



2025/12/01

RAG的进阶之路：从静态检索到 Agentic RAG的“纠错”机制研究

Evolution of RAG: From Static Retrieval to Corrective
Mechanisms in Agentic RAG

汇报人：rancan



CONTENTS

目录

01

趋势与选题

02

核心文献拆解

03

实施与复现计划





01

趋势与选题



技术演进：从“静态流程”到“动态智能体”（Agentic RAG）



Naive RAG（传统 RAG）

Agentic RAG（智能体 RAG）

核心逻辑： 线性链（Linear Chain）

核心逻辑： 决策图（Decision Graph）

Query → Retrieve → Generate

LLM 作为大脑动态规划路径



关键痛点（Pain Points）：

关键进化（Key Evolution）：

1. 盲目信任（Blind Trust）： 假设检索必对，无判断能力。

1. 自反思（Self-Reflection）： 引入“评估节点”，实现自我纠错与闭环。

2. 缺乏弹性（Rigid）： 简单/复杂问题走同一流程。

2. 工具化（Tool Use）： 向量检索降级为工具箱的一个选项（vs. Web Search）。

3. 幻觉风险： 检索错误 → 答案错误。

研究版图：Agentic RAG 的四大前沿方向



从流程到决策

动态决策 (Dynamic Decision)
智能路由 (Router: RAG vs. Web)
工具化 (Tool-use)



让大脑更聪明

知识冲突 (Conflict Resolution)
查询自适应 (Query Adaptive)
智能记忆 (Agentic Memory / RL)



工程系统

知识保鲜 (Knowledge Update)
效能平衡 (Cost vs. Quality)
动态评测 (New Eval Metrics)



多模态拓展

垂直场景 (Vertical Domain)
细粒度对齐 (Fine-grained Alignment)
跨模态推断 (Reasoning)

选择落脚：检索评估与纠错机制 (Corrective RAG)



选择理由

1. 典型性 (Representative):

CRAG 是 Agentic RAG 中“决策模块”的典型代表。搞懂了 CRAG 的“评估-决策-执行”闭环，就理解了整个智能体 RAG 的核心逻辑。

2. 可行性 (Feasibility):

实现门槛低：核心评估器仅需轻量级模型 (T5-Large, 0.77B) 即可实现，无需昂贵的大模型训练资源，适合快速上手验证。

3. 实验范式成熟 (Methodology):

标准化参照：提供了成熟的 RAG 实验框架，包括数据集选择 (PopQA, PubHealth) 和 Baseline 设置 (Self-RAG, Standard RAG)。

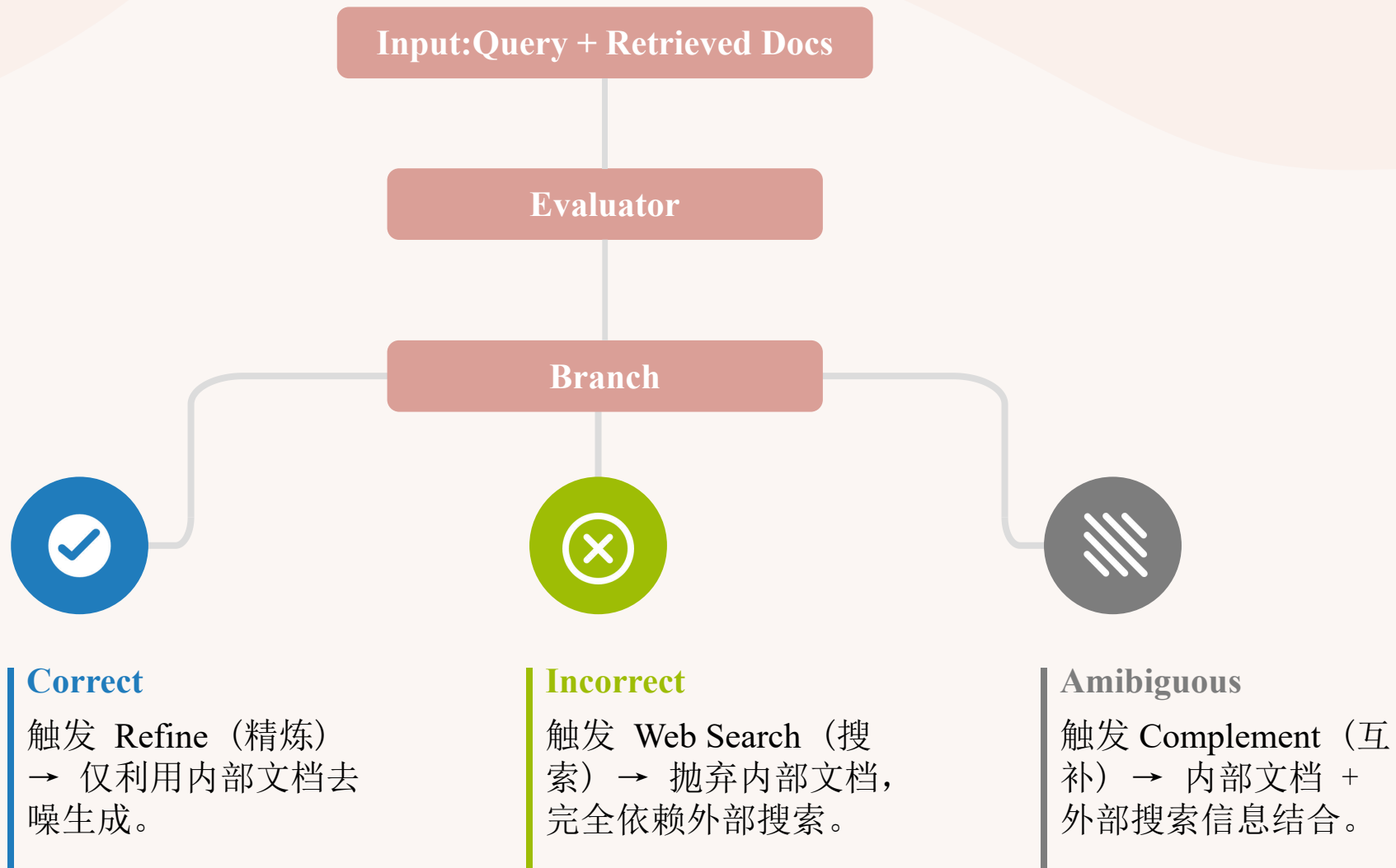
练兵场：通过复现该模块，可以系统掌握 RAG 论文的“运行流程”与“对比论证逻辑”，为后续改进打下基础。



02

核心文献拆解

CRAG 的核心机制——“红绿灯”判别系统



组件细节：轻量级评估器与先拆后拼



检索评估器 (Retrieval Evaluator)

模型选择： T5-Large (0.77B)。

优势： 相比 Self-RAG 使用的 LLaMA-2 (7B)，参数量减少 10倍，推理极快，不占用生成模型的显存。

训练目标： 判别 Query 与 Document 的相关性（二分类/打分任务）。

知识精炼 (Knowledge Refinement)

策略： Decompose-then-Recompose（先拆后拼）。

步骤：

Decompose: 将长文档切分为细粒度 Strip。

Filter: 再次利用评估器给每个 Strip 打分，过滤无关噪声。

Recompose: 拼接高分片段作为最终 Context。



实验论证——如何证明方法的有效性？



泛化性验证

数据集选择：

覆盖了 短文本（PopQA）、
长文本（Biography）
和 推理任务
（PubHealth）。

结论：

证明方法不偏科，在不
同长度和类型的生成任
务上均有效。



适配性验证

即插即用：

测试了不同底座
（LLaMA2, Alpaca）。

结论：

证明 CRAG 是一个通用插
件，不依赖底座模型的特
定能力（对比 Self-RAG
强依赖底座指令微调）。



鲁棒性压力测试

噪声实验：

人为向检索池注入噪声文档。

结论：

随着噪声比例上升，CRAG
的性能下降曲线最平缓，证
明“纠错机制”生效了。



消融实验验证

实验设置：

采用控制变量法，分别移除中
判别分支任一状态，或移除
“知识精炼”中的具体操作。

结论：

移除任何模块性能均下降，证
明系统“缺一不可”。特别是
Ambiguous（模糊状态）的移
除导致性能显著下滑，反向验
证了它作为“软性缓冲层”对
于防止错误决策的关键价值。



03

实施与复现计划

批判性思考：CRAG 的局限性与改进空间



阈值刚性



评估器瓶颈



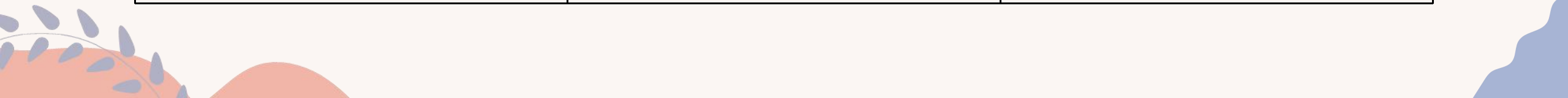
网络搜索的代价

局限：

论文使用了固定的置信度阈值判别文档可信度。但在不同领域（如医疗 vs. 闲聊），模型所需的置信度应该是不同的。	整个系统的生死完全依赖于那个轻量级的 T5 评估器。如果评估器在“知识冲突”场景下（即外部知识与内部记忆打架）判断失误，会导致错误的纠正。	论文在 Section 5.8 中虽然分析了 FLOPs，但掩盖了 Web Search 带来的巨大时间延迟。在“Ambiguous”状态下既做检索又做生成，延迟是双倍的。
---------------------------------------------------------	-----------------------------------------------------------------------	----------------------------------------------------------------------------------------

改进思路：

是否能将“固定阈值”改为“自适应阈值”？甚至引入强化学习（RL），让 Agent 根据问题的难度动态学习何时该“查”，何时该“停”。	引入 Conflict-aware（冲突感知）机制。当检测到检索内容与模型内隐记忆存在强冲突时，应该强制触发“红灯”或进行更深度的校验，而不仅仅看相关性分数。	论文在 Section 5.8 中虽然分析了 FLOPs，但掩盖了 Web Search 带来的巨大时间延迟。在“Ambiguous”状态下既做检索又做生成，延迟是双倍的。
--------------------------------------------------------------------	---------------------------------------------------------------------------------	----------------------------------------------------------------------------------------





复现路线

Phase 1: 全量复现

任务：跑通官方开源代码，验证 Table 1 中的 SOTA 结果。

环境对齐： 配置与论文一致的依赖库，固定随机种子 (Seed)。

数据准备： 直接使用官方提供的预处理好的 PopQA 数据和 web_search_results (避免因爬虫失败导致复现失败)。

基准确认： 运行 CRAG_Inference.py，确保 Accuracy 指标与论文一致。

Phase 2: 模块提取

任务：将核心组件剥离为独立的、可复用的 Python 类，构建个人的“实验工具箱”。

目的：摆脱原始“批处理脚本”的束缚，让这些模块能嵌入我未来的 Agentic RAG 框架中。

Phase 3: 改进与探索

任务：针对“局限性”进行针对性优化。

计划：

冲突测试：构造“知识冲突”样本，测试原版 CRAG 在此场景下的失误率。

机制升级：尝试引入 Conflict-aware (冲突感知) 机制，或用更高效的 BGE-Reranker 替换 T5 进行速度/精度权衡实验。

THE END

谢谢