域表现差 - 文档拆分和问答分句效果有限	- 领域微调embedding和LLM - 改进拆分和问答分句技术
8️⃣ 系统有效性验证	- 缺乏统一评估标准 - 需要大量测试和调优	- 建立代表性测试集 - 持续监控和调整模型与检索配置

3. Agentic RAG 概述

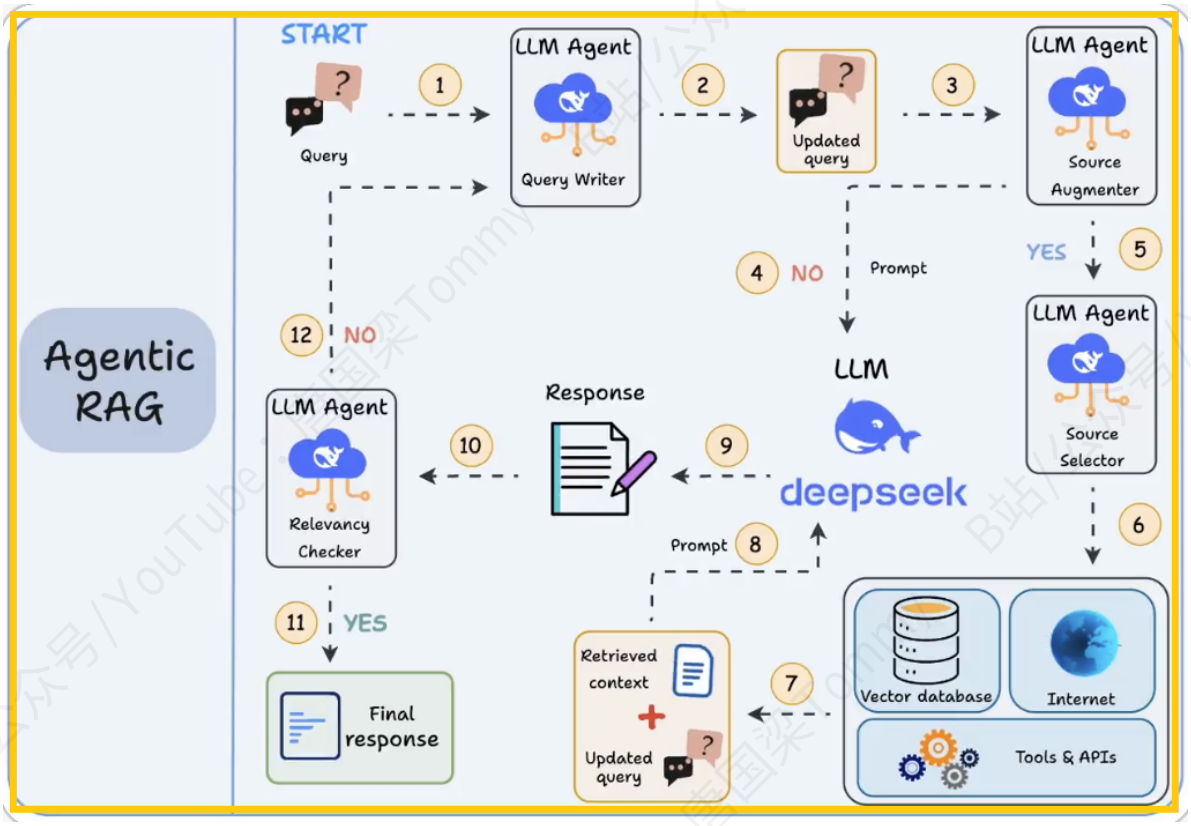
Agentic RAG（智能体驱动的检索增强生成）是对传统 RAG 架构的深度演进。通过引入自主 AI 智能体，Agentic RAG 实现了对检索策略的动态管理、上下文理解的迭代优化，以及任务流程的自适应调整，从而突破了传统 RAG 在处理复杂任务和多步推理方面的局限性。

- **核心理念**

  Agentic RAG 的核心在于将智能体引入 RAG 流程，使系统具备以下能力：

  - **动态检索策略**：智能体根据任务需求，灵活选择检索工具（如向量数据库、网络搜索、API 接口等），并优化查询方式，以获取最相关的信息。
  - **上下文理解与迭代优化**：通过反思机制，智能体能够评估生成结果的质量，必要时重新检索信息或调整生成策略，提升回答的准确性和相关性。

- **自适应工作流程**：智能体根据任务的复杂度和类型，动态调整工作流程，实现从顺序执行到多智能体协作的灵活转变。



- **关键特性**
  - **Agentic（智能体驱动）**：系统具备自主决策和行动能力，能够根据环境变化和任务需求，主动规划和执行操作。
  - **RAG（检索增强生成）**：结合外部知识库的信息检索能力与大语言模型的生成能力，提供更准确、上下文相关的响应。

通过融合智能体的自主性与 RAG 的信息整合能力，Agentic RAG 在处理复杂查询、多步推理和动态任务方面展现出更高的灵活性和效率。

---

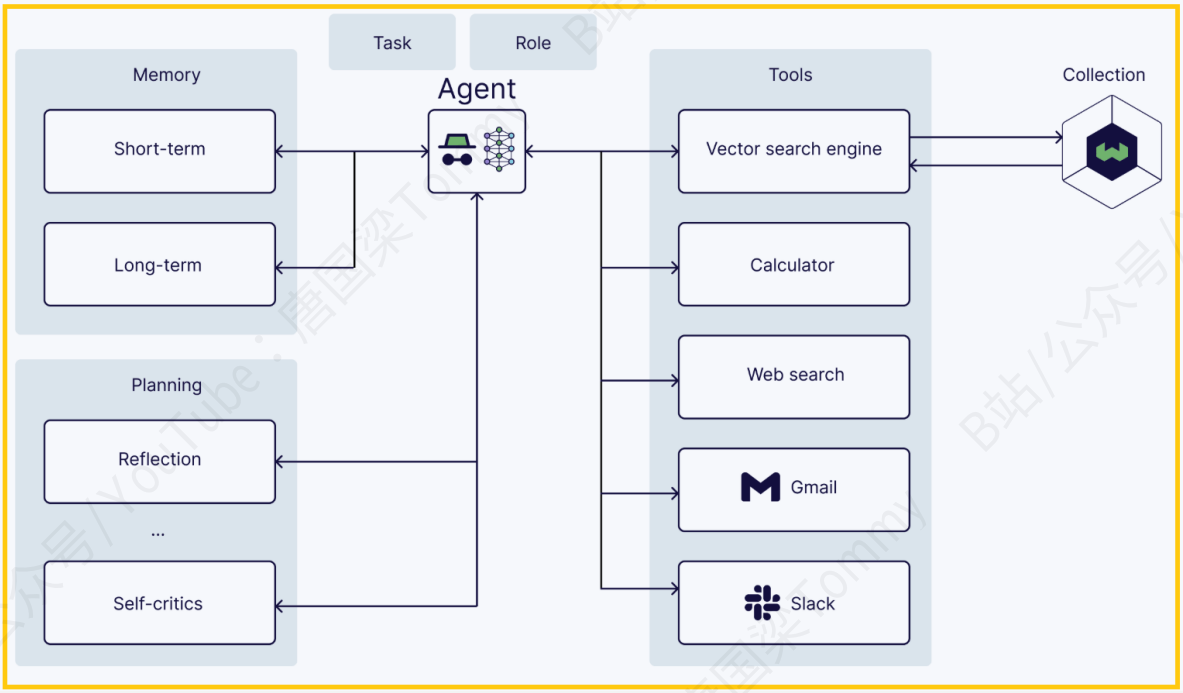
4. Agent 核心组件

Agent智能是 Agentic RAG 系统的基础，使其能够超越传统 RAG 的静态和被动性质。通过集成能够动态决策、迭代推理和协作工作流程的自主智能体(Self Agent)，Agentic RAG 系统表现出增强的适应性和精度。

一个 AI Agent 本质上包含几个组件:

- **LLM（具有定义的角色和任务）**: 作为智能体主要的推理引擎和对话接口，负责解释查询、生成响应和维护连贯性。
- **记忆（短期和长期）**: 捕捉跨交互的上下文和相关数据，短期记忆跟踪即时对话状态，长期记忆存储累积的知识和智能体经验。

- **规划（反思和自我批评）**：通过反思、查询路由或自我批评来指导智能体的迭代推理过程，确保复杂任务被有效分解。
- **工具（向量搜索、网络搜索、API 等）**：扩展智能体的能力，使其能够访问外部资源、实时数据或进行专业计算。



5. Agent 设计模式

```
# 参考资源
1. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG
https://arxiv.org/pdf/2501.09136v3
```

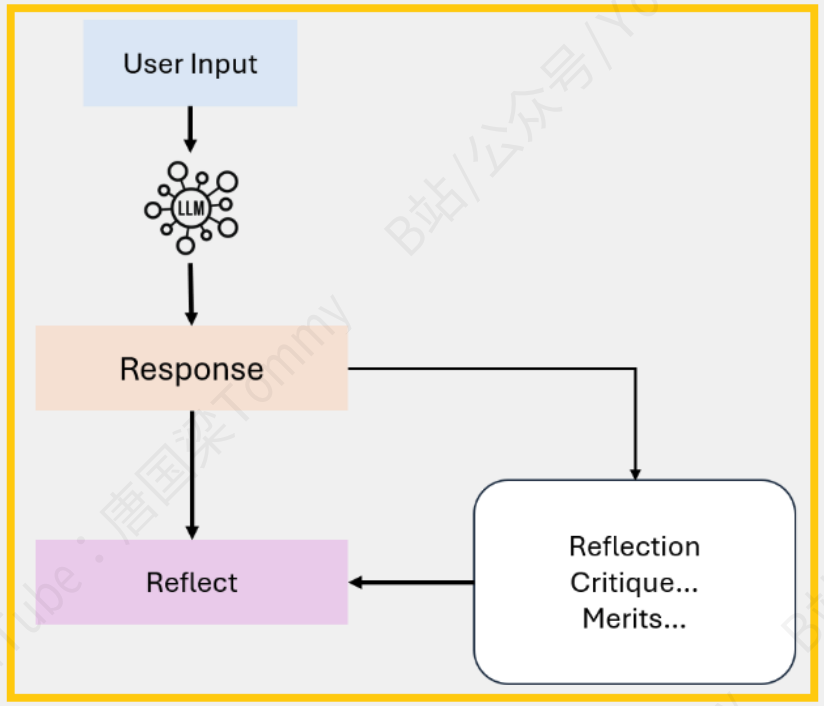
智能体设计模式为构建具备自主性、适应性和协作能力的智能系统提供了结构化的方法。这些模式使智能体能够在复杂、多变的环境中高效运行，提升系统的准确性和可扩展性。本文重点介绍四种关键模式：

1️⃣ 反思 (Reflection)

反思模式使智能体能够自我评估并持续优化其输出。通过引入自我反馈机制，智能体可以识别并修正错误、不一致或低效的部分，从而提高整体性能。在实践中，反思通常包括以下步骤：

- **自我评估**：智能体对其输出进行分析，判断其准确性、风格和效率。
- **反馈整合**：将评估结果纳入后续迭代，优化输出质量。
- **外部辅助**：借助单元测试、网络搜索等工具增强评估过程。

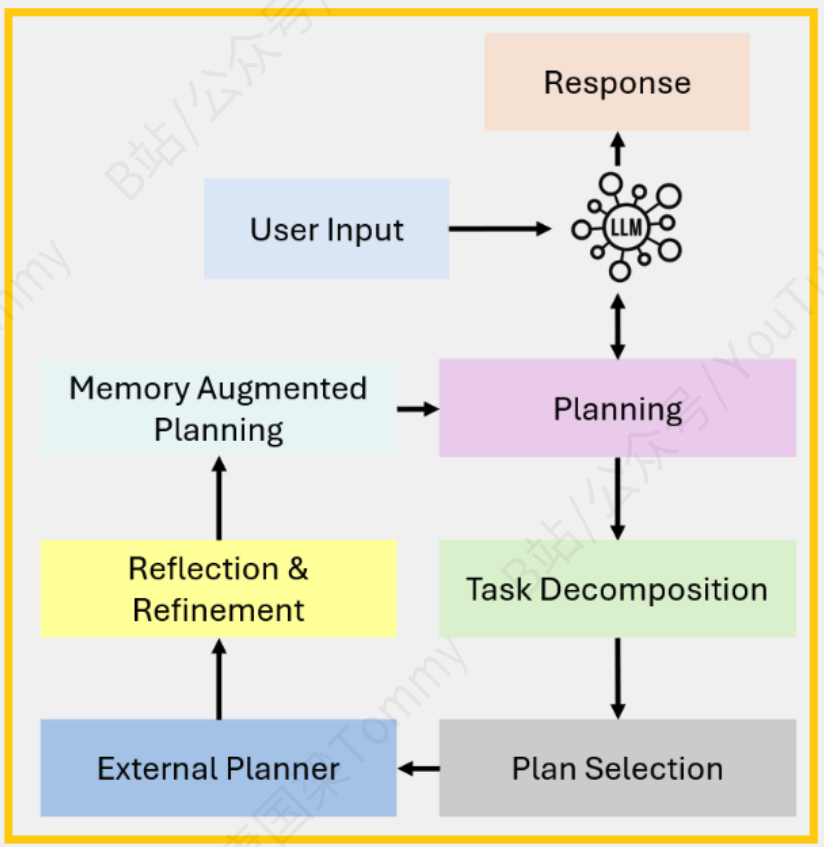
在多智能体系统中，反思可以通过角色分工实现，例如一个智能体生成输出，另一个进行审查和反馈。研究表明，反思机制能够显著提升智能体的表现，尤其在复杂任务中表现尤为突出。



2️⃣ 规划 (Planning)

规划模式使智能体能够将复杂任务分解为更小、可管理的子任务，并制定执行策略。这对于需要多步推理或在动态环境中操作的任务尤为重要。关键特性包括：

- **任务分解**：将整体目标拆分为具体步骤，明确每一步的目标和方法。
- **动态调整**：根据环境变化实时更新计划，确保任务的顺利完成。
- **协同执行**：在多智能体系统中，各智能体根据规划协同工作，提高效率。

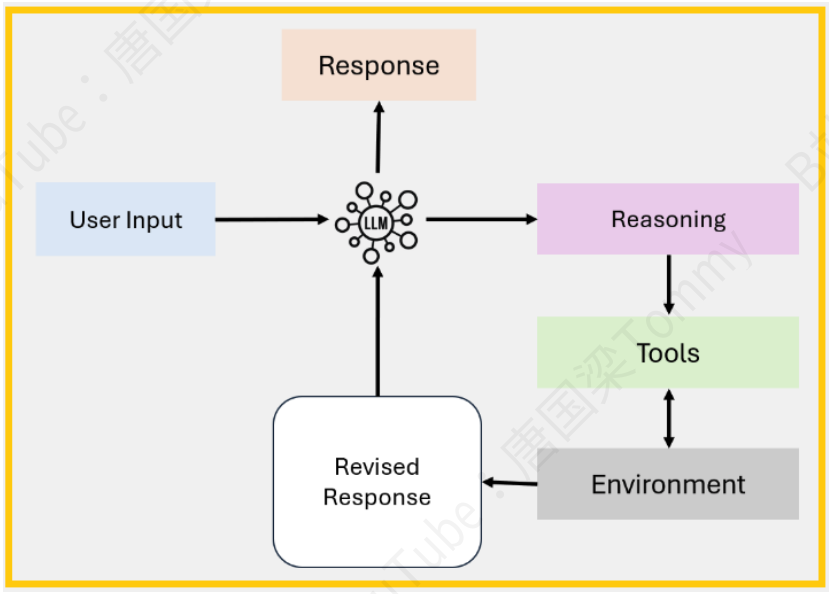


3️⃣ 工具使用 (Tool Use)

工具使用模式扩展了智能体的能力，使其能够利用外部资源（如API、数据库、计算引擎）执行超出其原生功能的任务。这包括：

- **信息检索**：通过网络搜索获取最新数据。
- **数据处理**：使用编程语言（如Python）进行复杂计算或数据分析。
- **系统集成**：与其他软件系统交互，实现任务自动化。

例如，AutoGen框架允许智能体调用多种工具，增强其在数据分析、代码生成等方面的能力。

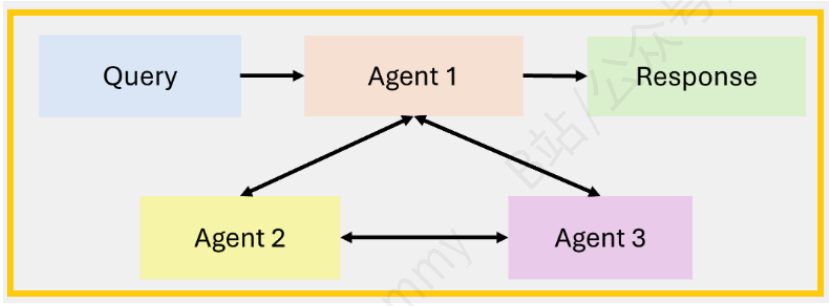


4️⃣ 多智能体协作 (Multi-Agent Collaboration)

多智能体协作模式通过多个智能体的协同工作，实现任务的专业化处理和并行执行。其核心优势包括：

- **角色分工**：不同智能体承担特定子任务，提高整体效率。
- **信息共享**：智能体之间交换中间结果，确保任务的一致性和连贯性。
- **弹性扩展**：系统可根据需求动态调整智能体数量和功能，适应不同规模的任务。

研究显示，多智能体协作在企业应用中表现出色，能够显著提升任务完成率和系统响应速度。

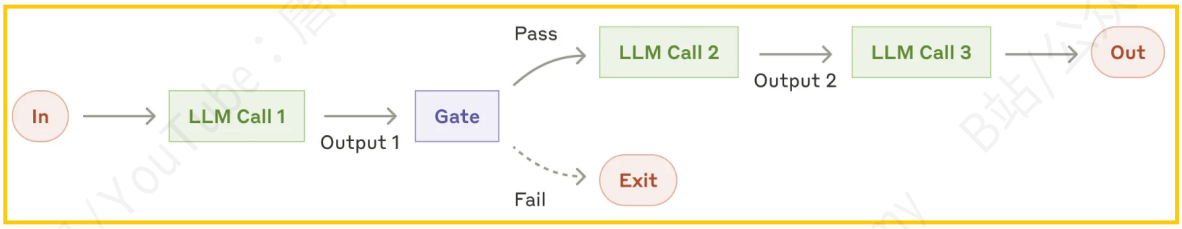


6. Agent 工作流模式

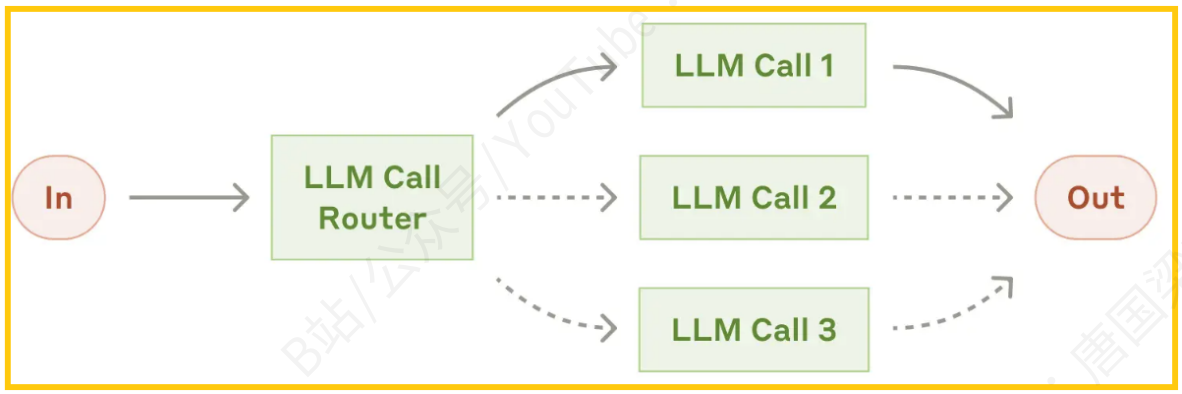
```
# 参考Anthropic定义
https://www.anthropic.com/engineering/building-effective-agents
```

智能体工作流模式构建 LLM 应用程序，以优化性能、准确性和效率。根据任务的复杂性和处理需求，有不同的适用方法：

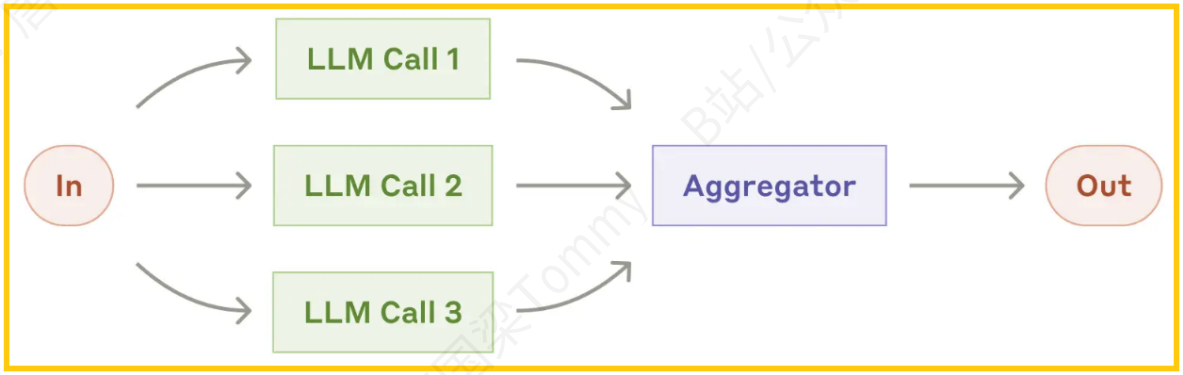
1️⃣ **提示链 (Prompt Chaining):** 将复杂任务分解为多个步骤，每个步骤都以前一步骤为基础。这是一种结构化的方法，通过简化每个子任务来提高准确性，但顺序处理可能增加延迟。



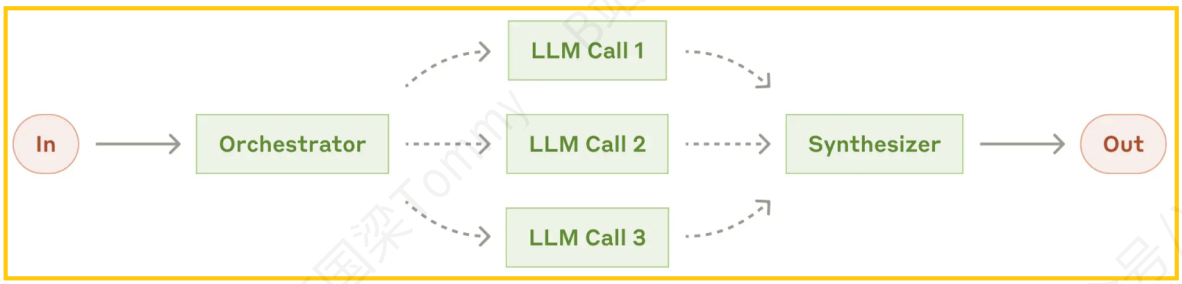
2️⃣ **路由 (Routing):** 对输入进行分类，并将其定向到适当的专业提示或流程。这确保了不同类型的查询或任务得到单独处理，提高了效率和响应质量。适用于不同输入需要不同处理策略的场景，例如客户服务查询分类。



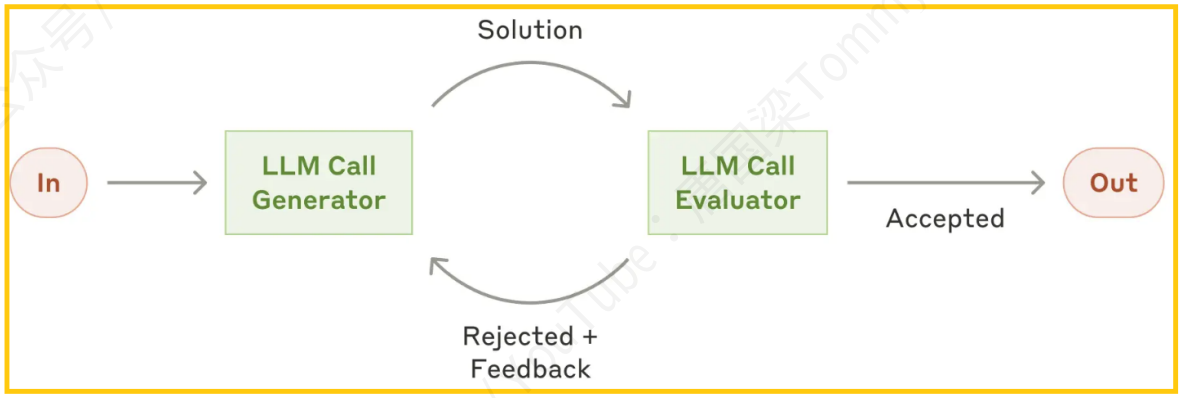
3️⃣ **并行化 (Parallelization):** 将任务分解为独立进程并行运行，从而减少延迟并提高吞吐量。可分为分段（独立子任务）和投票（多个输出以提高准确性）。适用于任务可独立执行以提高速度，或多个输出可提高置信度的场景。



4️⃣ **协调者-工作者 (Orchestrator-Workers):** 协调者模型动态地将任务分解为子任务，分配给专业工作者模型，并汇总结果。与并行化不同，它能适应不同的输入复杂性。适用于需要动态分解和实时适应的任务，其中子任务不是预定义的。



5️⃣ **评估者-优化者 (Evaluator-Optimizer):** 通过生成初始输出并根据评估模型的反馈进行优化，迭代地改进内容。当迭代优化可以显著提高响应质量，特别是在存在明确评估标准的情况下，这种方法是有效的。



---

7. Agentic RAG 架构分类

```
# 参考资源
1. What is Agentic RAG
https://weaviate.io/blog/what-is-agentic-rag
```

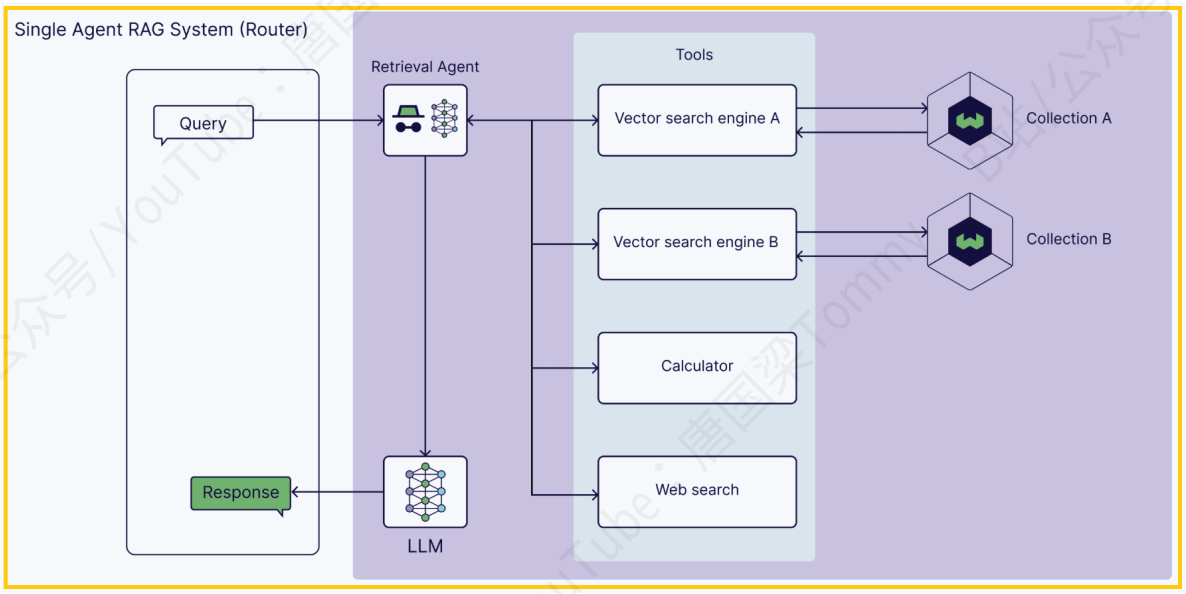
Agentic RAG 系统可以根据其复杂性和设计原则分为不同的架构框架。

7.1 单个 Agentic RAG

- **定义：**

  Single-Agent RAG 是 Agentic RAG 的最基础形式，系统中只引入一个智能体（Agent）来执行任务，主要用于处理多数据源检索任务。

- **架构特点：**
  - **核心角色**：一个智能体负责整个任务流程，包括检索、工具选择、上下文判断与答案生成。
  - **路由能力**：该智能体可以作为“路由器”（Router），根据查询内容动态决定从哪个知识源或工具中提取信息。
  - **接入工具**：支持接入多个外部工具或数据源，如：
    - 向量数据库（如 Weaviate）
    - Web 搜索引擎
    - 内部API（如Slack、邮箱等）



- **应用场景示例：**
  - FAQ系统：可根据问题类型判断是从数据库、网页还是FAQ文档中获取答案。
  - 客服Agent：接入多个业务系统（工单、邮件、用户行为日志）提供支持。

- **优点：**
  - 实现简单，结构清晰。
  - 适用于数据源不复杂或智能体任务边界明确的场景。
  - 较传统RAG更灵活智能，可动态选择信息源。

- **局限：**
  - 所有任务都集中在一个Agent上，易造成负载过重。
  - 智能体功能杂糅（路由、检索、判断、生成），不利于模块化优化。

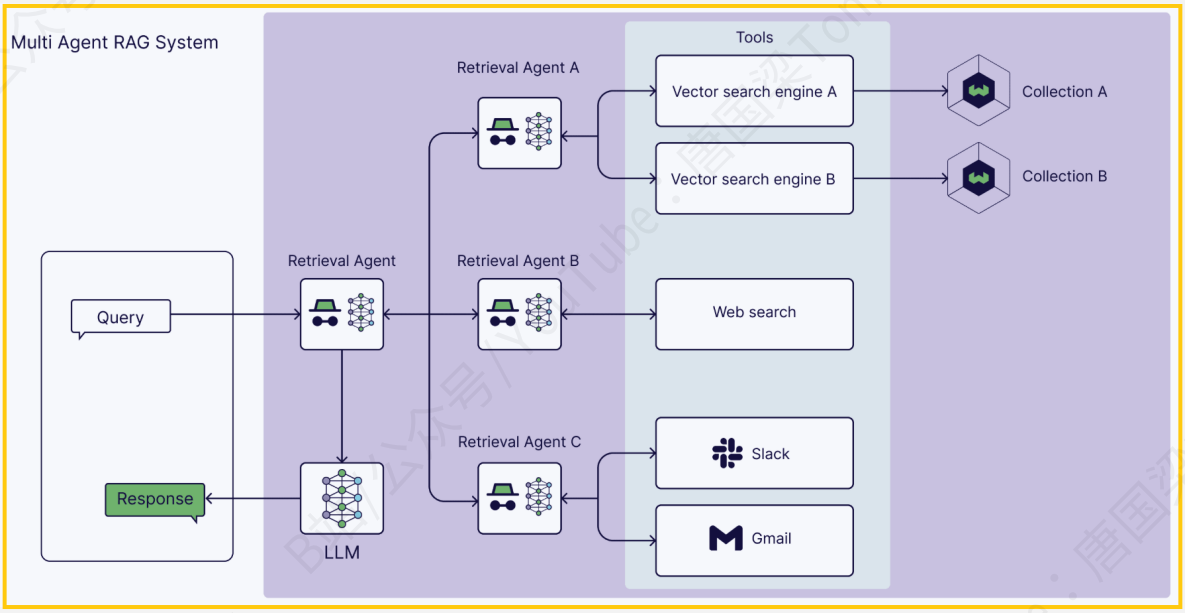
7.2 多个 Agentic RAG

- **定义：**

Multi-Agent RAG 是更复杂的 Agentic RAG 实现，系统中引入多个功能划分明确的智能体，通过**主控智能体（master agent）**协调各个**专职智能体（specialized agents）**协同完成任务。

- **架构特点：**
  - **主控智能体：**
    - 负责任务拆解、子任务分配、汇总各个子任务结果。
  - **专职智能体：**

    每个Agent负责特定检索任务，比如：
    - Agent A：检索内部文档数据
    - Agent B：基于Web执行实时搜索
    - Agent C：调用公司API查邮箱



- **应用场景示例：**
  - 企业级问答系统：结合私有知识库、ERP系统数据、外部API、公共网页搜索。
  - 多模态信息检索：文字Agent检索文本，图像Agent调用图像识别模块，视频Agent查找相关片段。

- **优点：**
  - 模块化强：任务明确、职责清晰，便于扩展和优化。
  - 支持更复杂、多阶段的任务链。
  - 各Agent可独立优化，便于并行处理，提升效率。

- **局限：**

- 架构更复杂，需要调度与通信机制。
- Agent间协调不当可能造成冲突、冗余或信息损失。
- 成本更高（模型调用频率上升），需优化任务拆解逻辑。

7.3 层级式 Agentic RAG

核心理念：

采用多层级 (Hierarchical) 智能体架构，由上层智能体（如主控 Agent）协调下层专职智能体（如检索 Agent、生成 Agent）完成复杂任务。

架构特点：

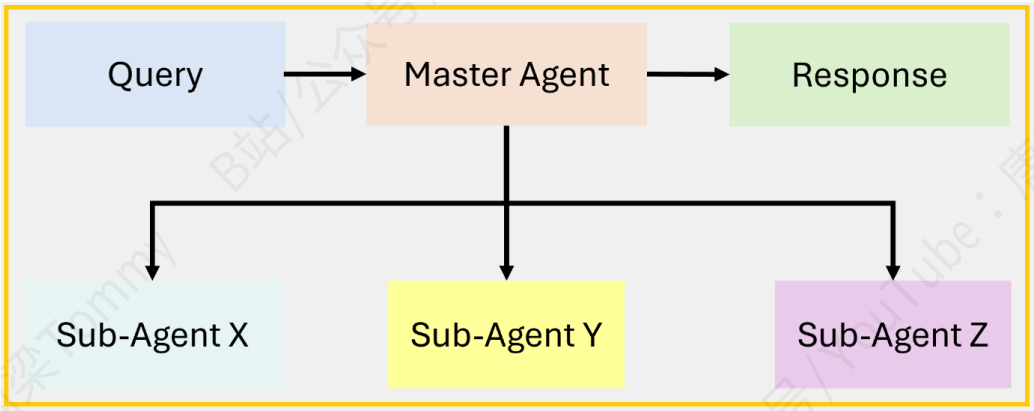
- **主控智能体：** 负责任务分解、子任务分配和结果整合。
- **专职智能体：** 各自承担特定功能，如检索、生成、验证等。
- **层级协作：** 上层智能体可监督和指导下层智能体的操作。

适用场景：

- 需要多步骤推理和复杂任务管理的应用，如多文档问答、跨领域信息整合等。

优势：

- 提高系统的模块化和可扩展性。
- 增强任务处理的灵活性和准确性。



7.4 纠错型 Agentic RAG

```
# 参考来源
1. Corrective Retrieval Augmented Generation
https://arxiv.org/pdf/2401.15884
```

核心理念：

引入反馈机制，智能体在生成初步回答后进行自我评估，识别并纠正潜在错误，提升回答质量。

架构特点：

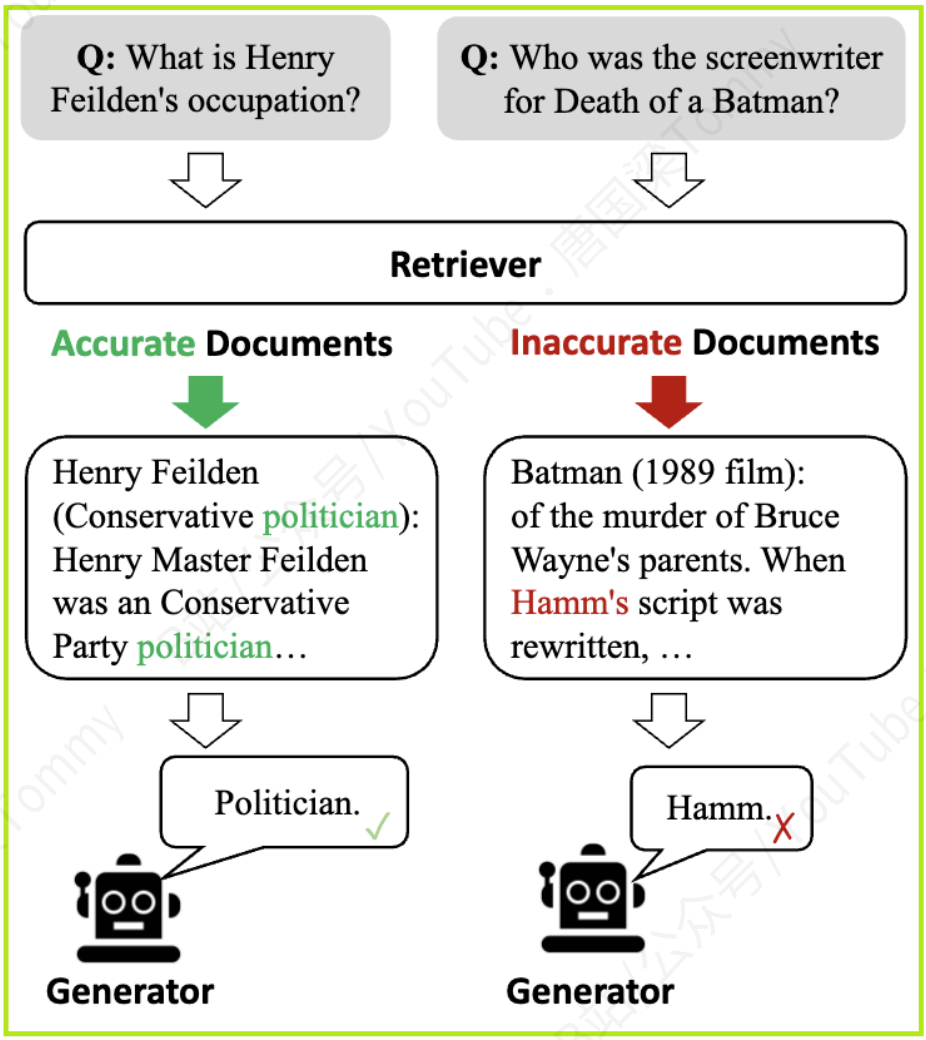
- **初步生成：** 智能体根据检索结果生成初步回答。
- **自我评估：** 引入“批评者（critic）”模块，评估回答的准确性和一致性。
- **迭代优化：** 根据评估结果，智能体对回答进行修正，直至满足质量标准。

适用场景：

- 对回答准确性要求高的应用，如医疗咨询、法律问答等。

优势：

- 显著降低错误率和幻觉现象。
- 提升用户对系统回答的信任度。



7.5 自适应 Agentic RAG

```
# 参考来源
1. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models
through Question Complexity
https://arxiv.org/pdf/2403.14403
2. https://www.analyticsvidhya.com/blog/2025/01/agentic-rag-system-
architectures/
```

核心理念： 智能体根据任务需求和中间结果，动态调整检索策略和工具选择，实现更灵活的任务处理。

架构特点：

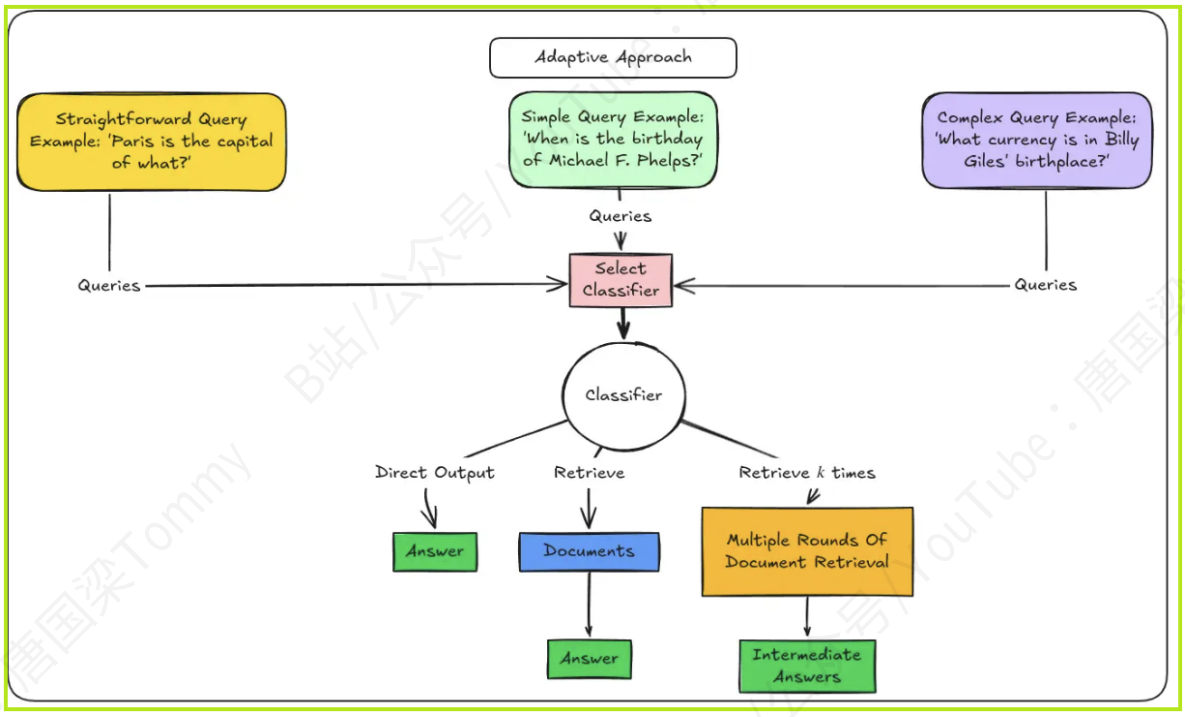
- **动态决策：** 智能体可根据当前上下文，决定是否需要进一步检索或更换工具。
- **多工具集成：** 支持接入多种工具和数据源，如向量数据库、Web 搜索、API 等。
- **实时反馈：** 根据中间结果，实时调整任务执行路径。

适用场景：

- 任务需求多变或信息来源多样的应用，如实时新闻摘要、跨领域研究等。

优势：

- 增强系统的灵活性和适应性。
- 提高任务处理的效率和准确性。



7.6 图结构 Agentic RAG

```
# 参考来源
https://neo4j.com/blog/developer/graphrag-and-agentic-architecture-with-neoconverse/
```

核心理念： 利用图数据库表示知识结构，智能体通过图遍历和多跳推理，实现更深层次的信息检索和理解。

架构特点：

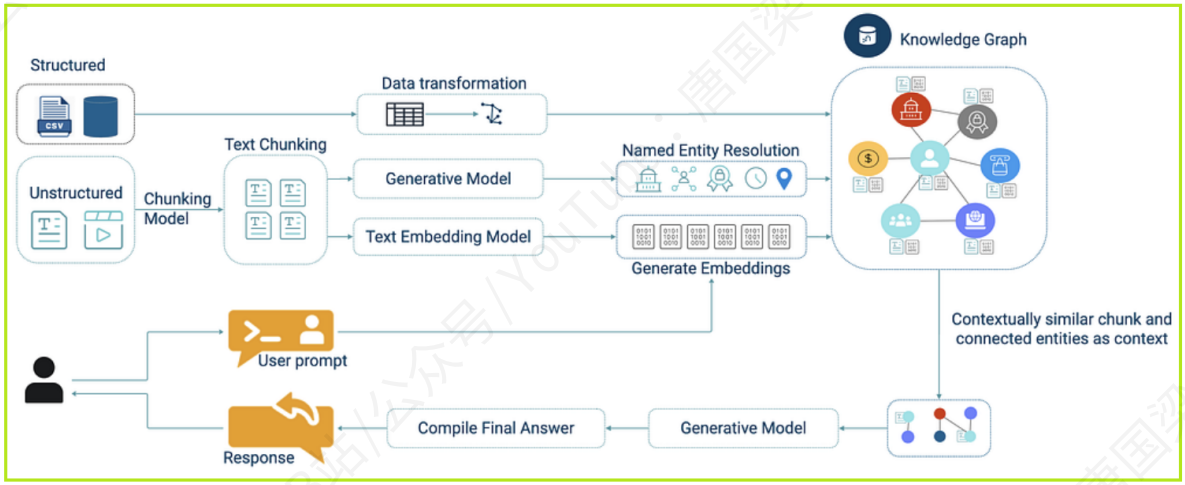
- **知识图谱：** 将实体及其关系表示为图结构，支持复杂的关系查询。
- **多跳推理：** 智能体可沿图中的路径进行多步推理，获取间接相关的信息。
- **混合检索：** 结合向量检索和图检索，提升信息检索的全面性。

适用场景：

- 需要理解复杂关系和上下文的应用，如学术研究、法律分析等。

优势：

- 提升信息检索的深度和广度。
- 增强系统的推理能力和上下文理解能力。



7.7 文档流程式 Agentic RAG

```
# 参考来源
https://www.llamaindex.ai/blog/introducing-agentic-document-workflows
```

核心理念： 通过智能体自动化处理文档相关任务，实现从文档解析、信息提取到结构化输出的全流程自动化。

架构特点：

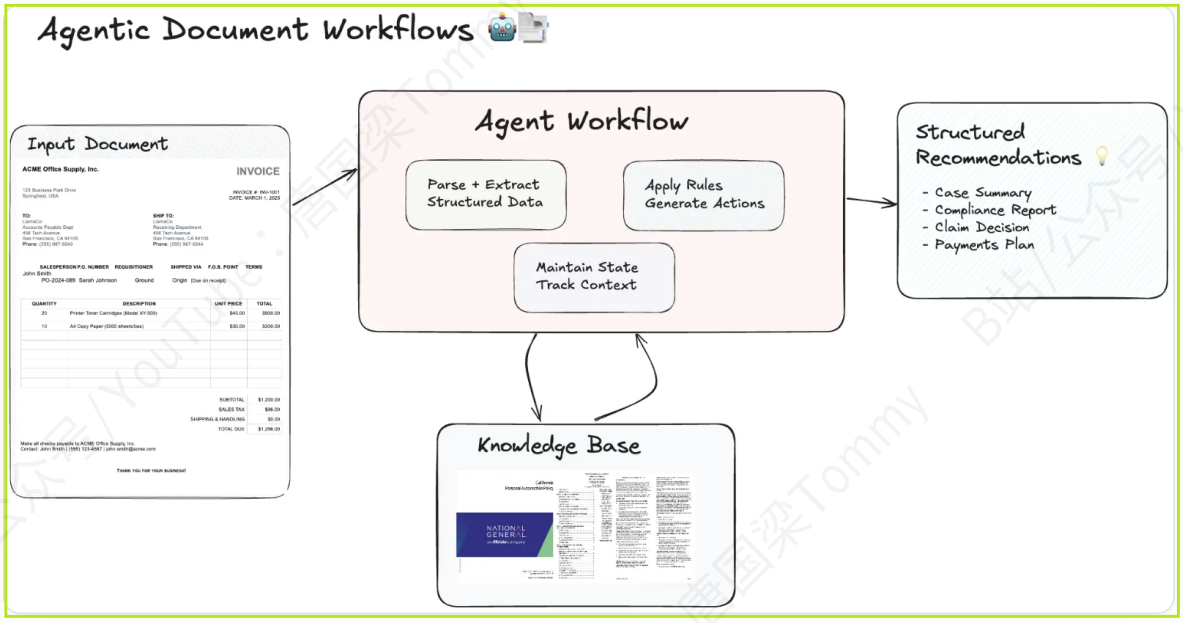
- **文档解析：** 智能体解析文档结构，识别关键信息。
- **信息提取：** 结合 RAG 技术，从文档中提取相关信息。
- **结构化输出：** 将提取的信息组织成结构化格式，便于后续处理。

适用场景：

- 需要处理大量文档的应用，如合同审查、报告生成、表单填写等。

优势：

- 显著提高文档处理的效率和准确性。
- 减少人工干预，降低操作成本。



8. RAG vs Agentic RAG 总结比较

在 RAG 领域，人们正从**“静态检索-后生成”**迈向**“具备规划、调用多工具、可自我反思的 Agentic RAG”**。后者把整条pipeline交给 AI Agent来主导：按需拆解任务、调用外部工具、多轮检索-推理-评估-再检索，并可在多智能体协作中处理实时数据和复杂决策。

8.1 核心差异一览

维度	传统 RAG	Agentic RAG
检索手段	单一向量数据库或搜索 API，静态一次性调用	可在多库/多模态源间多轮检索，检索策略由智能体动态调整
工具生态	基本无工具或仅嵌入检索	内建函数调用 / Toolformer-式自动用工具（计算器、SQL、代码执行等）
推理方式	单步"检索→生成"	规划-推理-反思-迭代的多步骤 reasoning（Self-Reflective / Recursive RAG）
协作模式	单一 LLM	多智能体分工：规划 Agent、检索 Agent、评估 Agent 等
实时数据	依赖预构建索引，难以对接流数据	支持 API / streaming 源，实时查询与融合
结果质量控制	无内置校验，幻觉由开发者外部兜底	Agent 级重检索、分级与自评估，显著降低幻觉率
适用场景	FAQ、静态知识库问答	金融行情解读、数据分析助手、企业 Copilot 等需要决策与写作的一体化工作流

8.2 能力深度对比

1️⃣ 工具与函数调用

Agentic RAG 通过 OpenAI Function Calling、LangChain Tool 等机制，把 LLM 生成的结构化调用参数映射到外部 API；智能体可以自动决定何时调用计算器、SQL、WebSearch 等工具，从而突破单纯检索的上限。

2️⃣ 多轮推理与自我反思

- **Recursive / Self-Reflective 回路**：模型先生成答案，对答案或检索结果自评打分，不足时重写查询或补充文档，再次生成，直至评分达标。
- **多级排序-质检**：检索阶段可引入二阶段 reranker；生成后用专门的 Critic Agent 追问事实依据，过滤幻觉。

3️⃣ 多智能体协作

Pathway、LangGraph 等框架提供基于消息流的多 Agent 流程编排：Planner 负责分解任务，Retriever 负责查询，Analyzer 进行数据处理，Writer 完成最终输出——提高可伸缩性与容错性。

4️⃣ 实时与多模态数据融合

实时行情或 IoT 流水通过流检索组件接入；同时可在文本外加入图像、表格检索，对结果进行跨模态融合，形成"动态-多模态 RAG"新形态。

Agentic RAG 把**检索增强**与**智能体规划**结合，补齐了传统 RAG 在多步骤推理、实时数据与任务自动化上的短板，是构建下一代企业 Copilot、智能决策助理的关键范式。通过分层代理、工具调用、反思循环与严格安全治理，可以在保证成本可控的前提下，显著提升系统的灵活性、准确性与扩展能力。

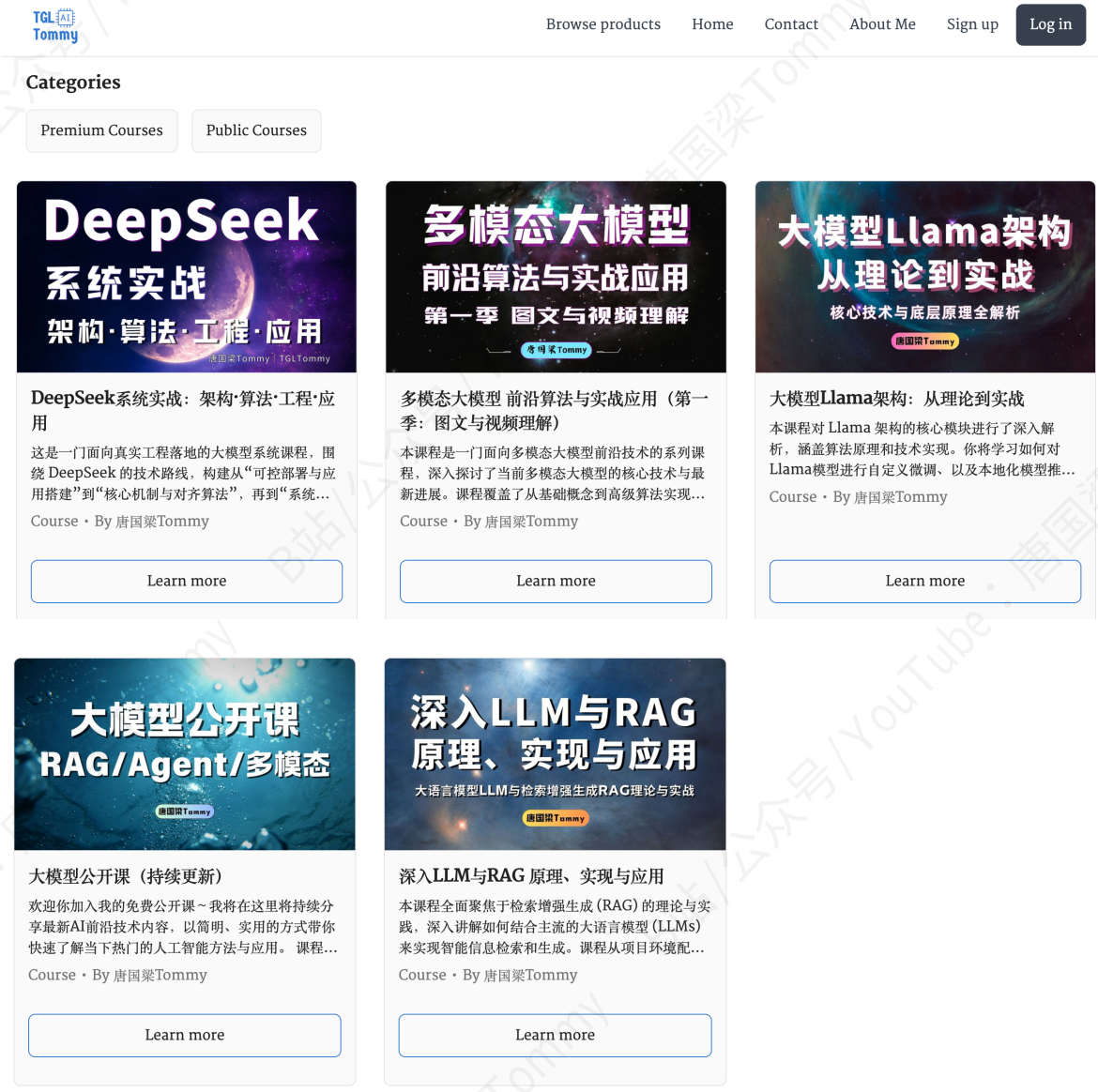
AI进阶学习

B站课堂：

https://space.bilibili.com/474347248/pugv

https://space.bilibili.com/3546748815411393/pugv

官方网站： https://www.tgltommy.com/ (国内访问需科学上网)



如果你希望系统掌握大模型核心技术、以及Agent应用开发，推荐你学习我刚上线的这门新课：**《DeepSeek 系统实战：架构·算法·工程·应用》**，这是一套从模型微调、部署，到强化学习训练的系统学习路线，课程以企业级落地为目标，你将掌握LLM核心原理、Agentic RAG、MoE/MLA/MTP机制拆解、PPO/GRPO强化学习与工业级DeepSeek-OCR多模态实战等，**想系统掌握并落地这些能力，就从这门课开始。**点击以下链接查看课程详情：

B站：https://www.bilibili.com/cheese/play/ss556613313

官网（国内访问需科学上网）：https://www.tgltommy.com/p/deepseek