

Towards AI Search Paradigm

Yuchen Li, Hengyi Cai, Rui Kong, Xinran Chen, Jiamin Chen, Jun Yang, Haojie Zhang, Jiayi Li, Jiayi Wu, Yiqun Chen, Changle Qu, Wenwen Ye, Lixin Su, Xinyu Ma, Lingyong Yan, Long Xia, Daiting Shi, Junfeng Wang, Xiangyu Zhao, Jiashu Zhao, Haoyi Xiong, Shuaiqiang Wang, Dawei Yin[†]

Baidu Search
{yuchenli13, yindawei}@acm.org

Abstract

In this paper, we introduce the AI Search Paradigm, a comprehensive blueprint for next-generation search systems capable of emulating human information processing and decision-making. The paradigm employs a modular architecture of four LLM-powered agents (Master, Planner, Executor and Writer) that dynamically adapt to the full spectrum of information needs, from simple factual queries to complex multi-stage reasoning tasks. These agents collaborate dynamically through coordinated workflows to evaluate query complexity, decompose problems into executable plans, and orchestrate tool usage, task execution, and content synthesis. We systematically present key methodologies for realizing this paradigm, including task planning and tool integration, execution strategies, aligned and robust retrieval-augmented generation, and efficient LLM inference, spanning both algorithmic techniques and infrastructure-level optimizations. By providing an in-depth guide to these foundational components, this work aims to inform the development of trustworthy, adaptive, and scalable AI search systems.

Keywords: AI Search, Web Search, Agentic Search, Search Engine, Large Language Models, Retrieval-Augmented Generation, Task Planning, Multi-Agent System

[†]Corresponding author

迈向AI搜索范式

李宇辰、蔡恒毅、孔瑞、陈欣然、陈佳敏、杨军、张浩杰、李佳怡、吴佳怡、陈一群、曲长乐、叶文文、苏立新、马欣雨、闫凌勇、夏龙、史代婷、王俊峰、赵翔宇、赵佳舒、熊浩毅、王帅强、尹大伟[†]

百度搜索
{yuchenli13, yindawei}@acm.org

摘要

在本文中，我们提出了AI搜索范式，这是一套用于下一代搜索系统的综合性蓝图，能够模拟人类的信息处理与决策过程。该范式采用由四个基于大语言模型的智能体（主控智能体、规划智能体、执行智能体和写作智能体）构成的模块化架构，可灵活适配从简单的事实查询到复杂的多阶段推理任务等各类信息需求。这些智能体通过协同工作流程动态配合，以评估查询复杂度、将问题拆解为可执行的计划，并统筹工具调用、任务执行以及内容生成工作。我们系统地阐述了实现这一范式的关键方法，涵盖任务规划与工具集成、执行策略、一致且稳健的检索增强生成技术，以及高效的大语言模型推理机制，涉及算法技术与基础设施层面的优化。通过深入讲解这些基础组件，本研究旨在为开发可信赖、自适应且可扩展的AI搜索系统提供指导。

关键词：AI搜索、网页搜索、智能体搜索、搜索引擎、大语言模型、检索增强生成、任务规划、多智能体系统

[†]通讯作者

Contents

1	Introduction	4
2	System Overview	6
3	Task Planner	10
3.1	Background	10
3.2	Task Universe and the MCP Abstraction	12
3.3	Dynamic Capability Boundary	12
3.3.1	Refinement of Tool API Documents	13
3.3.2	Tool Clustering in MCP	15
3.3.3	Query-Oriented Tool Retrieval	16
3.4	DAG-based Task Planning	18
3.5	Master-Guided Re-Action	19
3.6	Planner Optimization with RL Strategy	19
4	Task Executor	21
4.1	Background	21
4.2	LLM Preference Alignment	23
4.2.1	LLM Labeling	23
4.2.2	Reference Selection	26
4.2.3	Generation Reward	26
4.2.4	Distillation of LLM Ranking	27
4.3	Lightweight System	27
4.3.1	Lightweight Retrieval	28
4.3.2	Lightweight Ranking	29
4.4	LLM-Augmented Features	29
5	LLM-based Generation	31
5.1	Background	31
5.2	Robust RAG System	31
5.2.1	Adversarial Setups for Robust RAG	31
5.2.2	Multi-Agent Adversarial Training	32
5.3	RAG Task Alignment	33
5.3.1	RAG Task Requiriements	33
5.3.2	PA-RAG Optimization Objectives	34

目录

1	简介 引言	4
2	系统概述	6
3	任务规划器	10
3.1	背景	10
3.2	任务宇宙与MCP抽象	12
3.3	动态能力边界	12
3.3.1	工具API文档的优化	13
3.3.2	MCP中的工具聚类	15
3.3.3	面向查询的工具检索	16
3.4	基于有向无环图的任务规划	18
3.5	主引导式再行动	19
3.6	基于强化学习的规划器优化	19
4	TaskExecutor	21
4.1	背景	21
4.2	大语言模型偏好对齐	23
4.2.1	大语言模型标注	23
4.2.2	参考信息选择	26
4.2.3	生成奖励	26
4.2.4	大语言模型排序的蒸馏	27
4.3	轻量级系统	27
4.3.1	LightweightRetrieval	28
4.3.2	轻量级排序	29
4.4	基于大语言模型增强的功能	29
5	大语言模型-基于 Generation	31
5.1	背景介绍	31
5.2	RobustRAGSystem	31
5.2.1	Robust RAG的对抗性设置	31
5.2.2	多智能体对抗训练	32
5.3	RAG任务对齐	33
5.3.1	RAG任务要求	33
5.3.2	PA-RAG优化目标	34

5.3.3	Methodology of PA-RAG	34
5.4	Optimization with User Feedback	36
5.5	Multi-Agent Joint Optimization	38
6	Light-Weighting LLM	42
6.1	Background	42
6.2	Algorithmic-Level Lightweighting	43
6.2.1	Local Attention	43
6.2.2	Model Pruning	44
6.3	Infrastructure-Level Lightweighting	45
6.3.1	Output Length Reduction	45
6.3.2	Semantic Caching	46
6.3.3	Quantization	47
6.3.4	Prefill-Decode Separating Deployment	47
6.3.5	Speculative Decoding	48
7	Evaluation	48
7.1	Human Evaluation	48
7.2	Online A/B Test	50
7.3	Case Study	50
8	Conclusion	51
	Acknowledge	51

5.3.3	PA-RAG的方法论	34
5.4	基于用户反馈的优化	36
5.5	多智能体联合优化	38
6	轻量化 Weighting LLM 4	2
6.1	背景介绍	42
6.2	算法层面的轻量化	43
6.2.1	局部注意力	43
6.2.2	模型剪枝	44
6.3	基础设施层面的轻量化	45
6.3.1	输出长度缩减	45
6.3.2	语义缓存	46
6.3.3	量化	47
6.3.4	预填充与解码分离部署	47
6.3.5	推测性解码	48
7	评估	48
7.1	人工评估	48
7.2	在线A/B测试	50
7.3	案例研究	50
8	结论	51
	致谢	51

1. Introduction

In an age where individuals are constantly managing an overwhelming flow of data, *Information Seeking* (IS), an active behavior that constructs new cognition when faced with knowledge gaps (Dervin et al., 1983; Wilson, 1981), has emerged as a vital process for making informed decisions and solving complex problems. The advent of web search engines marked a significant leap in IS. Web search engines function as specialized systems built upon fundamental *Information Retrieval* (IR) technologies, systematically crawling, indexing, and retrieving internet-based information in response to user queries. Over the past decades, the field of IR has undergone transformative generational shifts to achieve superior performance. Researchers and practitioners have progressed systematically from early lexical models to machine learning paradigms, and ultimately, to Large Language Models (LLMs).

Lexical IR technologies predominantly rely on keyword matching techniques, encompassing vector space models, probabilistic frameworks, and traditional language models (Brin and Page, 1998; Mikolov, 2013; Robertson et al., 2009; Yang, 2005). These approaches represent documents and queries as bags of words, estimating relevance based on the exact or partial overlap of terms. While effective for exact term matching, they inherently struggle with semantic mismatches and vocabulary variation—failing, for example, to associate synonymous expressions or contextual nuances. Such limitations have increasingly highlighted the need for more robust retrieval mechanisms. This motivated the field’s transition toward semantic-aware retrieval paradigms capable of deeper information assimilation.

Learning-to-Rank (LTR) methodologies (Burgess et al., 2005; Cao et al., 2007; Qin et al., 2010) emerged from the pursuit of more relevant and higher-quality search results, driving the application of machine learning to ranking problems. This next generation of information retrieval systems replaced heuristic ranking approaches with machine-learned models that directly optimize ranking objectives. LTR techniques leveraged feature engineering—incorporating signals including text-matching scores, document structure metadata, authority metrics, and user behavior signals (e.g., click-through rates)—combined with supervised machine learning to optimize relevance ranking. This represented a significant advancement in ranking effectiveness, consistently placing more relevant documents in top positions compared to lexical retrieval models. The output of such systems is a ranked list of documents, requiring users to perform an additional step to access the target information: clicking on relevant documents and then locating or synthesizing the needed content — revealing a gap between system outputs and true information needs.

The advent of LLMs (Achiam et al., 2023; Bubeck et al., 2023; Dubey et al., 2024; Mesnard et al., 2024; Wei et al., 2022a) has enabled information retrieval systems to bridge this gap, transitioning from document retrieval to direct generation of precise contextualized answers. However, current RAG systems primarily function as single-shot answer generators and struggle with queries involving complex information needs, such as innovation-related inquiries, emotional nuance, or proactive planning, the orchestration of diverse tools, and deep reasoning across multiple, sometimes conflicting, pieces of evidence (Fan et al., 2024; Zhao et al., 2024). These systems’ output quality is critically dependent on the initial document retrieval phase, lacking both robustness in handling imperfect retrieval and the sophisticated reasoning capabilities needed to address truly multifaceted user needs. For instance, a seemingly straightforward question like "Who was older, Emperor Wu of Han or Julius Caesar, and by how many years?" poses significant challenges for current systems. While individual birth dates for both historical figures may be easily retrievable, no single document explicitly compares their ages. This requires the system to perform a complex, multi-stage process: (1) retrieving

1. 引言

在人们需要不断应对海量数据流的时代，信息检索（Information Seeking, IS）作为一种在面临知识缺口时构建新认知的主动行为（Dervin等人，1983；Wilson, 1981），已成为做出明智决策以及解决复杂问题的关键过程。网络搜索引擎的出现标志着信息检索领域的重大飞跃。网络搜索引擎是基于基础信息检索（Information Retrieval, IR）技术构建的专业系统，能够针对用户查询系统地抓取、索引并获取互联网上的相关信息。过去几十年间，信息检索领域经历了多轮具有变革意义的代际演进，以实现更优异的性能。研究人员与实践者循序渐进地从早期的词汇模型发展到机器学习范式，最终迈向了大语言模型（Large Language Models, LLMs）阶段。

词汇级信息检索技术主要依赖关键词匹配机制，包括向量空间模型、概率框架以及传统语言模型（Brin和Page, 1998；Mikolov, 2013；Robertson等, 2009；Yang, 2005）。这些方法将文档与查询表示为词袋形式，依据词语的完全或部分重叠程度来估算相关性。尽管在精确词项匹配方面表现良好，但它们天生难以处理语义不匹配及词汇差异问题——例如无法识别同义表达或语境细微差别。此类局限性日益凸显出对更稳健检索机制的需求，也推动了该领域向能够更深入理解信息的语义感知型检索范式转型。

学习排序（LTR）方法（Burgess等, 2005；Cao等, 2007；Qin等, 2010）源于对更相关、更高质量搜索结果的追求，从而促使机器学习被应用于排序问题。新一代信息检索系统用机器学习模型取代了传统的启发式排序方式，直接优化排序目标。LTR技术借助特征工程，整合文本匹配得分、文档结构元数据、权威性指标以及用户行为信号（如点击率）等信息，并结合监督式机器学习来优化相关性排序。这在排序效果上实现了显著进步，相比词汇级检索模型，能将更相关的文档置于靠前位置。不过，这类系统的输出仅为排序后的文档列表，用户仍需额外操作才能获取目标信息：点击相关文档后，再从中定位或整合所需内容——这反映出系统输出与真实信息需求之间仍存在差距。

大型语言模型的出现（Achiam等人, 2023；Bubeck等人, 2023；Dubey等人, 2024；Mesnard等人, 2024；Wei等人, 2022a）使得信息检索系统能够填补这一空白，从传统的文档检索转向直接生成精准且符合语境的答案。然而，当前的RAG系统主要仅能作为单次答案生成器使用，在处理涉及复杂信息需求的查询时表现欠佳，例如与创新相关的询问、情感细微差别的识别、前瞻性规划、多种工具的协同调用，以及针对相互矛盾的证据进行深度推理等（Fan等人, 2024；Zhao等人, 2024）。这些系统的输出质量严重依赖于最初的文档检索阶段，既缺乏应对检索不完备情况的鲁棒性，也缺少满足用户真正多样化需求所需的复杂推理能力。比如，像“汉武帝和凯撒大帝谁年纪更大？大多少岁？”这样看似简单的问题，对现有系统而言却极具挑战性。尽管这两位历史人物的出生日期可能很容易就能查到，但没有任何一份文档会直接对比他们的年龄。这就要求系统执行一个复杂的多阶段流程：（1）

and verifying each emperor’s birth year from separate sources, (2) resolving potential conflicts between contradictory records, (3) calculating the age difference, and (4) synthesizing the final comparison. Such queries demand multi-step reasoning capabilities—the ability to decompose complex questions, execute sequential sub-queries, evaluate evidence reliability, and integrate intermediate results into a coherent answer.

This paper establishes that the historical evolution of IR systems necessitates a fundamental advancement: cognitive architectures that genuinely emulate human information foraging behaviors and multi-stage reasoning processes. To meet this need, we propose a revolutionary IS paradigm, **AI Search Paradigm** — a collaborative, multi-agent framework powered by large language models that reasons, plans, and executes complex problem — solving strategies on the user’s behalf. Drawing inspiration from early work on computational algorithms (Knuth, 1997) and system interactions (Cohen et al., 2007), this AI search paradigm coordinates specialized agents, each responsible for a distinct phase of the information-seeking process, to jointly deliver accurate, context-rich answers that mirror human-level inquiry. The key agent roles include:

- **Master Agent.** *Master* serves as the initial point of contact, responsible for analyzing the user’s query to assess its complexity and intent. Based on the nature and difficulty of the problem, *Master* dynamically coordinates and assembles an appropriate team of subsequent agents. This is a unique feature of the proposed paradigm, compared to traditional IR systems, which incorporate static query understanding modules (Xiong et al., 2024), and RAG systems, where initial query analysis typically leads to a fixed processing pipeline (Ma et al., 2023). In addition, it continuously evaluates the performance of subordinate agents; in the event of task failures, *Master* conducts reflective analysis and guides the team to re-plan and re-execute accordingly.
- **Planner Agent.** *Planner* is only invoked for complex queries that require multi-step reasoning or information gathering. Given the user’s input query, it selects appropriate tools from tool platforms—e.g., the conceptualized Model-Context Protocol (MCP) Servers Platform (Edwards, 2025)—to dynamically adjust the capability boundaries of the LLMs. *Planner* then decomposes the overarching query into a structured sequence of manageable sub-tasks, represented as a Directed Acyclic Graph (DAG). This step involves strategic planning of how to approach the problem. While advanced RAG systems might perform query decomposition or multi-hop reasoning (Ma et al., 2023), *Planner*’s function of generating a comprehensive, explicit DAG of sub-tasks with interdependencies and potential tool bindings represents a more flexible and proactive planning capability than typically found in standard RAG systems (Lewis et al., 2020b).
- **Executor Agent.** *Executor* is responsible for carrying out the simple query or the individual sub-tasks defined by *Planner*. A crucial aspect of its role is the invocation of external tools, selected by *Planner* from tool platforms (e.g. MCP), to gather information or perform computations necessary for each sub-task. It also evaluates the outcomes of these executions. Traditional IR systems execute internal algorithms, and while some RAG models can use tools, *Executor*’s role in managing a diverse set of tools for DAG-derived sub-tasks, including outcome assessment and fallback mechanisms, suggests a more robust and orchestrated execution layer (Qu et al., 2025b).
- **Writer Agent.** The final agent in the workflow, *Writer*, synthesizes the information and results gathered from all completed sub-tasks. Its goal is to generate a coherent, contextually rich, and potentially multi-perspective response to the user’s original query, including necessary filtering and disambiguation. While the generator component in RAG systems also synthesizes answers (Lewis et al., 2020b), *Writer* in the AI search paradigm is envisioned to work with a more structured and diverse set of inputs from various planned

从不同来源分别核实每位皇帝的出生年份；（2）解决记录之间可能存在的矛盾；（3）计算年龄差；（4）整合得出最终结论。这类查询要求系统具备多步骤推理能力——即能够拆解复杂问题、依次执行子查询、评估证据的可靠性，并将中间结果整合成连贯的答案。

本文指出，信息检索系统的历史演进迫切要求实现一次根本性突破：即开发出能够真正模拟人类信息搜寻行为及多阶段推理过程的认知架构。为满足这一需求，我们提出了一种革命性的信息检索范式——**AI搜索范式**。这是一种由大型语言模型驱动、具备协作能力的多智能体框架，能够替用户进行推理、规划并执行复杂的问题解决策略。该AI搜索范式的灵感源自早期关于计算算法（Knuth, 1997）以及系统交互的研究（Cohen等人，2007），它通过协调各类专业智能体来共同完成信息获取过程中的不同环节，从而给出与人类查询逻辑相契合的准确且富含上下文信息的答案。其中关键的智能体角色包括：

- **主智能体。** 主智能体作为初始接触点，负责分析用户的查询内容以判断其复杂程度与意图。根据问题的性质与难度，主智能体会动态协调并组建合适的后续智能体团队。相较于传统的信息检索系统（其采用静态的查询理解模块，Xiong等人，2024）以及RAG系统（初始查询分析后通常遵循固定的处理流程，Ma等人，2023），这正是所提出范式独具的特色。此外，主智能体还会持续评估下属智能体的表现；一旦任务出现失败情况，主智能体便会展开反思性分析，指导团队重新规划并继续执行相关任务。

- **规划器智能体。** 规划器仅会在需要多步骤推理或信息收集的复杂查询中被调用。针对用户的输入查询，它会从工具平台中选取合适的工具——例如已概念化的模型上下文协议（MCP）服务器平台（Edwards, 2025）——以此来动态调整大语言模型的能力边界。规划器随后会将整体查询拆解为一系列结构化的、易于处理的子任务，这些子任务以有向无环图的形式呈现。这一步骤涉及对问题解决方式的战略性规划。虽然先进的检索增强生成系统也能执行查询分解或多跳推理操作（Ma等人，2023），但规划器的功能在于.....

生成包含各子任务间依赖关系及潜在工具绑定的完整且明确的有向无环图，这种能力比传统检索增强生成系统中常见的规划方式更具灵活性与主动性（Lewis等人，2020b）。

- **执行器智能体。** 执行器负责处理简单查询，或是执行由规划器所定义的各个子任务。其职责的关键环节在于调用由规划器从工具平台（如MCP）中选出的外部工具，以此收集信息或完成各子任务所需的计算操作。它还会评估这些执行结果。传统的信息检索系统依靠内部算法运行，尽管部分检索增强生成模型也能使用工具，但执行器在管理用于有向无环图派生子任务的各类工具方面所发挥的作用，包括结果评估与回退机制，体现出其作为更稳健且协调有序的执行层的能力（Qu等人，2025b）。

- **写作智能体。** 作为工作流中的最后一个智能体，写作智能体会整合所有已完成的子任务所收集到的信息与结果。其目标是针对用户的原始查询生成逻辑连贯、内容丰富且可能包含多视角解读的回复，同时完成必要的筛选与歧义消除工作。尽管检索增强生成系统中的生成器组件同样能够合成答案（Lewis等人，2020b），但在面向AI的搜索范式下，写作智能体被设计为能够处理来自各类规划任务的更为结构化且多样化的输入。

sub-tasks, aiming for a more comprehensive and conversational synthesis than generation from a flat list of retrieved documents.

This paradigm emphasizes adaptability through dynamic team configurations and aims to achieve comprehensive and context-aware information synthesis, moving beyond simple document retrieval (Abril and Plant, 2007).

This paper presents a comprehensive exploration of the proposed **AI Search paradigm**. Rather than focusing on a singular system and its evaluation, the contributions of this work lie in the exploration of essential techniques for the AI Search paradigm and can be summarized as follows:

- **Conceptualizing a New Paradigm.** We introduce and detail a dynamic, modular multi-agent architecture for AI Search, featuring specialized agents that perform distinct search-oriented roles (e.g. mastering, planning, execution, and synthesis). The system dynamically coordinates these agents on the fly to adapt to query complexity, enabling more flexible and powerful reasoning capabilities than traditional approaches. This represents a significant departure from both the monolithic models of conventional web search and the “linear” pipelines of Retrieval-Augmented Generation (RAG) systems.
- **Foraging Core Agentic Search Methodologies.** We identify and contextualize critical techniques for the core AI search process. This includes methods for *Advanced Task Planning*, including dynamic adjustment of LLMs’ capability boundary, advanced tool retrieval methods, DAG-based task planning, *Master*-guided reflection and re-action mechanism, and reinforcement learning-enhanced optimization strategy. It also covers *Flexible Task Execution strategies* that focus on aligning evidence retrieval with LLM’s preferences, lightweighting retrieval and ranking systems, and constructing LLM-augmented features.
- **Detailing Search-Oriented Generation and Optimization Techniques.** We subsequently examine methodologies for generating robust responses and optimizing the overall system. This discussion encompasses strategies for ensuring both robustness and alignment, as well as a joint multi-module optimization approach tailored for RAG systems. Finally, we address *Efficient LLM Inference Strategies*, termed “Lightning LLM’s Generation”, which include optimizations at both the algorithmic and infrastructure levels.
- **Demonstrating Improved User Experience Through Case Studies.** To showcase the practical advantages of our paradigm, this paper conducts extensive experiments and presents several case studies evaluated through user experience studies. Instead of relying on statistics of performance metrics, these cases illustrate how the system handles complex, multi-step queries from a user’s perspective. By walking through these examples, we demonstrate the qualitative improvements in the search experience, highlighting the system’s ability to deliver more comprehensive, accurate, and satisfying answers compared to previous technologies.

The overarching contribution of this paper is thus to provide a structured and detailed blueprint that assembles best-of-breed techniques from industry and academia, outlining an evolving design paradigm that can serve as a guide for future research and development in AI-driven search.

2. System Overview

In this section, we introduce a modular AI search paradigm designed to address the limitations of conventional retrieval-augmented systems in handling complex, multi-step information

它旨在实现比单纯基于检索到的文档列表进行生成更为全面且符合对话逻辑的整合结果。

该范式强调通过动态团队配置来实现适应性，旨在实现全面且具备上下文感知能力的信息合成，从而超越简单的文档检索（Abriland Plant, 2007）。

本文对所提出的**AI搜索范式**进行了全面探讨。本研究并非着眼于单一系统及其评估，其贡献在于探索AI搜索范式所需的关键技术，可概括如下：

- **新范式的概念化。**我们提出并详细阐述了用于AI搜索的动态模块化多智能体架构，该架构包含承担不同搜索任务的专用智能体。
这些智能体分别承担不同的导向性角色（如掌握、规划、执行与合成）。系统会实时动态调整这些智能体的协作方式以适应查询的复杂性，从而展现出比传统方法更灵活、更强大的推理能力。这标志着它既不同于传统网络搜索中的单体模型，也不同于检索增强生成（RAG）系统的“线性”流程。
- **核心智能体搜索方法论的研究。**我们识别并梳理了AI搜索核心过程的关键技术。其中包括高级任务规划方法，比如动态调整大语言模型的能力边界、先进的工具检索方法、基于有向无环图的任务规划、*Master*引导的反思与再行动机制，以及基于强化学习的优化策略。此外还包括灵活的任务执行策略，这类策略侧重于让证据检索与大语言模型的偏好相匹配、简化检索与排序系统，以及构建由大语言模型增强的特征。
- **详述面向搜索的生成与优化技术。**随后，我们探讨了生成鲁棒响应以及优化整个系统的方法。相关讨论涵盖了确保系统鲁棒性与一致性的策略，以及专为RAG系统设计的多模块联合优化方案。最后，我们介绍了高效的LLM推理策略，即所谓的“Lightning LLM生成技术”，这些策略涵盖了算法层面与基础设施层面的优化。
- **通过案例研究展示改进后的用户体验。**为了彰显该范式在实际应用中的优势，本文开展了大量实验，并呈现了多项经用户体验研究评估的案例。这些案例并非基于性能指标统计数据，而是从用户视角展现了系统处理复杂、多步骤查询的能力。通过剖析这些实例，我们展示了搜索体验在质量上的提升，凸显出该系统相较于以往技术能够提供更全面、准确且令人满意的答案。

因此，本文的核心贡献在于提供一份结构化且详尽的蓝图，整合了业界与学术界的最优技术，勾勒出一一种不断演进的设计范式，从而为未来人工智能驱动型搜索领域的研究与发展提供指引。

2. 系统概述

在本节中，我们介绍了一种模块化的AI搜索范式，旨在解决传统检索增强系统在应对复杂、多步骤信息需求时的局限性。

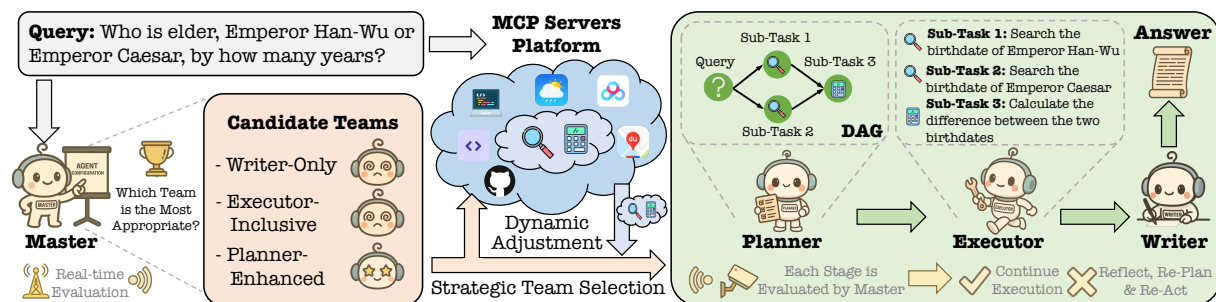


Figure 1 | The Overview of AI Search Paradigm.

needs. Inspired by human collaborative search behavior, the framework coordinates multiple specialized agents to evaluate, plan, execute, and synthesize in a scalable and context-aware manner. Specifically, the agents are assigned a range of specialized responsibilities, including proactively evaluating the complexity of user queries, configuring appropriate multi-agent teams, conducting task decomposition and planning, selecting suitable tools via tool platforms (e.g. the MCP servers platform), executing the selected tools to fulfill specific sub-tasks, and synthesizing the outcomes of all sub-tasks to generate comprehensive and multi-perspective responses. Such a coordinated architecture ensures a methodical progression of the search process toward coherent and comprehensive outcomes.

The multi-agent setting offers significant advantages over single-agent configurations, which often suffer from task overload and diminished efficiency when a single LLM-based agent is responsible for managing multiple complex responsibilities. By assigning well-defined roles to each agent, the system ensures clear task allocation and robust operational management, thereby preventing bottlenecks and enhancing overall performance. Moreover, dynamically configuring agent teams in response to varying levels of query complexity enhances the system's flexibility and adaptability across diverse information-seeking tasks. Building on this, the modular design further reinforces scalability, enabling the system to maintain consistently high performance in a wide range of complex and demanding scenarios.

Details of Agent Roles. As illustrated in Fig. 1, the AI search paradigm is implemented as a multi-agent collaborative system. Based on their designated functional roles, the system comprises four specialized agents: *Master*, *Planner*, *Executor*, and *Writer*. The specific tasks assigned to each agent are detailed as follows:

- **Master.** In the AI Search system, the Master agent acts as the team coordinator. It is responsible for analyzing the user's input query to assess its complexity and intent, and for assembling an appropriate team of specialized agents to handle the task — without directly participating in the downstream query processing itself. Specifically, the Master agent analyzes the input query along with relevant contextual information to assess its complexity and determine the optimal team configuration. For relatively simple queries, it may assign a minimal setup involving only the Writer or a combination of the Executor and Writer. For more complex queries, it initiates a larger team that includes the Planner agent to perform task decomposition and planning, enabling a step-by-step resolution of the query. Furthermore, the Master continuously monitors the performance of subordinate agents. In the event of task failure, it conducts a reflective analysis and directs the team to re-plan and re-execute accordingly.
- **Planner.** This agent is only invoked for complex queries that require multi-step reasoning or information gathering, and is responsible for task decomposition and strategic planning

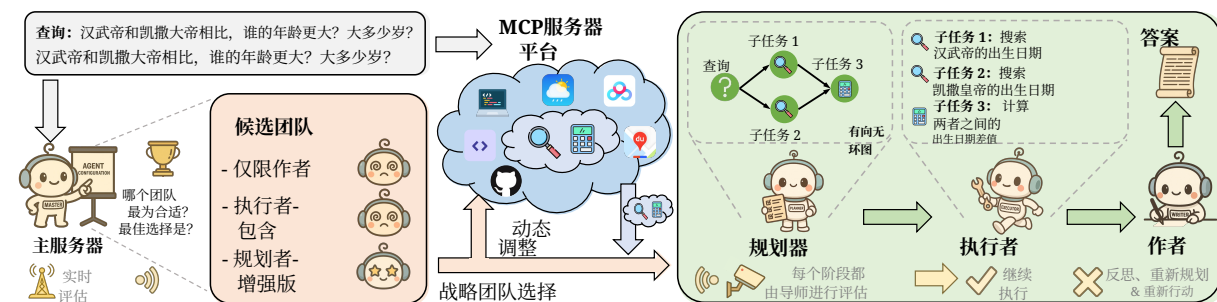


图 1 |：AI搜索范式概览。

该范式受到人类协作式搜索行为的启发，能够协调多个专业智能体，以可扩展且具备语境感知能力的方式进行评估、规划、执行和整合。具体而言，这些智能体被赋予了各类专业化职责：主动评估用户查询的复杂程度、配置合适的多智能体团队、进行任务分解与规划、通过工具平台（如MCP服务器平台）挑选适配的工具、执行所选工具来完成具体子任务，以及整合所有子任务的结果以生成全面且多角度的回答。这种协同架构确保了搜索过程能够有条不紊地推进，最终得出连贯且详尽的结果。

多智能体架构相较于单智能体配置具有显著优势：当由单一基于大语言模型的智能体承担多项复杂任务时，往往会出现任务过载与效率下降的问题。通过为各智能体分配明确的角色，系统能够实现清晰的任务分配与稳健的运行管理，从而有效避免瓶颈并提升整体性能。此外，系统还能根据查询复杂度的变化动态调整智能体团队 구성，进而增强其在各类信息检索任务中的灵活性与适应性。在此基础上，模块化设计进一步强化了系统的可扩展性，使其能够在各种复杂且高要求的场景下始终保持出色的性能。

智能体角色详情。如图1所示，AI搜索范式是以多智能体协作系统形式实现的。根据其指定的功能角色，该系统包含四种专业智能体：*Master*、*Planner*、*Executor*以及*Writer*。各智能体所承担的具体任务如下：

- **Master.** 在AI搜索系统中，Master智能体扮演着团队协调员的角色。它负责分析用户的输入查询，评估其复杂度与意图，并组建合适的专业智能体团队来处理该任务——不过它本身并不直接参与后续的查询处理过程。具体而言，Master智能体会结合输入查询及相关上下文信息来评估任务复杂度，并确定最优的团队配置。对于较为简单的查询，它可能仅安排Writer智能体，或让Executor与Writer协同工作；而对于更复杂的查询，则会启动规模更大的团队，纳入Planner智能体来进行任务分解与规划，从而实现查询的逐步解决。此外，Master还会持续监控下属智能体的表现；一旦任务出现失败情况，它会展开反思性分析，并指导团队重新规划并执行相应操作。
- **规划器。**该智能体仅在涉及多步推理或信息收集的复杂查询中才会被调用，其职责是进行任务分解与战略规划。

in response to complex queries, enabling the AI search system to address them through a sequence of simpler, structured sub-tasks. Its core function is to select appropriate tools from a tool set (e.g. the MCP platform) to dynamically adjust the LLMs' capability boundaries and to generate a DAG that encodes a global task plan. In this DAG, each node represents an atomic, schedulable sub-task, and each edge captures a conditional dependency between sub-tasks, derived from the Planner's reasoning process. Furthermore, when the Master evaluates sub-tasks as having execution errors or lacking critical data, the Planner, under the Master's direction, undertakes a reconfiguration of the DAG.

- **Executor.** In the task execution phase, the Executor plays a central role in carrying out the task and evaluating its outcome, ensuring successful completion and compliance with predefined requirements. Each sub-task arrives at execution with either an external tool pre-assigned or a "tool-free" configuration to be handled solely by the Executor using its own built-in capabilities. During execution, the Executor may repeatedly invoke the assigned tool based on task complexity, while continuously assessing whether the accumulated outputs are sufficient to fulfill the sub-task's objectives. Once the evaluation indicates that the requirements are satisfied, the current output is returned; otherwise, the execution continues iteratively. In cases where the assigned tool becomes unresponsive or fails, the system seamlessly switches to a functionally equivalent backup tool within the same tool module, thereby maintaining execution continuity and robustness. For example, consider a sub-task that requires acquiring relevant knowledge via a web search tool. The Executor invokes the bound *web search tool* using the generated sub-queries based on the current sub-task and performs multiple rounds of search as needed. After each round, it evaluates the retrieved documents to determine whether the information sufficiently supports the factual or contextual requirements of this sub-task. If not, these sub-queries are refined and the tool is re-invoked. Once adequate coverage is achieved, the results are finalized and returned. Through this integrated approach, the Executor effectively combines execution and quality control within a unified mechanism, ensuring robust, accurate, and context-aware execution of sub-tasks.
- **Writer.** This agent is responsible for synthesizing the information and results gathered from all completed preceding sub-tasks and generating the final response to the user's query. Its goal is to generate a coherent, contextually rich, and potentially multi-perspective response to the user's original query, including necessary filtering and disambiguation. During this process, it identifies the semantic relationships and logical structures among the outputs of multiple sub-tasks, and reorganizes the information using either predefined templates or adaptive structuring strategies. This enables the Writer to produce a response that is coherent, well-organized, and easy to comprehend. When redundancy or semantic inconsistency exists across sub-task results, the Writer performs content filtering and disambiguation to ensure that the final output is accurate and free from misleading information. Beyond simply answering the core query, the generated response also incorporates contextual background and explanatory content when necessary, thereby improving the completeness of the answer and enhancing overall user satisfaction.

Workflow of Multi-Agents. According to the complexity of the query and the specific functions of these agents, the AI search system adopts three distinct team configurations to support different levels of reasoning and execution: (1) *Writer-Only Configuration*, (2) *Executor-Inclusive Configuration*, and (3) *Planner-Enhanced Configuration*. The operational details of each configuration are described as follows:

- **Writer-Only Configuration.** This configuration is suitable for the simple query that can be answered directly based on the system (LLM)'s reasoning capability and internalized

针对复杂查询, 该规划器能够使AI搜索系统通过一系列更简单、结构化的子任务来予以应对。其核心功能是从工具集(例如MCP平台)中选择合适的工具, 动态调整大语言模型的能力边界, 并生成用于描述全局任务计划的有向无环图。在该有向无环图中, 每个节点代表一个可调度的基本子任务, 每条边则体现了子任务之间的条件性依赖关系, 这些关系源自规划器的推理过程。此外, 当主控制器判定某些子任务存在执行错误或缺少关键数据时, 规划器会在主控制器的指导下对现有有向无环图进行重新配置。

- **执行器。**在任务执行阶段, 执行器承担着执行任务并评估其结果的核心职责, 以确保任务能够顺利完成且符合预定义的要求。每个子任务在执行时, 要么会预先分配好外部工具, 要么采用“无需工具”的配置, 完全依靠执行器自身的内置能力来处理。执行过程中, 执行器会根据任务的复杂程度反复调用指定的工具, 同时持续判断已累积的输出结果是否足以满足该子任务的目标要求。一旦评估结果判定要求已被满足, 当前输出结果就会被返回; 否则, 执行过程会继续迭代进行。如果所分配的工具出现无响应或故障情况, 系统会无缝切换至同一工具模块内功能等效的备用工具, 从而保障执行的连续性与稳定性。例如, 假设某个子任务需要通过网页搜索工具来获取相关知识, 执行器会依据当前子任务的内容生成子查询, 进而调用绑定的网页搜索工具进行多轮搜索。每轮搜索结束后, 执行器都会对检索到的文档进行评估, 以确定其中的信息是否足以支撑该子任务所需的事实或上下文要求。若不满足要求, 则会对这些子查询进行优化后再重新调用工具。一旦达到足够的覆盖度, 结果就会被确定并返回。通过这种集成方式, 执行器将任务执行与质量控制整合在同一机制中, 从而确保子任务能够稳健、准确且贴合语境地被执行。

- **撰写员。**该智能体负责整合此前所有已完成的子任务所收集到的信息与结果, 进而生成针对用户查询的最终答复。其目标是产出连贯、内容丰富且可能包含多视角观点的答复, 同时完成必要的筛选与歧义消除工作。在此过程中, 它会识别多个子任务输出之间的语义关系与逻辑结构, 并借助预定义的模板或自适应结构化策略来重新组织这些信息。如此一来, 撰写员便能生成条理清晰、组织有序且易于理解的答复。一旦子任务结果中存在冗余或语义不一致的情况, 撰写员会进行内容筛选与歧义消除, 以确保最终输出准确无误、不含误导性信息。除了回答核心问题外, 生成的答复还会在必要时补充相关背景信息与解释性内容, 从而提升答案的完整性, 进一步增强用户的满意度。

多智能体工作流程。根据查询的复杂程度以及这些智能体的具体功能, AI搜索系统采用了三种不同的团队配置来支持不同层次的推理与执行: (1) 仅含撰写员的配置, (2) 包含执行器的配置, 以及(3) 有规划器辅助的配置。各配置的具体运作方式如下:

- **仅写作器配置。**这种配置适用于那些能够直接凭借系统(大语言模型)的推理能力和内部知识库来解答的简单查询。

knowledge. In this case, the Master analyzes the query and determines that neither external tool invocation nor task decomposition is necessary. It then directly delegates the task to the Writer, who generates a response relying solely on its built-in generative capabilities. For example, given the query “*What is the name of Emperor Han-Wu?*”, the Master forwards the prompt to the Writer, who directly generates the response “*The name of Emperor Han-Wu is Liu Che.*”

- **Executor-Inclusive Configuration.** This configuration handles moderately complex queries that require factual information from external sources but do not involve multi-step reasoning or decomposition. The Master determines that the query can be resolved through single-step execution and assigns it directly to the Executor. Meanwhile, the AI search system uses the query semantics along with contextual cues derived from the team configuration to query the tool platform server (e.g. MCP server), retrieving a focused and semantically relevant subset of tools. The Executor selects the top-ranked tool from this subset and invokes it to perform the task. Upon obtaining the result, it evaluates whether the output meets the task requirements. Once execution is complete, the output is forwarded to the Writer, who synthesizes the final response. For instance, given the query “*Is Beijing’s weather good for going out today?*”, the Executor invokes the retrieved *weather query tool*, obtains real-time weather data, and after verifying its completeness, passes the result to the Writer. The response might be: “*Beijing’s weather today is sunny, with temperatures ranging from 12°C to 25°C. It is suitable for outdoor activities, although precautions against UV exposure are advised.*”
- **Planner-Enhanced Configuration.** This configuration is designed for complex queries that require multi-step reasoning and structured task execution. When high query complexity is detected, the Master delegates control to the Planner, initiating a planning-based workflow. In parallel, tool selection by the Planner is supported by querying the tool platform server (e.g. MCP server) for a semantically filtered candidate set, based on the input query and current execution context. The Planner decomposes the query into a sequence of atomic sub-tasks and organizes them into a DAG that captures their logical and execution dependencies. For each sub-task, a suitable tool is selected from the retrieved tool subset and explicitly bound to the corresponding node. The Executor then traverses the DAG layer by layer, invoking the bound tools and evaluating the intermediate results. Once all sub-tasks are completed, the aggregated outputs are passed to the Writer, who synthesizes them into a coherent and context-aware final response. For example, consider a complex query “*Who is elder, Emperor Han-Wu or Emperor Caesar, by how many years?*”. In this scenario, the Master delegates the query to the Planner, who is responsible for both planning and decomposing it into three specific sub-tasks, each associated with a specific tool from the candidate set:

- *Sub-Task 1: Search the birthdate of Emperor Han-Wu. Using the Web Search Tool.*
- *Sub-Task 2: Search the birthdate of Emperor Caesar. Using the Web Search Tool.*
- *Sub-Task 3: Calculate the difference between the two birthdates. Using the Programmer Tool.*

These sub-tasks are structured into a DAG according to their execution dependencies and are executed sequentially by the Executor. Once all results are collected, the Writer synthesizes them into a coherent and contextually accurate response: “*Emperor Wu of Han (156–87 BC) lived for approximately 69 years, whereas Julius Caesar (100–44 BC) lived for about 56 years. Emperor Wu of Han was therefore older by approximately 56 years.*”

Overall, these configurations illustrate the system’s adaptive execution flow, where different agent teams are engaged based on query complexity. By selecting suitable configurations that range from direct generation for simple tasks to tool-assisted multi-step planning for complex

此类情况下，Master会先对查询进行分析，判定无需调用外部工具或拆解任务，随后便直接将任务交由写作器处理。写作器仅依靠自身的内置生成能力来产出回答。例如，对于查询“汉武帝的名字是什么？”，Master会将相关提示词发送给写作器，后者便会直接生成回答：“汉武帝的名字是刘彻。”。

- **包含执行器的配置。**这种配置用于处理中等复杂程度的查询，这类查询需要从外部来源获取事实性信息，但不涉及多步骤推理或任务分解。主控制器会判定该查询可通过单步执行来解决，并将其直接分配给执行器。与此同时，AI搜索系统会利用查询的语义信息以及从团队配置中提取的上下文线索，向工具平台服务器（如MCP服务器）发起查询，从而获取一组具有针对性且语义相关的工具子集。执行器会从该子集中选出排名最高的工具并调用它来完成任务。获得结果后，执行器会评估输出内容是否满足任务要求。一旦执行完毕，输出结果会被传送给编写器，由其整合生成最终回复。例如，对于查询“今天北京的天气是否适合外出？”，执行器会调用所获取的天气查询工具，获取实时天气数据；在确认数据完整无误后，便将结果交给编写器。最终的回复可能是：“今天北京天气晴朗，气温在12°C到25°C之间。非常适合户外活动，不过建议做好防晒措施。”

- **规划器增强型配置。**该配置专为需要多步推理与结构化任务执行的复杂查询而设计。一旦检测到查询复杂度较高，Master节点便会把控制权移交给规划器，从而启动基于规划的流程。与此同时，规划器在选取工具时会先查询工具平台服务器（如MCP服务器），根据输入查询内容与当前执行上下文筛选出语义上匹配的候选工具集。规划器会将查询拆分成一系列原子级子任务，并把它们组织成能够体现其逻辑与执行依赖关系的有向无环图。针对每个子任务，都会从获取到的工具子集里挑选合适的工具，并将其明确绑定到对应的节点上。随后，执行器会逐层遍历该有向无环图，调用已绑定的工具并对中间结果进行评估。当所有子任务均完成后，汇总后的输出会被传递给撰写器，由其整合成连贯且贴合语境的最终回答。例如，对于复杂查询“汉武帝和凯撒大帝谁年纪更大？前者比后者大多少岁？”，Master节点会将查询交由规划器处理，规划器负责制定计划并将查询拆分成三个具体的子任务，每个子任务都对应候选工具集中的某一特定工具：

- 子任务 1：使用网页搜索工具查询汉武帝的出生日期。
- 子任务 2：使用网页搜索工具查询凯撒大帝的出生日期。
- 子任务 3：使用编程工具计算两位皇帝出生日期之间的差值。

这些子任务会根据其执行依赖关系被组织成有向无环图，并由执行器按顺序逐一处理。当所有结果收集完毕后，撰写器会将其整合成一段连贯且语境准确的回答：“汉武帝（公元前156年—前87年）大约活了69年，而尤利乌斯·凯撒（公元前100年—前44年）则约活了56年。因此，汉武帝比凯撒大帝年长约13年。”

总体而言，这些配置展现了系统的自适应执行流程：系统会根据查询的复杂程度调用不同的智能体团队。通过选取合适的配置——从简单任务的直接生成，到复杂查询所需的工具辅助型多步骤规划——系统能够在各类搜索场景中兼顾效率与可扩展性。

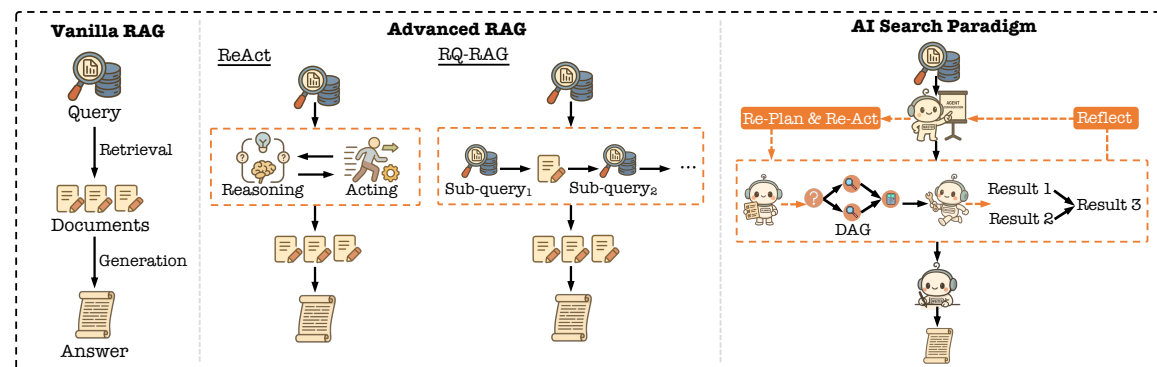


Figure 2 | Comparison of RAG frameworks. **Left:** Vanilla RAG conducts one-shot retrieval followed by direct answer generation. **Middle:** Advanced RAG methods, such as ReAct and RQ-RAG, involve reasoning-action cycles or sequential sub-query execution. **Right:** AI search paradigm introduces a multi-agent system wherein *Master* guides *Planner* to formulate a plan based on the input query, while also continuously evaluating the execution status and completeness of sub-task results, and performing reflection and re-planning when necessary. *Planner* is responsible for constructing a DAG of sub-tasks and dynamically selecting the appropriate tools, thereby enabling structured and adaptive multi-step execution. *Executor* executes the specific sub-tasks using these tools, and finally, *Writer* generates the final answer.

queries, the system ensures both efficiency and scalability across diverse search scenarios.

In the remainder sections, we provide a detailed exploration of the implementation algorithms and optimization strategies applied to these agents.

3. Task Planner

As the core reasoning component in the AI search system, the Planner is responsible for decomposing complex queries into structured sub-tasks and orchestrating execution through appropriate tools. Unlike conventional systems that rely on static retrieval and fixed response generation, the Planner enables dynamic task planning, effective management of multiple tools, and adaptive decision-making. In this section, we elaborate on its motivation, functional design, and optimization strategies within the broader context of AI Search paradigm.

3.1. Background

When addressing complex queries that necessitate multi-step sub-task execution and the coordinated invocation of multiple tools, traditional RAG systems often fall short in effective task planning and dynamic multi-tool coordination. Consequently, their reliance on single-step task execution or fixed-tool utilization frequently results in incomplete information retrieval or inaccurate responses. We revisit the motivating example from a large-scale real-world search engine in Section 2 to underscore the limitations of traditional systems and motivate the design of the Planner: “Who is elder, Emperor Han-Wu or Emperor Caesar, by how many years?”. To resolve this query, it is first decomposed into three specific sub-tasks, each assigned to a dedicated tool. Then, these sub-tasks are organized into a DAG that reflects their logical execution order and subsequently executed in sequence. Throughout this section, we will employ this example to provide a comparative analysis of how recent RAG systems and our proposed search system approach the same problem.

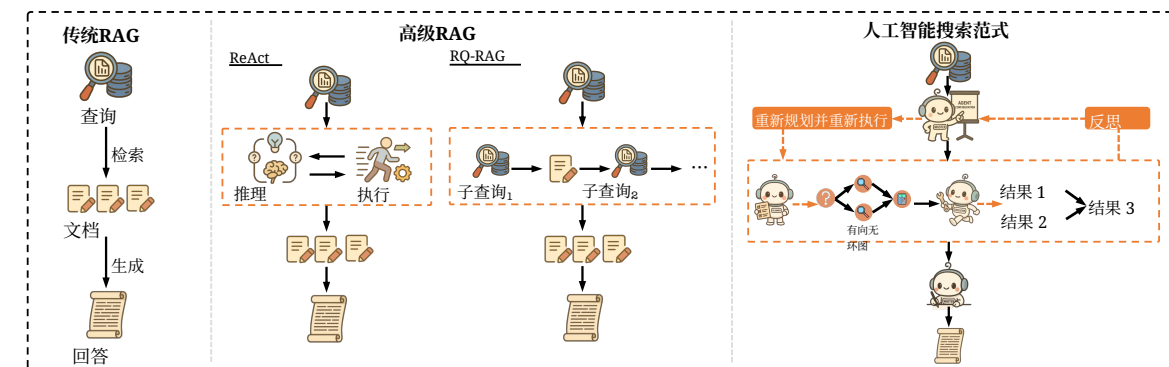


图 2 | : RAG框架对比。左侧：传统RAG仅执行单次检索，随后直接生成答案。中间：先进的RAG方法，如ReAct和RQ-RAG，会涉及推理-行动循环或顺序子查询执行。右侧：AI搜索范式引入了多智能体系统，其中主智能体引导规划器根据输入查询制定计划，同时持续评估子任务的执行状态与完成情况；必要时还会进行反思并重新规划。规划器负责构建子任务的有向无环图，并动态选择合适的工具，从而实现结构化且自适应的多步骤执行。执行器利用这些工具完成具体的子任务，最终由撰写器生成最终答案。

在各类查询中，该系统都能确保高效性与可扩展性，以适应多样化的搜索场景。

在后续章节中，我们将详细探讨应用于这些智能体的实现算法与优化策略。

3. 任务规划器

作为AI搜索系统中的核心推理组件，规划器负责将复杂查询拆解为结构化的子任务，并通过合适的工具协调执行。与传统依赖静态检索和固定响应生成的系统不同，规划器能够实现动态任务规划、高效管理多种工具以及自适应决策。在本节中，我们将结合AI搜索范式的宏观背景，详细阐述其设计动机、功能架构及优化策略。

3.1. 背景

在处理需要多步骤子任务执行以及多个工具协同调用的复杂查询时，传统的RAG系统往往在有效的任务规划与动态多工具协调方面表现欠佳。正因如此，它们依赖单步任务执行或固定工具使用的方式，常常会导致信息检索不完整或回答不准确。我们在第2节中重新引用了某大型搜索引擎中的实际案例，以此说明传统系统的局限性，并借此引出Planner的设计动机：“汉武帝和凯撒大帝谁年纪更大？大多少岁？”。为解决这一查询，首先将其拆解为三个具体的子任务，每个子任务交由专门的工具处理；随后将这些子任务组织成反映其逻辑执行顺序的有向无环图，再按顺序依次执行。在本节中，我们将借助该示例，对比分析近期出现的RAG系统与我们所提出的搜索系统在处理同一问题时的差异。

Recent RAG Workflow. A conventional RAG approach begins by submitting the query to a retriever to obtain relevant documents, as illustrated in the left panel of Fig. 2. These documents, along with the query, serve as a prompt for an LLM to generate an answer. In practice, this method has yielded incorrect and incomplete results (*i.e.*, returning “Emperor Caesar” in our case), due to incomplete retrieval and the failure to capture the necessary birth dates for Emperor Han-Wu and Emperor Caesar. This shortcoming primarily arises from a retrieval process that lacks prior reasoning, as the query only mentions the pertinent names without specifying the need for birth dates, leading the language model to generate inferences without appropriate factual support.

Moreover, advanced RAG approaches, as illustrated in the middle part of Fig. 2 (*i.e.*, ReAct (Yao et al., 2023a)), introduce a structured loop that alternates between *Thought*, *Action*, and *Observation* phases. In such frameworks, the model plans the subsequent reasoning step, retrieves relevant information, and iteratively incorporates the results. When applied to the current case, a ReAct-like agent would first consider retrieving the birth dates, then carry out document retrievals for both emperors, and update its internal state accordingly. However, despite these enhancements, ReAct’s reliance solely on in-context memory precludes genuine external tool invocation; consequently, even if the agent correctly determines which emperor is older, it cannot precisely compute the age difference due to the absence of a dedicated calculator tool and structured numerical reasoning. Furthermore, the iterative context expansion inherent in ReAct complicates the maintenance of entity alignment and consistent task planning, thereby increasing the risk of error propagation. Similarly, RQ-RAG (Chan et al., 2024) seeks to enhance RAG performance by explicitly decomposing complex queries into sequential sub-queries, solving them step-by-step while retaining intermediate results. For instance, a query decomposition-based agent would first retrieve Emperor Han-Wu’s birth date, then that of Emperor Caesar, and subsequently deduce their age difference. Nonetheless, despite the improved decomposition capabilities, RQ-RAG still operates within a single retrieval-and-generation framework that lacks dynamic tool utilization. As a result, although it may successfully obtain individual birth dates, it is unable to perform the subsequent computational reasoning necessary to determine the precise age difference, leaving the answer incomplete. In summary, while ReAct and RQ-RAG represent significant advancements over traditional RAG architectures by incorporating iterative reasoning and query decomposition, their inability to dynamically invoke external tools and to plan tasks restricts their effectiveness in accurately resolving complex, multi-step queries.

Necessity of Task Planner. To overcome these limitations, a dedicated Planner is indispensable for complex query resolution. Unlike previous RAG systems that rely exclusively on retrieval or in-context decomposition, a Planner explicitly decomposes queries into fine-grained sub-tasks, determines their logical dependencies via a DAG, and dynamically selects appropriate tools for task execution beyond simple retrieval, as illustrated in the right part of Fig. 2. Moreover, the planner is capable of re-planning; if any intermediate results deviate from the expected objectives, the Planner, under the guidance of the Master, adjusts the task plan accordingly. These capabilities fundamentally expand the scope of retrieval-augmented systems, transforming passive “retrieve-then-generate” pipelines into “reason, plan, execute and re-plan”, a proactive AI search system. Concretely, the planner must decide what can be planned, which tools should be invoked, and how to optimize its own behaviour over time.

近期RAG工作流程。传统的RAG方法首先将查询提交给检索器以获取相关文档，如图2左侧面板所示。这些文档会与查询一起作为提示词，输入给大语言模型以生成答案。实践中，这种方法往往会产生错误且不完整的结果（也就是说，在我们的案例中返回的是“凯撒大帝”），原因在于检索过程不全面，且未能获取汉武帝与凯撒大帝的出生日期。这一缺陷主要源于缺乏事先推理的检索过程：查询中仅提及了相关人名，并未指明需要出生日期，从而导致语言模型在缺乏恰当事实支撑的情况下便做出了推断。

此外，先进的RAG方法——如图2中间部分所示（即，ReAct（Yao等人，2023a））——引入了一种结构化的循环机制，该循环会在思考、行动和观察阶段之间交替进行。在此类框架中，模型会规划后续的推理步骤、检索相关信息，并反复整合这些结果。若将其应用于当前案例，类似ReAct的智能体首先会考虑获取两位皇帝的出生日期，随后针对他们分别进行文档检索，并据此更新其内部状态。然而，尽管有这些改进，ReAct仅依赖上下文记忆的特性使其无法真正调用外部工具；因此，即便智能体能正确判断哪位皇帝年龄更大，由于缺乏专用的计算器工具以及结构化的数值推理能力，它也无法精确算出年龄差。此外，ReAct所固有的迭代式上下文扩展机制会给实体对齐和任务规划的维持带来困难，进而增加错误传播的风险。同样，RQ-RAG（Chan等人，2024）试图通过将复杂查询显式拆解为一系列子查询、逐步求解并保留中间结果的方式来提升RAG的性能。例如，基于查询分解的智能体会先获取汉武帝的出生日期，再获取凯撒大帝的出生日期，随后推导出他们的年龄差。不过，尽管其分解能力有所提升，RQ-RAG仍运行在单一的检索与生成框架内，缺乏动态工具利用能力。因此，尽管它能成功获取各自的出生日期，却无法进行后续必要的计算推理以确定精确的年龄差，导致答案不完整。总而言之，虽然ReAct和RQ-RAG通过引入迭代推理和查询分解机制，较传统的RAG架构有了显著进步，但它们无法动态调用外部工具并进行任务规划，这限制了它们在准确解决复杂的多步骤查询时的有效性。

任务规划器的必要性。为克服这些局限，针对复杂查询解析而言，专门的规划器不可或缺。与以往仅依赖检索或上下文分解的RAG系统不同，规划器会将查询明确拆解为细粒度的子任务，通过有向无环图确定它们之间的逻辑依赖关系，并动态选择合适的工具来执行任务——而不仅限于简单的检索操作，如图2右侧所示。此外，规划器还具备重新规划的能力；一旦任何中间结果偏离预期目标，规划器便会在Master的指导下相应调整任务计划。这些能力从根本上拓展了检索增强系统的应用范围，将被动的“先检索后生成”流程转变为“推理、规划、执行并重新规划”，即一种主动式AI搜索系统。具体而言，规划器必须决定哪些内容可被规划、应调用哪些工具，以及如何随时间优化自身行为。

3.2. Task Universe and the MCP Abstraction

Early tool-augmented LLM systems relied on vendor-specific “function-calling” JSON schemas such as those introduced by OpenAI (OpenAI, 2025a,b) and rapidly copied by many frameworks (LangChain, 2024). While simple, these ad-hoc contracts were *coupled* to one provider, lacked machine-typed guarantees, and made it impossible for independent agents to share tools or reason about cost, latency, or security across organizational boundaries. For AI Search whose planner must orchestrate heterogeneous knowledge look-ups, computations, and transformations in a single multi-step plan, such fragmentation is a major bottleneck.

Model-Context Protocol. MCP addresses this fragmentation by specifying a vendor-neutral, HTTP+JSON-RPC interface through which *Servers* expose tools and data, while *Clients* (i.e., LLMs or agents) discover, invoke, and monitor those tools in a secure, typed fashion (Anthropic, 2024; Schmid, 2025). Specifically, the protocol bundles four ingredients: (1) a *manifest* that advertises each endpoint’s name, semantic role, cost, and latency bounds; (2) machine-readable *input/output schemas* that ground the LLM’s function-calling tokens; (3) a *capability handshake* for tool discovery, and (4) an *execution contract* that guarantees idempotent, auditable calls.

3.3. Dynamic Capability Boundary

Given a user’s input query and MCP servers, we define the combination of the LLM for the Planner and the tool set as *Capability Boundary*. Specifically, the capability boundary encompasses the reasoning capability and internalized knowledge of LLMs, as well as tools, such as web search, calculators, and programmers. Typically, in the absence of these tools, an LLM merely follows the traditional “text input-and-text output” paradigm inherent to conventional language model-based search paradigms; however, the integration of these tools, which enables functionalities such as web search, computation, and programming, substantially enhances the AI search process. With the boundary defined, the AI search system can generate plans tailored to the input query. Specifically, the Planner constructs a DAG in which each node represents an individual sub-task (i.e., a single invocation of a tool), and the edges capture the dependency relationships between two nodes. This framework ensures that any task is executed only after all its prerequisite tasks have been completed.

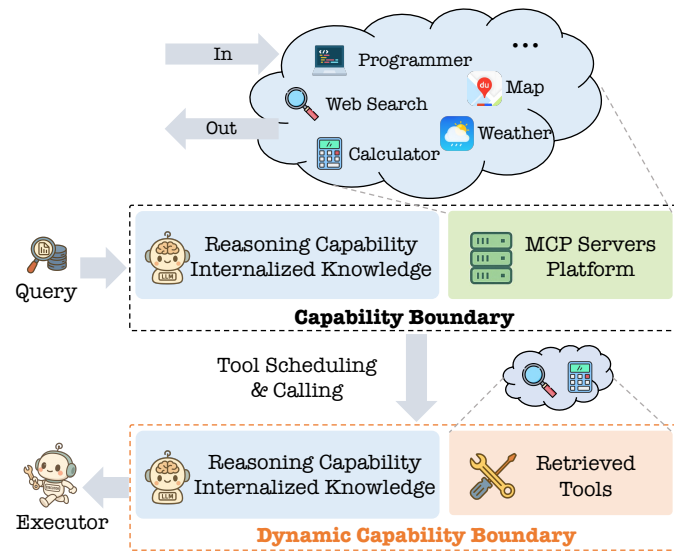


Figure 3 | An illustration of the dynamic capability boundary of our search paradigm.

However, a notable challenge arises as the portfolio of available tool APIs grows geometrically over time, eventually exceeding the representational capacity of a static capability boundary. To address the above challenge, AI Search paradigm introduces a novel concept termed the *Dynamic Capability Boundary* within the task planner phase. As shown in Fig. 3, the AI search

3.2. 任务宇宙与MCP抽象

早期的工具增强型LLM系统依赖于厂商特定的“函数调用”JSON模式，例如OpenAI所推出的相关模式（OpenAI, 2025a,b），随后许多框架也迅速效仿之（LangChain, 2024）。尽管这类模式较为简单，但它们与单一提供商绑定，缺乏机器可识别的规范性保障，以至于独立代理无法跨组织边界共享工具，也无法评估相关成本、延迟或安全性问题。对于AI搜索而言，其规划器需要在单一的多步骤计划中协调异构的知识查询、计算及转换操作，此类碎片化现象无疑构成了重大瓶颈。

模型上下文协议。 MCP通过定义一种与厂商无关的HTTP+JSON-RPC接口来解决这种碎片化问题。借助该接口，服务器可以暴露工具和数据，而客户端（即大语言模型或智能体）则能以安全且类型化的方式发现、调用并监控这些工具（Anthropic, 2024; Schmid, 2025）。具体而言，该协议包含四个核心要素：(1) 一个清单，用于说明每个端点的名称、语义角色、成本及延迟限制；(2) 机器可读的输入输出模式，用以支撑大语言模型的函数调用功能；(3) 用于工具发现的能力握手机制；以及(4) 能够确保调用具有幂等性和可审计性的执行契约。

3.3. 动态能力边界

给定用户的输入查询以及MCP服务器，我们将规划器所使用的LLM与工具集的组合定义为能力边界。具体而言，该能力边界涵盖了LLM的推理能力与内化知识，同时也包含各类工具，例如：

网页搜索、计算器以及编程工具。通常而言，在缺乏这些工具的情况下，LLM仅遵循传统基于语言模型的搜索范式中所固有的“文本输入—文本输出”模式；然而，一旦集成这些能够实现网页搜索、计算及编程等功能的工具，AI搜索过程便会得到显著增强。明确界定这一边界后，AI搜索系统便能针对输入查询生成定制化的计划。具体而言，规划器会构建一个有向无环图，其中每个节点代表一个子任务（即对某一工具的单次调用），而边则用于描述两个节点之间的依赖关系。这种

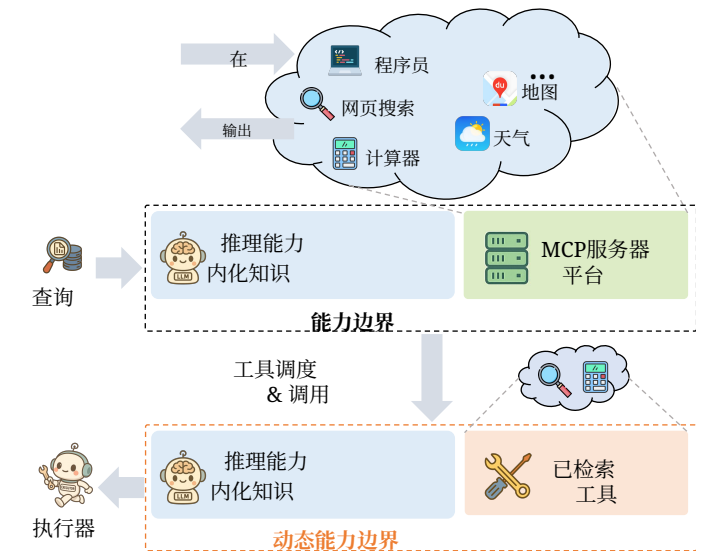


图 3 |：我们的搜索范式下动态能力边界的示意图。

该框架确保任何任务只有在所有前置任务全部完成后才会被执行。

然而，随着可用工具 API 的数量随时间呈几何级数增长，最终会超出静态能力边界的表征能力，这一难题不容忽视。为应对上述挑战，“面向AI的搜索范式”在任务规划阶段引入了一种名为动态能力边界的新概念。如图3所示，AI搜索

system leverages an LLM to process the input query and, within a short time, select a potential subset of tools. Given the selected tool subset, the AI search system combines it with the reasoning ability and internalized knowledge of the LLM to constitute the new dynamic capability boundary. Based on our extensive practical experience, the dynamic boundary typically consists of a relatively small set of tools, around a dozen tools, which proves sufficient to carry out task planning.

3.3.1. Refinement of Tool API Documents

Considering the critical importance of precise tool description within AI Search paradigm, employing tool learning techniques to refine these documents represents an effective approach (Mialon et al., 2023; Qin et al., 2023b). Specifically, tool learning has emerged as a powerful approach for enhancing the ability of LLMs to tackle complex, real-world tasks (Qu et al., 2025a; Schick et al., 2024). By interacting with tools such as calculators, search engines, APIs, and web environments, LLMs could overcome key limitations arising from their fixed pretraining data and inherently passive text-in/text-out interface (M. Bran et al., 2024; Nakano et al., 2021; Qin et al., 2023a). This integration allows LLMs to acquire real-time information, perform dynamic actions, and extend their utility beyond traditional language modeling (Wang et al., 2024; Zhuang et al., 2024). To facilitate such interactions, LLMs are typically equipped with tool documentation that serves as contextual input, describing each tool’s functionalities, usage methods, and applicable scenarios (Shen et al., 2024; Song et al., 2023; Xu et al., 2023). The effectiveness of LLMs in leveraging external tools largely depends on the clarity, structure, and accuracy of the accompanying API documentation. While traditional tool documents are generally written for human developers, they often fall short of meeting the specific needs of LLMs by containing ambiguities, redundancies, or missing information, thereby impeding the accurate interpretation of tool functionalities and task execution (Chemero, 2023; Yuan et al., 2024). Given these limitations, a key question arises: How can we construct tool documentation that is better aligned with the needs of the Planner?

To enhance the quality of tool documentation, the AI search utilizes an iterative refinement method *DRAFT*, which leverages the interactions between LLMs and external tools, as well as the feedback generated during these interactions, to progressively optimize tool documents (Qu et al., 2025b). As illustrated in Fig. 4, *DRAFT* involves three iterative phases: *Experience Gathering*, *Learning from Experience*, and *Documentation Rewriting*. Specifically, *DRAFT* first systematically simulates diverse use-cases, encompassing typical interactions, edge cases, error scenarios, and parameter limits. Such exhaustive exploratory interactions uncover gaps and inaccuracies in existing tool descriptions. Subsequently, *DRAFT* analyzes collected interaction data to identify discrepancies and ambiguities. Insights derived from practical tool interactions ensure that re-

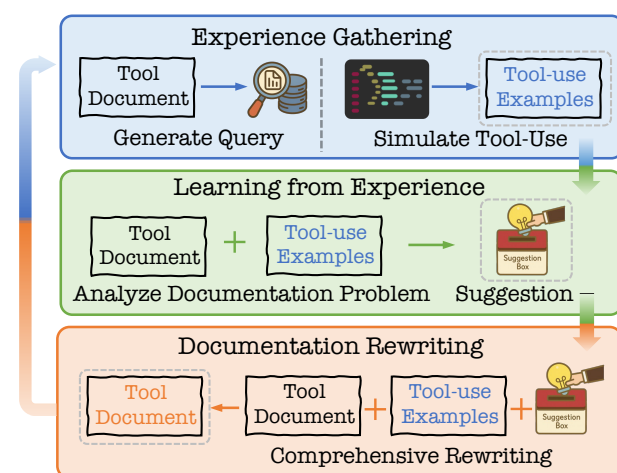


Figure 4 | The pipeline of our proposed self-driven iterative refinement framework, which leverages the interactions between LLMs and external tools, as well as the feedback generated during these interactions, to progressively optimize tool documents

该系统借助大语言模型来处理输入查询，并在短时间内筛选出潜在的工具子集。选定该工具子集后，AI搜索系统会将其与大语言模型的推理能力及内置知识相结合，从而构成新的动态能力边界。根据我们丰富的实践经验，这种动态边界通常由十几种工具组成，这一规模的工具集足以支撑任务规划工作。

3.3.1. 工具API文档的优化

鉴于在AI搜索范式中精准描述工具具有至关重要的意义，采用工具学习技术来优化这些文档是一种行之有效的方法（Mialon等人，2023年；Qin等人，2023b年）。具体而言，工具学习已成为提升大语言模型处理复杂现实任务能力的有力手段（Qu等人，2025a年；Schick等人，2024年）。通过与计算器、搜索引擎、API及网络环境等工具交互，大语言模型能够克服其固定预训练数据以及固有的文本输入/输出接口所带来的局限（M. Bran等人，2024年；Nakano等人，2021年；Qin等人，2023a年）。这种集成使得大语言模型能够获取实时信息、执行动态操作，进而将其应用范围拓展至传统的语言建模之外（Wang等人，2024年；Zhuang等人，2024年）。为了促进此类交互，通常会为大语言模型配备工具文档作为上下文输入，用以说明各工具的功能、使用方法及适用场景（Shen等人，2024年；Song等人，2023年；Xu等人，2023年）。大语言模型利用外部工具的效果在很大程度上取决于所附带的API文档的清晰度、结构合理性及准确性。虽然传统的工具文档主要是为开发人员编写的，但它们往往存在歧义、冗余或信息缺失等问题，难以满足大语言模型的特定需求，进而妨碍对工具功能的准确理解及任务执行（Chemero，2023年；Yuan等人，2024年）。鉴于上述局限，一个关键问题随之浮现：我们该如何构建更贴合规划器需求的工具文档？

为了提升工具文档的质量，AI搜索采用了一种迭代优化方法*DRAFT*。该方法借助大语言模型与外部工具之间的交互，以及交互过程中产生的反馈，来逐步优化工具文档（Qu等人，2025b）。如图4所示，*DRAFT*包含三个迭代阶段：经验收集、从经验中学习以及文档重写。具体而言，*DRAFT*首先会系统地模拟各类使用场景，涵盖常规交互、极端情况、错误场景以及参数限制等。这种全面的交互探索能够发现现有工具描述中存在的漏洞与不准确之处。随后，*DRAFT*会对收集到的交互数据进行分析，以识别其中的差异.....

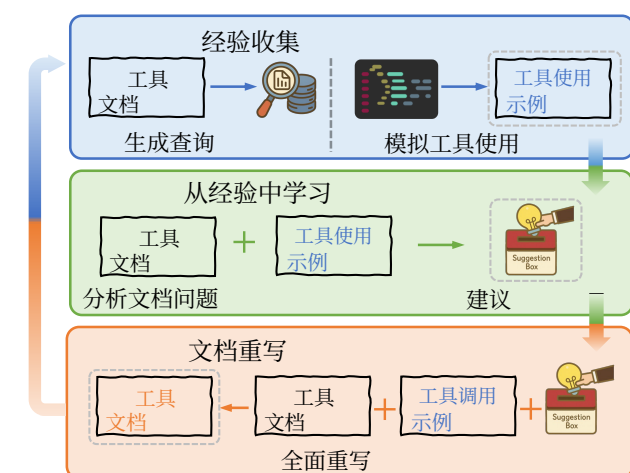


图 4 | 展示了我们提出的自驱动迭代优化框架的流程，该框架利用大语言模型与外部工具之间的交互，以及交互过程中产生的反馈，来逐步完善工具文档。

差异与歧义。从实际工具交互中获得的见解能够确保后续优化紧密贴合真实使用场景。

finements align closely with actual usage contexts. This analytical process yields targeted recommendations aimed at correcting inaccuracies, clarifying ambiguities, and eliminating redundancy. Eventually, DRAFT integrates these recommendations, synthesizing refined tool descriptions specifically optimized for effective interpretation by LLMs. The method incorporates strategies that promote diverse exploratory coverage and adaptive termination criteria to prevent excessive iterations, ensuring concise yet comprehensive documentation. Next, we will describe the algorithm details of our proposed method as follows.

Experience Gathering. In the first phase, DRAFT leverages an explorer to simulate plausible scenarios for tool use, similar to how humans experiment with unfamiliar tools when manuals are unclear. Specifically, at the i -th iteration, $\mathcal{M}_E$ generates an exploration instance e_i based on the current documentation t_{i-1} , the suggested direction d_{i-1} from the rewriter $\mathcal{M}_R$, and prior history $\mathcal{H}_i = \{(e_j, r_j) \mid j < i\}$. The learning process could be formulated as

$$e_i = \mathcal{M}_E(t_{i-1}, d_{i-1}, \mathcal{H}_i), \quad (1)$$

where e_i comprises a query e_i^q and its parameters e_i^p . The tool is then invoked to obtain the result returned by the tool, denoted as r_i .

Given the complexity of tool usage, such as diverse parameter ranges and potential errors, it is essential to ensure exploration diversity. To this end, DRAFT implements a *diversity-promoting strategy* to optimize this process. Specifically, the explorer will calculate the cosine similarity between the newly generated query e_i^q and previous queries e_j^q ($j < i$). A new instance is accepted only if:

$$\max_{j < i} \text{sim}(\mathbf{e}_i^q, \mathbf{e}_j^q) < \phi, \quad (2)$$

where ϕ is a predefined similarity threshold. If the constraint is violated, the explorer discards the instance and reflects on the redundancy (Shinn et al., 2024), iteratively refining the generated query until a sufficiently novel query is generated.

The diversity-promoting exploration strategy enables the explorer to engage with not only common tool functionalities but also less typical and edge-case scenarios. Such comprehensive interaction coverage helps reveal hidden flaws in the documentation, such as ambiguous instructions, incomplete descriptions, or misleading usage guidance. The exploration outcomes provide critical experiential data that form the foundation for later stages of analysis and document revision.

Learning from Experience. The second phase builds on the experiential data to refine the tool documentation. Here, DRAFT introduces an analyzer $\mathcal{M}_A$, which analyzes the exploration instance e_i , tool response r_i , current documentation t_{i-1} , and revision history $\mathcal{T}_i = \{t_j \mid j < i\}$ to generate targeted revision suggestions s_i as

$$s_i = \mathcal{M}_A(t_{i-1}, e_i, r_i, \mathcal{T}_i). \quad (3)$$

To ensure relevance and quality, the analyzer is guided by criteria such as alignment with tool behavior, clarity, and informativeness without redundancy. By leveraging the revision history $\mathcal{T}_i$, it avoids repetitive suggestions and focuses on unresolved documentation gaps.

Documentation Rewriting. The final phase centers on document rewriting, aiming to improve the clarity, precision, and interpretability of tool documentation for LLMs. This phase not only incorporates feedback from prior exploration and revision but also proactively informs the next cycle of tool interaction. To achieve this, DRAFT introduces a rewriting module, the rewriter $\mathcal{M}_R$, which consolidates multiple sources of information: (1) the exploration instances

这些优化措施与实际使用情境高度契合。这一分析过程会生成针对性的建议，旨在纠正不准确之处、消除歧义并减少冗余内容。最终，DRAFT会整合这些建议，生成经过精心优化的工具描述，以便让大语言模型能够高效解读。该方法采用了能够促进多样化探索覆盖、设定自适应终止条件的策略，从而避免不必要的反复迭代，确保生成的文档既简洁又全面。接下来，我们将详细介绍所提出方法的算法细节。

经验收集阶段。在第一阶段，DRAFT利用探索器来模拟工具使用的合理场景，这与人们在手册不清晰时对陌生工具进行尝试的方式类似。具体而言，在第 i 次迭代时， $\mathcal{M}_E$ 会基于当前的文档 t_{i-1} 、重写器 $\mathcal{M}_R$ 给出的建议方向 d_{i-1} 以及历史记录 $\mathcal{H}_i = \{(e_j, r_j) \mid j < i\}$ ，生成一个探索实例 e_i 。整个学习过程可以表述为：

$$e_i = \mathcal{M}_E(t_{i-1}, d_{i-1}, \mathcal{H}_i), \quad (1)$$

其中， e_i 包含一个查询 e_i^q 及其对应的参数 e_i^p 。随后调用该工具以获取其返回的结果，记为 r_i 。

鉴于工具使用的复杂性——比如参数范围多样且容易出现错误——确保探索过程的多样性至关重要。为此，DRAFT采用了促进多样性的策略来优化这一过程。具体来说，探索器会计算新生成的查询 e_i^q 与以往查询 e_j^q ($j < i$) 之间的余弦相似度。只有当满足以下条件时，才会接受该新实例：

$$\max_{j < i} \text{sim}(\mathbf{e}_i^q, \mathbf{e}_j^q) < \phi, \quad (2)$$

这里 ϕ 是预先设定的相似度阈值。一旦违反这一约束，探索器便会丢弃该实例，并对冗余性进行反思（Shinn等人，2024），反复优化生成的查询，直至得到足够新颖的查询为止。

这种促进多样性的探索策略能够让探索者不仅接触常见的工具功能，还能涉足那些较为罕见或处于边缘情况的场景。这种全面的交互覆盖有助于发现文档中隐藏的缺陷，比如表述模糊的说明、不完整的描述或具有误导性的使用指南。探索所得的结果提供了至关重要的经验性数据，为后续的分析与文档修订工作奠定了坚实基础。

从经验中学习。第二阶段以这些经验性数据为基础，对工具文档进行优化。在此环节，DRAFT引入了分析器 $\mathcal{M}_A$ ，该分析器会对探索实例 e_i 、工具响应 r_i 、当前文档 t_{i-1} 以及修订历史 $\mathcal{T}_i = \{t_j \mid j < i\}$ 进行分析，从而生成有针对性的修订建议 s_i 。

$$s_i = \mathcal{M}_A(t_{i-1}, e_i, r_i, \mathcal{T}_i). \quad (3)$$

为确保建议的相关性与质量，分析器的运作遵循一系列准则，包括与工具行为保持一致、表述清晰、信息丰富且无冗余。借助修订历史 $\mathcal{T}_i$ ，分析器能够避免提出重复的建议，专注于填补文档中尚未解决的空白。

文档重写。最终阶段聚焦于文档重写，旨在提升面向大语言模型的工具文档的清晰度、精确性与可解释性。该阶段不仅会采纳此前探索与修订环节所反馈的意见，还会为下一轮工具交互提供前瞻性指导。为实现这一目标，DRAFT引入了重写模块——rewriter $\mathcal{M}_R$ ，它能够整合多源信息：(1)由explorer $\mathcal{M}_E$ 生成的探索实例及对应的工具输出；(2)由analyzer $\mathcal{M}_A$ 提出的修订建议；

e_i and corresponding tool outputs r_i generated by the explorer M_E ; (2) the revision suggestions s_i provided by the analyzer M_A ; and (3) the full rewrite history $\mathcal{T}_i$, which captures all previous versions of the documentation up to the current iteration. By considering the historical evolution of the documentation, the model can better preserve useful edits and avoid regressions. Formally, the rewriting process could be defined as

$$d_i, t_i = \mathcal{M}_R(t_{i-1}, e_i, r_i, s_i, \mathcal{T}_i), \quad (4)$$

where t_i represents the revised documentation at iteration i , and d_i outlines recommended directions for further exploration. This closed-loop mechanism ensures that each iteration of rewriting is both context-aware and forward-looking, leading to increasingly robust and LLM-friendly documentation over time.

Recognizing that tool complexity varies, we adopt a *tool-adaptive termination mechanism* to decide when to stop iterations. We measure the change Δ between consecutive versions using a combination of the word-match metric BLEU score (Papineni et al., 2002) and the semantic-match metric cosine similarity, formulated as

$$\Delta = \frac{\text{sim}(\mathbf{e}_i^t, \mathbf{e}_{i-1}^t) + \text{BLEU}(t_i, t_{i-1})}{2}. \quad (5)$$

If Δ exceeds a threshold τ , we consider the documentation sufficiently refined and halt further revisions. This tool-adaptive termination mechanism improves efficiency by avoiding redundant updates and ensures that the final documentation is both semantically and structurally aligned with LLM understanding.

DRAFT offers several key benefits that render it both practical and scalable for maintaining tool documentation. Fundamentally, DRAFT reimagines documentation refinement as a fully automated, feedback-driven process, thereby reducing the reliance on labor-intensive manual revisions. By leveraging LLMs’ iterative interactions with tools, it continuously identifies and rectifies discrepancies between the documented specifications and the LLMs’ actual behavior or interpretation. This iterative learning loop not only enhances clarity and usability but also guarantees that the documentation evolves concurrently with changes in the underlying tools. Moreover, as DRAFT operates through natural language prompts and responses, it provides strong interpretability and facilitates human review, auditing, and feedback, thereby ensuring an effective balance between automation and human-in-the-loop flexibility.

In this way, existing tool documentation can be systematically aligned with the specific requirements of the Planner, thereby laying a solid foundation for the subsequent planning process.

3.3.2. Tool Clustering in MCP

Effectively leveraging external tools in the AI search system requires a clear and fine-grained understanding of their functional characteristics. Existing categorizations of tool APIs, particularly in current MCP platforms, are often overly broad and fail to reflect task-specific functionalities. As a consequence, tools with distinct purposes may be grouped together, while those serving similar functions remain unlinked. This imprecise classification increases the complexity of task execution and reduces system reliability. In practical scenarios, when a tool fails during execution, the system often lacks access to alternative tools with equivalent capabilities. This absence of functional redundancy leads to higher error rates and diminished robustness, particularly under conditions of task complexity or environmental uncertainty.

e_i 以及由explorer M_E 生成的对应工具输出 r_i ；(2)analyzer M_A 给出的修订建议 s_i ；(3)完整的重写历史记录 $\mathcal{T}_i$ ，它涵盖了截至当前迭代的所有文档版本。通过考量文档的历史演变过程，模型能够更好地保留有效的编辑内容并避免性能倒退。从形式上看，重写过程可定义为：

$$d_i, t_i = \mathcal{M}_R(t_{i-1}, e_i, r_i, s_i, \mathcal{T}_i), \quad (4)$$

t_i 代表第 i 次迭代中修订后的文档，而 d_i 则指明了进一步探索的推荐方向。这种闭环机制确保了每一次重写都既能够契合上下文语境，又具备前瞻性，从而随着时间的推移生成愈发稳健且更易于大语言模型理解的文档。

考虑到工具复杂度的差异，我们采用了适配工具的终止机制来决定何时停止迭代。我们通过BLEU分数（Papineni等人，2002年提出的一种词匹配指标）与余弦相似度这一语义匹配指标相结合的方式衡量连续版本间的差异 Δ ，具体公式如下：

$$\Delta = \frac{\text{sim}(\mathbf{e}_i^t, \mathbf{e}_{i-1}^t) + \text{BLEU}(t_i, t_{i-1})}{2}. \quad (5)$$

一旦 Δ 超出阈值 τ ，我们就认为文档已足够完善，便会停止进一步的修订。这种适配工具的终止机制能够避免冗余更新，从而提升效率；同时还能确保最终生成的文档在语义与结构上均与大语言模型的理解方式相契合。

DRAFT具备多项显著优势，使其在维护工具文档方面既实用又具备可扩展性。从根本上说，DRAFT将文档优化重新定义为一个完全自动化、以反馈为驱动的过程，从而减少了对耗费大量人力的人工修订的依赖。通过利用大语言模型与工具之间的迭代交互，它能持续识别并纠正文档说明与大语言模型的实际行为或理解之间的差异。这种迭代学习循环不仅提升了文档的清晰度和可用性，还保证了文档能随底层工具的变化同步演进。此外，由于DRAFT是通过自然语言提示与响应来运行的，因此具备很强的可解释性，便于人工审核、审计与反馈，进而实现了自动化与人工干预之间的有效平衡。

通过这种方式，现有的工具文档能够与规划器的具体需求实现系统性匹配，从而为后续的规划流程奠定坚实基础。

3.3.2. MCP中的工具聚类

要在AI搜索系统中有效利用外部工具，必须对其功能特性有清晰且细致的理解。现有的工具API分类方式，尤其是当前MCP平台中的分类，往往过于宽泛，无法准确反映特定任务所需的功能。结果就是，用途迥异的工具被归为一类，而功能相近的工具却未能关联起来。这种不精确的分类增加了任务执行的复杂度，降低了系统的可靠性。在实际场景中，一旦某个工具在运行过程中出现故障，系统往往无法找到具备同等能力的替代工具。这种功能冗余的缺失会导致更高的错误率，并削弱系统的鲁棒性，尤其是在任务复杂或环境不确定的情况下。

To address these limitations, the AI search system concentrates on the automated categorization of tool APIs based on detailed functional similarities. Given the refined tool API document D_{tool_i} introduced in Section 3.3.1, the AI search system leverages LLMs to generate concise functional descriptions $desc_{tool_i}$ of tool APIs. These descriptions are subsequently transformed into high-dimensional semantic embeddings e_{tool_i} using a text embedding model $\mathcal{M}_e$, which facilitates the identification of functionally coherent groups through rigorous clustering analyses. The above process can be formulated as

$$desc_{tool_i} = \mathcal{LLM}(D_{tool_i}), \quad (6)$$

$$e_{tool_i} = \mathcal{M}_e(desc_{tool_i}). \quad (7)$$

To organize the tool embeddings into functionally coherent groups, the AI search system uses the k -means++ algorithm (Arthur and Vassilvitskii, 2006), which partitions the embedding space into k distinct clusters. The optimization objective of the clustering process can be formulated as

$$\arg \min_{\mathcal{T}} \sum_{i=1}^k \sum_{e_{tool_i} \in \mathcal{T}_i} \left\| e_{tool_i} - \frac{1}{|\mathcal{T}_i|} \sum_{e \in \mathcal{T}_i} e \right\|_2^2, \quad (8)$$

where k denotes the predefined number of clusters, and each cluster $\mathcal{T}_i$ aims to minimize intra-cluster variance by aligning elements with their respective centroids. In this way, the tool APIs can be grouped based on semantic similarity in functionality, regardless of their original grouping within the MCP platform¹.

Such functional clustering is essential for ensuring system resilience. When a tool API fails during execution, having pre-grouped alternatives allows for immediate substitution with a functionally similar tool API. By implementing automated categorization, the system reduces reliance on manual annotations while enhancing the scalability and precision of tool API organization. Additionally, the AI search system refers to each cluster as a *toolkit*, representing a functionally unified set of tool APIs capable of addressing a specific class of problems, such as *search toolkit*, *location toolkit*, *weather toolkit*, *programmer toolkit*, et al. For instance, in the *search toolkit*, there are *Baidu AI Search*, *ArXiv MCP Server*, *Perplexity MCP Server*, *OpenAI Web Search MCP*, et al. It is imperative to emphasize that the *search toolkit* contributes substantial practical value by ensuring resilient, diversified, and timely access to information. This enhanced functionality not only streamlines the search process but also significantly improves the operational efficiency and robustness of AI search systems.

3.3.3. Query-Oriented Tool Retrieval

While the AI search system has tailored the tool documentation to meet the specific needs of the Planner, interaction with a vast array of tools is often indispensable in real-world applications. In such settings, feeding all tool descriptions into an LLM is neither feasible nor efficient, due to input length restrictions and latency concerns. This scalability challenge highlights the necessity of a robust tool retrieval mechanism, one that can select the most suitable tools based on the task at hand, enabling LLMs to operate effectively in complex, tool-rich environments. The AI search system initially employs an LLM-based dual-tower retrieval model for tool retrieval, which consists of an *LLM-based Prompt Encoder* and an *LLM-based Tool Encoder*. Specifically, the LLM-based prompt encoder extracts the user features and input prompts and generates the user

为克服这些局限，AI搜索系统着重于基于详细功能相似性对工具API进行自动分类。借助第3.3.1节中介绍的精细化工具API文档 D_{tool_i} ，AI搜索系统利用大语言模型生成工具API的简洁功能描述 $desc_{tool_i}$ 。随后，这些描述会通过文本嵌入模型 $\mathcal{M}_e$ 转化为高维语义嵌入 e_{tool_i} ，从而通过严谨的聚类分析识别出功能上高度一致的组别。上述过程可表述为：

$$desc_{tool_i} = \mathcal{LLM}(D_{tool_i}), \quad (6)$$

$$e_{tool_i} = \mathcal{M}_e(desc_{tool_i}). \quad (7)$$

为了将工具嵌入向量按功能相关性进行分组，AI搜索系统采用了 k -均值++算法（Arthur和Vassilvitskii, 2006），该算法可将嵌入空间划分为 k 个不同的簇。聚类过程的优化目标可表述为：

$$\arg \min_{\mathcal{T}} \sum_{i=1}^k \sum_{e_{tool_i} \in \mathcal{T}_i} \left\| e_{tool_i} - \frac{1}{|\mathcal{T}_i|} \sum_{e \in \mathcal{T}_i} e \right\|_2^2, \quad (8)$$

其中 k 表示预先设定的簇数量，每个簇 $\mathcal{T}_i$ 旨在通过使各元素与其对应质心对齐，来最小化簇内方差。如此一来，无论这些工具API在MCP平台中原本属于何种分组，都能依据功能语义相似性被归为一类。

这种功能性聚类对于保障系统韧性至关重要。当某个工具API在执行过程中发生故障时，预先分组的替代方案能实现与功能相似的工具API之间的即时替换。通过实施自动化分类机制，系统既能减少对人工标注的依赖，又能提升工具API组织的可扩展性与精确度。此外，AI搜索系统将每个簇称为工具包，它代表一组功能统一的工具API，能够解决特定类别的问题，例如搜索工具包、位置工具包、天气工具包、编程工具包等。例如在搜索工具包中，就包含百度AI搜索、ArXiv MCP服务器、Perplexity MCP服务器、OpenAI网页搜索MCP等。必须强调的是，搜索工具包凭借其确保信息获取的稳定性、多样性与及时性，创造了巨大的实用价值。这种增强后的功能不仅简化了搜索流程，还显著提升了AI搜索系统的运行效率与鲁棒性。

3.3.3. 面向查询的工具检索

尽管AI搜索系统已经针对规划器的特定需求定制了工具文档，但在现实应用中，与大量工具进行交互往往是必不可少的。在此类场景下，由于输入长度限制和延迟问题，将所有工具描述输入给大语言模型既不可行也不高效。这一可扩展性挑战凸显出构建强大工具检索机制的必要性——该机制能够根据当前任务选出最合适的工具，从而让大语言模型能够在工具繁多且复杂的环境中有效运作。AI搜索系统最初采用基于大语言模型的双塔检索模型来进行工具检索，该模型包含基于大语言模型的提示编码器和基于大语言模型的工具编码器。具体而言，基于大语言模型的提示编码器会提取用户特征与输入提示，进而生成用户嵌入向量。

¹<https://www.mcpworld.com/ai/mcp>

¹<https://www.mcpworld.com/ai/mcp>

embeddings. Meanwhile, the LLM-based tool encoder encodes the tool description, including the tool name, the tool usage, and the tool invoked queries, and produces the tool encodings. Subsequently, the AI search system computes the cosine similarity between the resulting user and tool embeddings and performs matching based on the similarity scores to retrieve the corresponding tools.

However, the above LLM-based tool retrieval strategies, typically focusing on semantic matching between queries and individual tools (Gao and Callan, 2021; Hofstätter et al., 2021; Izacard et al., 2021; Ma et al., 2022; Xiong et al., 2020), inadequately address complex queries that necessitate collaborative tool use, resulting in incomplete or suboptimal outcomes. For example, a query like calculating the value of 5 ounces of gold plus 1 million AMZN stocks in CNY requires tools for gold pricing, stock valuation, and currency conversion. If any are missing, the answer is incomplete. Traditional dense retrieval methods, focused on semantic similarity, often retrieve redundant tools (e.g., multiple stock tools) while omitting others, failing to capture the need for tool collaboration. This underscores the often-overlooked importance of completeness in tool retrieval. Considering these factors, a central challenge arises: how can we design a retriever that selects a complete and task-relevant set of tools to support the Planner effectively?

To address the above challenge, the AI search system leverages a PLM-enhanced retrieval method *COLT*, which integrates the semantic and collaborative dimensions of tool functionality (Qu et al., 2024). Initially, semantic relationships between queries and tools are established through representation learning techniques based on pre-trained language models. Subsequently, *COLT* employs graph-based learning methods to capture and refine collaborative interactions among tools, queries, and task scenarios. By adopting dual-view graph structures that explicitly model these interactions, *COLT* effectively captures the high-order collaborative information between tools. Furthermore, to promote balanced retrieval across all relevant tools, the learning objective incorporates a list-wise multi-label loss, preventing over-reliance on individual tools. Next, we detail the proposed tool retrieval method as follows.

Semantic Learning. *COLT* first focuses on semantic representation learning through dense retrieval, a widely adopted approach in modern retrieval systems (Guo et al., 2022; Zhao et al., 2023a). It employs a bi-encoder architecture powered by PLMs, such as BERT (Kenton and Toutanova, 2019), to independently encode both the query q and each tool description t into dense vector embeddings. These embeddings, obtained via mean pooling over the final layer outputs, are then compared using cosine similarity to estimate their initial semantic relevance. The training process is performed using the InfoNCE loss (Gutmann and Hyvärinen, 2010; Xiong et al., 2020), which contrasts positive tool-query pairs against negatives:

$$-\log \frac{e^{\text{sim}(q, t^+)}}{e^{\text{sim}(q, t^+)} + \sum_{j=1}^k e^{\text{sim}(q, t_j^-)}}, \quad (9)$$

where t^+ is the relevant tool, and $\{t_1^-, \dots, t_k^-\}$ is a set of k irrelevant tools. In this way, *COLT* is trained to learn effective semantic representations. However, such retrieval alone often fails to handle complex, multifaceted queries, limiting the completeness of tool selection.

Collaborative Learning. To move beyond simple semantic similarity and support completeness-oriented tool retrieval, it is crucial to model the collaborative relationships among tools. *COLT* reinterprets the ground-truth tool set for a given query as a “scene”, a conceptual unit that encapsulates the coordinated use of tools necessary to accomplish the underlying task. Consider the query “I want to travel to Paris.” Rather than seeking a single response, the query implies a composite intent involving transportation, accommodation, weather forecasting, and tourist

与此同时，基于大语言模型的工具编码器会对工具描述进行编码，这些描述涵盖工具名称、使用方法以及调用查询信息，最终生成工具编码。随后，AI搜索系统会计算所得到的用户嵌入向量与工具嵌入向量之间的余弦相似度，并依据相似度得分进行匹配，从而检索出对应的工具。

然而，上述基于大语言模型的工具检索策略通常仅侧重于查询与单个工具之间的语义匹配（Gao和Callan, 2021; Hofstätter等人, 2021; Izacard等人, 2021; Ma等人, 2022; Xiong等人, 2020），却未能有效应对那些需要协同使用多种工具的复杂查询，从而导致结果不完整或不够理想。例如，若要计算5盎司黄金加上100万股亚马逊股票以人民币计价的价值，就需要分别用到黄金定价、股票估值以及货币转换等工具；一旦缺少其中任何一类工具，计算结果便无法完整呈现。传统基于语义相似度的密集检索方法往往能召回大量功能重复的工具（比如多个用于股票相关计算的工具），却遗漏了其他必要的工具，因而无法体现工具协同使用的需求。这凸显出工具检索中完整性这一常被忽视的重要性。综合上述因素，一个核心难题随之浮现：我们该如何设计一种检索器，从而选出完整且符合任务需求的工具集，来有效辅助规划器开展工作？

为应对上述挑战，AI搜索系统采用了一种基于PLM增强的检索方法*COLT*，该方法融合了工具功能的语义维度与协同维度（Qu等人, 2024）。首先，借助预训练语言模型驱动的学习技术来建立查询与工具之间的语义关联；随后，*COLT*利用基于图的学习方法来捕捉并优化工具、查询及任务场景间的协同交互。通过采用能够显式建模这些交互的双视图图结构，*COLT*能够有效提取工具间的高阶协同信息。此外，为确保对所有相关工具的检索结果保持均衡，其学习目标中引入了列表级多标签损失函数，从而避免对个别工具的过度依赖。接下来，我们将详细介绍所提出的工具检索方法。

语义学习。*COLT*首先通过密集检索来实现语义表示学习，这也是现代检索系统中广泛应用的方法（Guo等人, 2022; Zhao等人, 2023a）。它采用由PLM驱动的双编码器架构，例如BERT（Kenton和Toutanova, 2019），分别独立地将查询 q 与每条工具描述 t 编码为密集向量嵌入。这些嵌入是通过对最后一层输出取均值池化得到的，随后利用余弦相似度来衡量它们之间的初始语义相关性。训练过程采用InfoNCE损失函数（Gutmann和Hyvärinen, 2010; Xiong等人, 2020）来进行，该函数会将正样本的工具-查询对与负样本进行对比：

$$-\log \frac{e^{\text{sim}(q, t^+)}}{e^{\text{sim}(q, t^+)} + \sum_{j=1}^k e^{\text{sim}(q, t_j^-)}}, \quad (9)$$

其中， t^+ 表示相关的工具，而 $\{t_1^-, \dots, t_k^-\}$ 则是一组 k 不相关的工具。通过这种方式，*COLT*被训练以学习有效的语义表征。不过，仅靠这种检索方式往往难以应对复杂且多面的查询需求，从而限制了所选取工具的完整性。

协同学习。若要超越简单的语义相似性并支持以完整性为导向的工具检索，对工具间的协作关系进行建模至关重要。*COLT*将针对特定查询的真实工具集重新定义为一种“场景”，即能够体现完成相应任务所需工具协同使用方式的概念单元。以查询“我想去巴黎旅行”为例，该查询并非寻求单一答案，而是隐含着涉及交通、住宿、天气预报以及旅游信息等方面的复合意图。

information. Addressing this multifaceted need requires retrieving a cohesive set of tools that operate collaboratively within the travel planning scene. To model these intricate relationships, COLT constructs three bipartite graphs that jointly encode the associations among queries, tools, and scenes: *query-scene*, *query-tool* and *scene-tool*. Building on the semantic representations from the first stage and the three constructed bipartite graphs, we propose a dual-view graph collaborative learning framework that captures both direct query-tool relevance and higher-order tool interactions within scenes. This dual perspective improves retrieval completeness by modeling tool cooperation more effectively. To align collaborative signals across views, COLT uses a cross-view contrastive loss. Additionally, a list-wise multi-label loss is introduced to ensure balanced retrieval from the full ground-truth set, avoiding overemphasis on individual tools.

With the cooperation of LLM-based tool retrieval and COLT, the AI search system could effectively retrieve a functionally complete set of tools from the entire tool repository in response to a given user query. This ensures that the downstream Planner is equipped with all the necessary tool capabilities, thereby facilitating more coherent and comprehensive planning and execution.

3.4. DAG-based Task Planning

Given the input query and the tool APIs retrieved accordingly, conventional approaches involve directly using these tools to obtain the required information and generate an answer. However, when the query is excessively complex and necessitates reasoning, this paradigm always fails to achieve the task through reasoning and multi-step tool invocation. In response, AI Search paradigm proposes a dynamic reasoning framework based on a DAG, effectively addressing the challenges associated with complex queries in the AI search system.

Specifically, the AI search system leverages a candidate tool API set $\mathcal{T}$ and the user query q as contextual input of the Planner, in a single inference pass, and generates a task graph $G = (V, E)$, where V is the set of vertices (*i.e.*, sub-tasks) and E refers to the set of dependency edges. Each vertex $v \in V$ represents an atomic and schedulable sub-task, which may either bind to an external tool or be fulfilled by the Executor using local LLM computation. For uniform specification, every vertex is equipped with the interface signature described as $\langle \arg(v), \text{ret}(v) \rangle$, where $\arg(v)$ declares the parameter slots that must be instantiated before execution of the sub-task, and $\text{ret}(v)$ describes the structure of the data returned afterwards. If the sub-task requires an external tool, the binding is recorded as $\tau(v) \in \mathcal{T}$; otherwise, $\tau(v) = \emptyset$ and the computation is performed locally.

Overall, the planning process can be formulated as the following mapping:

$$\Phi : (q, \mathcal{T}) \longrightarrow G, \quad (10)$$

which subjects to two adequacy conditions:

- *Atomic Sub-Tasks.* For every vertex v at most one tool type $\tau(v)$ can be bound, although that tool may be invoked multiple times inside the sub-task.
- *Sequential Dependency.* Second, the transitive closure of G must capture all sequential dependencies inherent in the semantic relations required to address q , thereby ensuring completeness while minimizing inter-task coupling.

To elicit a well-structured DAG G in a single forward pass, the AI search system employs a *chain-of-thought* $\rightarrow$ *structured-sketch* prompting strategy. Concretely, the Planner is first encouraged to decompose q implicitly through reasoning, annotating latent variables and dependencies.

要满足这种多元化的需求，就必须检索出能在旅行规划场景中协同工作的工具集合。为了建模这些复杂的关系，COLT构建了三种二分图来共同描述查询、工具和场景之间的关联：查询-场景、查询-工具以及场景-工具。基于第一阶段得到的语义表示及这三种二分图，我们提出了一种双视图图协同学习框架，该框架既能捕捉查询与工具之间的直接相关性，又能反映场景内工具间的高阶交互作用。这种双重视角通过更有效地建模工具协作关系，提升了检索的完整性。为使不同视图间的协同信号保持一致，COLT采用了跨视图对比损失函数；此外，还引入了列表级多标签损失函数，以确保从完整的真实集合中实现均衡检索，避免过分侧重个别工具。

借助基于大语言模型的工具检索机制与COLT的协同作用，AI搜索系统能够针对给定的用户查询，从整个工具库中高效检索出功能完备的工具集合。这确保了下游规划器具备所有必要的工具能力，进而实现更加连贯且全面的规划与执行。

3.4. 基于有向无环图的任务规划

在给定输入查询及相应检索出的工具API后，传统方法会直接利用这些工具来获取所需信息并生成答案。然而，当查询过于复杂、需要推理时，这种范式往往无法通过推理和多步工具调用来完成任务。为此，AI搜索范式提出了一种基于有向无环图的动态推理框架，有效解决了AI搜索系统中复杂查询带来的挑战。

具体而言，AI搜索系统会将候选工具API集合 $\mathcal{T}$ 以及用户查询 q 作为规划器的上下文输入，通过单次推理生成任务图 $G = (V, E)$ ，其中 V 代表顶点集合（即子任务）， E 则表示依赖关系边集合。每个顶点 $v \in V$ 都代表一个可调度的基本子任务，它要么与外部工具绑定，要么由执行器通过本地大语言模型计算来完成。为了统一规范，每个顶点均配备如下接口签名： $\langle \arg(v), \text{ret}(v) \rangle$ 。其中， $\arg(v)$ 用于声明子任务执行前必须填充的参数槽位，而 $\text{ret}(v)$ 则描述了后续返回的数据结构。若该子任务需要调用外部工具，则绑定关系记为 $\tau(v) \in \mathcal{T}$ ；否则，记为 $\tau(v) = \emptyset$ ，计算过程则在本地完成。

总体而言，规划过程可表述为如下映射关系：

$$\Phi : (q, \mathcal{T}) \longrightarrow G, \quad (10)$$

该映射需满足两项充分性条件：

- *原子子任务。*对于每个顶点 v ，最多只能绑定一种工具类型 $\tau(v)$ ；不过该工具可在子任务内部被多次调用。
- *顺序依赖性。*其次， G 的传递闭包必须涵盖为解决 q 所需语义关系中固有的所有顺序依赖关系，从而在确保完整性的同时尽量减少任务间的耦合。

为了在单次前向传播中生成结构良好的DAG G ，AI搜索系统采用了思维链 $\rightarrow$ 结构化草图提示策略。具体而言，首先引导规划器通过推理过程隐式地分解 q ，同时标注其中的潜变量及其依赖关系。

Then, the resulting draft of the reasoning is reorganized into a JSON standard format for DAG via instruction prompting. Unlike iterative ReAct-style reasoning, this approach produces a machine-readable, verifiable global plan simultaneously, substantially reducing token consumption and simplifying downstream parsing.

Once the task DAG G is established, the AI search system schedules execution by topological depth $\ell(\cdot)$. For each layer $V_d = \{v \mid \ell(v) = d\}$, all vertices are carried out in parallel. If $\tau(v_i) \neq \emptyset$ the corresponding tool is invoked one or more times, otherwise the sub-task is performed by the Executor individually. After executing, the resulting $\text{ret}(v_i)$ is then propagated along edge (v_i, v_j) to satisfy $\text{arg}(v_j)$. In the event of a vertex failure, the Master forwards the failure signal to the Planner. Subsequently, the Planner initiates a localized rollback and re-planning on the affected subgraph while preserving the remainder of the DAG G . This method enhances robustness without incurring the cost of a global restart.

In this way, DAG-based task planning externalizes multi-step reasoning into a structured task graph, achieving minimal context length, layer-wise parallelism, and full traceability. This design provides a principled, modular backbone for subsequent reflection mechanisms and reinforcement-learning optimization, which will be discussed in the remaining sections.

3.5. Master-Guided Re-Action

Real-time monitoring of sub-task execution and the evaluation of intermediate results, coupled with the ability to reflect, re-plan, and re-act based on those outcomes, is critical for AI Search paradigm. Therefore, the AI search system leverages a *Master-guided reflection, re-planning and re-action* mechanism.

Given an input query, the Planner initially constructs a DAG, after which the Executor carries out each task node until the final result is produced. Throughout this process, the Master continuously evaluates both the execution status of the sub-tasks and the completeness of their outcomes, engaging in reflection and dynamically updating the DAG as necessary. Specifically, in the AI search system, the Planner initially constructs a DAG based on the provided query, thereby outlining a series of sub-tasks to be executed. Subsequently, the Executor carries out each prescribed sub-task according to the DAG. If any sub-task fails to execute, the Master directs the Planner to re-plan either the affected portion of the DAG or the entire DAG. Conversely, upon the completion of each stage, whether the subtasks are executed individually or concurrently, the Master assesses the execution results of each subtask. If the results are found to be incomplete, the Master guides the Planner to augment the existing DAG by incorporating additional downstream sub-task nodes to achieve complete results.

3.6. Planner Optimization with RL Strategy

In practice, high-quality, accurately annotated data for supervised fine-tuning (SFT) presents substantial challenges, primarily because of the significant cost associated with manual annotation. Furthermore, AI search paradigm comprises four specialized agents (i.e., Master, Planner, Executor, and Writer) that collaboratively process queries to generate the final output. Consequently, optimizing the Planner in isolation may prove insufficient and could lead to suboptimal performance of the overall system. To address this issue, a promising strategy is to employ reinforcement learning (RL) to optimize the Planner in the context of multi-agent cooperation. Specifically, the proposed AI search system introduces a rule-based reward function that consists of four components: (1) *Final Answer Rewards*, (2) *User Rewards*, (3) *Formatting Rewards*, and (4) *Intermediate Execution Rewards*, as detailed below.

随后, 通过指令提示将推理生成的草稿重新组织为符合JSON标准的DAG格式。与迭代式的ReAct风格推理不同, 该方法能够同步生成机器可读且可验证的全局计划, 从而大幅降低令牌消耗并简化后续解析流程。

一旦任务DAG G 被建立, AI搜索系统便会依据拓扑深度 $\ell(\cdot)$ 来安排执行顺序。对于每一层 $V_d = \{v \mid \ell(v) = d\}$, 其中的所有顶点都会并行执行。如果 $\tau(v_i) \neq \emptyset$ 对应的工具被调用一次或多次, 否则该子任务便由执行器单独处理。执行完成后, 所得结果 $\text{ret}(v_i)$ 会沿着边 (v_i, v_j) 传播, 以满足 $\text{arg}(v_j)$ 的要求。一旦某个顶点执行失败, 主控制器会将故障信号发送给规划器; 随后, 规划器会在不影响DAG G 其余部分的前提下, 针对受影响的子图启动局部回滚与重新规划。这种方法在提升系统鲁棒性的同时, 无需承担全局重启所带来的开销。

通过这种方式, 基于DAG的任务规划将多步推理过程外化为结构化的任务图, 从而实现最短的上下文长度、逐层并行执行以及完整的可追溯性。这种设计为后续的反思机制与强化学习优化提供了原则清晰且模块化的基础, 相关内容将在后续章节中详细探讨。

3.5. 主控制器引导下的重新反应机制

对子任务执行情况的实时监测以及对中间结果的评估, 再加上基于这些结果进行反思、重新规划并作出响应的能力, 对于AI搜索范式而言至关重要。因此, AI搜索系统采用了由主控制器引导的反思、重新规划和重新反应机制。

给定输入查询后, 规划器首先构建有向无环图, 随后执行器会依次完成各个任务节点, 直至生成最终结果。在此过程中, 主控制器会持续评估子任务的执行状态及其结果的完整性, 并进行反思, 必要时动态更新有向无环图。具体而言, 在AI搜索系统中, 规划器会根据输入的查询先构建有向无环图, 从而确定一系列待执行的子任务。接着, 执行器按照该图的要求逐一执行各子任务。一旦某个子任务执行失败, 主控制器便会指示规划器重新规划受影响的图部分或整个图。相反, 在每个阶段结束后——无论子任务是单独还是并行执行——主控制器都会评估各子任务的执行结果。若发现结果不完整, 主控制器会引导规划器在现有图中添加额外的下游子任务节点, 以得到完整的结果。

3.6. 基于强化学习策略的规划器优化

在实践中, 用于监督微调的高质量、标注准确的数据获取存在巨大挑战, 这主要是因为人工标注的成本极高。此外, AI搜索范式包含四个专业智能体(即Master、Planner、Executor和Writer), 它们协同处理查询以生成最终输出。因此, 仅对Planner进行单独优化可能并不充分, 甚至可能导致整个系统的性能不佳。为解决这一问题, 一种颇具前景的策略是利用强化学习在多智能体协作场景下优化Planner。具体而言, 所提出的AI搜索系统引入了一种基于规则的奖励函数, 该函数由四个组成部分构成: (1) 最终答案奖励, (2) 用户反馈奖励, (3) 格式规范奖励, 以及(4)中间执行奖励, 具体内容如下。

- **Final Answer Rewards.** The primary objective of the Planner is to decompose the task to obtain the correct final answer. Accordingly, the final answer reward could be defined as

$$\mathcal{R}_{Answer} = \alpha_{Answer} \cdot \mathbb{1}[Answer = G_{Answer}], \quad (11)$$

where $\alpha_{Answer} > 0$ is a scaling factor, and $Answer$ denotes the final answer produced by the multi-agent workflow. G_{Answer} is the ground-truth answer.

- **User Feedback Rewards.** User feedback serves as an effective signal, enabling the Planner to further adjust the decomposition of sub-tasks. Consequently, the reward of user feedback could be formulated as

$$\mathcal{R}_{Feedback} = \begin{cases} \alpha_{Feedback}, & \text{if users accept the sub-tasks;} \\ \beta_{Feedback}, & \text{if users reject the sub-tasks,} \end{cases} \quad (12)$$

where $\alpha_{Feedback} > 0$ is the reward when the sub-tasks are accepted by users, while $\beta_{Feedback} < 0$ is the penalty when they are rejected. $\alpha_{Feedback}$ and $\beta_{Feedback}$ are both scaling factors.

- **Formatting Rewards.** To ensure the generated output adheres to the desired format, the formatting reward could be defined as

$$\mathcal{R}_{Format} = \begin{cases} \alpha_{Format}, & \text{if output is correctly formatted;} \\ \beta_{Format}, & \text{if output is incorrectly formatted,} \end{cases} \quad (13)$$

where $\alpha_{Format} > 0$ represents the reward for correct formatting, whereas $\beta_{Format} < 0$ denotes the penalty for formatting errors. α_{Format} and β_{Format} are two scaling factors.

- **Intermediate Execution Rewards.** To ensure the executability of the decomposed sub-tasks, the intermediate execution reward could be defined as

$$\mathcal{R}_{Execution} = \frac{\alpha_{Execution}}{m} \cdot \sum_{i=1}^m f_{Executor}(task_i), \quad (14)$$

where $\alpha_{Execution} > 0$ is a scaling factor, and m is the total number of sub-tasks. Moreover, $task_i$ denotes the i -th sub-task, and $f_{Executor}(task_i) \in [0, 1]$ represents the execution success score of each sub-task.

- **Overall Reward and Objective Function.** The overall reward could be formulated as the sum of all the individual reward components mentioned above

$$\mathcal{R}_{All} = \mathcal{R}_{Answer} + \mathcal{R}_{Feedback} + \mathcal{R}_{Format} + \mathcal{R}_{Execution}. \quad (15)$$

Given the above rewards, the AI search system leverages Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to optimize the Planner. Specifically, for a given complex query q , the AI search system samples a set of outputs $o = \{o_1, o_2, \dots, o_N\}$ from the previous planner policy model $\pi_{\theta_{old}}$. Corresponding to these outputs, it computes a set of rewards $r = \{r_1, r_2, \dots, r_N\}$ based on the rule-defined reward function (Eq. 15). The advantage for each token is defined as the normalized reward as

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}. \quad (16)$$

The AI search system optimizes the policy by maximizing the following objective function as

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)] \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \{\gamma - \beta \mathbb{D}_{KL}[\pi_{\theta} || \pi_{ref}]\}, \quad (17)$$

- **最终答案奖励。** Planner的主要目标是拆解任务以获取正确的最终答案。据此，最终答案奖励可定义为：

$$\mathcal{R}_{Answer} = \alpha_{Answer} \cdot \mathbb{1}[Answer = G_{Answer}], \quad (11)$$

其中 $\alpha_{Answer} > 0$ 为缩放因子， $Answer$ 表示多智能体工作流产生的最终答案。 G_{Answer} 即为真实答案。

- **用户反馈奖励。** 用户反馈是一种有效的信号，能够帮助规划器进一步调整子任务的分解方式。因此，用户反馈的奖励可表述为：

$$\mathcal{R}_{Feedback} = \begin{cases} \alpha_{Feedback}, & \text{if users accept the sub-tasks;} \\ \beta_{Feedback}, & \text{if users reject the sub-tasks,} \end{cases} \quad (12)$$

其中 $\alpha_{Feedback} > 0$ 表示子任务被用户接受时的奖励，而 $\beta_{Feedback} < 0$ 则表示子任务被拒绝时的惩罚值。 $\alpha_{Feedback}$ 和 $\beta_{Feedback}$ 均为缩放因子。

- **格式奖励。** 为确保生成的输出符合预期格式，格式奖励可定义为：

$$\mathcal{R}_{Format} = \begin{cases} \alpha_{Format}, & \text{if output is correctly formatted;} \\ \beta_{Format}, & \text{if output is incorrectly formatted,} \end{cases} \quad (13)$$

其中 $\alpha_{Format} > 0$ 代表格式正确的奖励， $\beta_{Format} < 0$ 则表示格式错误的惩罚值。 α_{Format} 和 β_{Format} 是两个缩放因子。

- **中间执行奖励。** 为确保分解后的子任务具备可执行性，中间执行奖励可定义为：

$$\mathcal{R}_{Execution} = \frac{\alpha_{Execution}}{m} \cdot \sum_{i=1}^m f_{Executor}(task_i), \quad (14)$$

$\alpha_{Execution} > 0$ 为缩放因子， m 表示子任务的总数。此外， $task_i$ 代表第 i 个子任务，而 $f_{Executor}(task_i) \in [0, 1]$ 则表示每个子任务的执行成功得分。

- **总体奖励与目标函数。** 总体奖励可表述为上述所有单个奖励分量之和。

$$\mathcal{R}_{All} = \mathcal{R}_{Answer} + \mathcal{R}_{Feedback} + \mathcal{R}_{Format} + \mathcal{R}_{Execution}. \quad (15)$$

基于上述奖励，AI搜索系统利用分组相对策略优化（GRPO）算法（Shao等人，2024年）来对规划器进行优化。具体而言，针对给定的复杂查询 q ，AI搜索系统会从之前的规划器策略模型 $\pi_{\theta_{old}}$ 中采样一组输出 $o = \{o_1, o_2, \dots, o_N\}$ 。针对这些输出，系统会根据规则定义的奖励函数（公式15）计算出一组奖励 $r = \{r_1, r_2, \dots, r_N\}$ 。每个词元的优势被定义为归一化后的奖励。

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}. \quad (16)$$

AI搜索系统通过最大化下述目标函数来优化相关策略。

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^G \sim \pi_{\theta_{old}}(O|q)] \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \{\gamma - \beta \mathbb{D}_{KL}[\pi_{\theta} || \pi_{ref}]\}, \quad (17)$$

where $\gamma = \min [\lambda \hat{A}_{i,t}, \text{clip}(\lambda, 1 - \varepsilon, 1 + \varepsilon) \hat{A}_{i,t}]$, and $\lambda = \frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}$. Here, π_{θ} denotes the current planner policy model, and $o_{i,t}$ is the t -th token of the output sequence o_i . Additionally, $o_{i,<t}$ represents the tokens preceding $o_{i,t}$, and β and ε are hyper-parameters.

4. Task Executor

Following the meticulously designed task plan generated by the Planner, the AI search paradigm incorporates a task executor module, known as the Executor, which is responsible for invoking the appropriate tools in accordance with the prescribed plan and executing specific subtasks to produce the requisite outcomes for final answer generation. Among these tools, the web search functionality is a critical component and represents the most frequently utilized tool within the system. This need arises not only from queries that require information beyond the model’s parametric knowledge, such as retrieving current gold prices, but also from the imperative to provide query responses that are both timely and authoritative, thereby meeting key user satisfaction dimensions. The following section presents our methodological advancements aimed at enhancing the capability of the web search component.

4.1. Background

In traditional pre-trained language model (PLM)-based retrieval systems, task execution primarily involves retrieving and ranking a list of documents in response to a single search query, with the goal of meeting the immediate needs of the user. This approach typically transforms a user’s query into a ranked document list, from top to bottom, aiming to deliver the most relevant results efficiently. With the rapid development of LLMs, there has been growing interest among researchers to leverage them for document ranking. Currently, there are three main approaches for document ranking with LLMs based on their input, such as *Pointwise*, *Pairwise*, and *Listwise*, which are introduced as follows.

Pointwise. These approaches utilize LLMs to independently assess the relevance of each candidate document with the given query. Formally, given a query q and a retrieved document $\mathcal{D}_i$, we aim to instruct the LLM to generate a quantitative, comparable score for each query and candidate document pair, denoted as the basic labeling unit as $u(q, \mathcal{D}_i)$. This score can be a graded label commonly used in human annotation or a logit value output by the model. Then, we apply the labeling unit to all the candidate documents and obtain the final rankings based on the sorted scores. For instance, some works directly prompt LLMs to generate whether the provided candidate document is relevant to the query and the documents are labeled based on the normalized likelihood of generating a “yes” response (Liang et al., 2022; Nogueira et al., 2020; Sun et al., 2023a). The other method involves employing LLMs to generate a relevant query for each candidate document and then re-rank these documents based on the likelihood of generating the actual query (Sachan et al., 2022). Another approach involves appending an output layer to the LLM’s representation and then employing SFT with human-annotated samples for re-ranking tasks. This process produces a scalar score that reflects the relative query-document relevance (Ma et al., 2024).

Pairwise. These approaches use LLMs to compare the relevance of a pair of candidate documents with the given query. Formally, given a query q and a pair of retrieved documents $(\mathcal{D}_i, \mathcal{D}_j)$, we aim to ask the LLM to determine which document is more relevant to the given query q , denoted as the basic labeling unit $u(q, \mathcal{D}_i, \mathcal{D}_j)$. There are two principal approaches leveraging the pairwise score unit: the all-pairs method and the sorting-based method (Pradeep et al., 2021; Qin et al., 2023c). The all-pairs method generates all possible permutations of

其中, $\gamma = \min [\lambda \hat{A}_{i,t}, \text{clip}(\lambda, 1 - \varepsilon, 1 + \varepsilon) \hat{A}_{i,t}]$, 而 $\lambda = \frac{\pi_{\theta}(o_{i,t}|q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q, o_{i,<t})}$ 。此处, π_{θ} 表示当前的规划器策略模型, $o_{i,t}$ 分别是输出序列 o_i 中的第 t 个词元。此外, $o_{i,<t}$ 代表 $o_{i,t}$ 之前的词元, β 和 ε 则为超参数。

4. 任务执行器

在规划器生成的精心设计的任务计划指导下, AI搜索范式引入了名为“执行器”的任务执行模块。该模块负责依据既定计划调用合适的工具, 执行具体的子任务, 从而为最终答案的生成提供必要的结果。在这些工具中, 网页搜索功能是一项关键组件, 也是系统中使用频率最高的工具。这种需求不仅源于那些需要获取模型参数知识之外信息的查询(例如查询当前金价), 还源于提供及时且权威的查询响应这一要求, 以满足用户满意度的关键维度。下一节将介绍我们为提升网页搜索组件能力所做出的方法学改进。

4.1. 背景

在传统的基于预训练语言模型(PLM)的检索系统中, 任务执行主要涉及针对单个查询检索并排序文档列表, 以满足用户的即时需求。这种方法通常将用户的查询转化为从上到下的排序文档列表, 旨在高效提供最相关的结果。随着大语言模型(LLM)的快速发展, 研究人员越来越倾向于利用它们来进行文档排序。目前, 根据输入形式的不同, 基于大语言模型的文档排序主要有三种方法: *pointwise*、*pairwise* 和 *listwise*, 下文将对它们进行详细介绍。

逐点式方法。这类方法利用大语言模型独立评估每个候选文档与给定查询的相关性。具体而言, 给定一个查询 q 以及一个检索到的文档 $\mathcal{D}_i$, 我们的目标是指示大语言模型为每一对查询与候选文档生成可比较的量化分数, 这一基本标注单元被记为 $u(q, \mathcal{D}_i)$ 。该分数既可以是人类标注中常用的分级标签, 也可以是模型输出的逻辑值。随后, 我们将这一标注单元应用于所有候选文档, 再根据排序后的分数得出最终排名。例如, 部分研究直接提示大语言模型判断所提供的候选文档是否与查询相关, 并根据生成“是”响应的归一化概率来对文档进行标注 (Liang等人, 2022年; Nogueira等人, 2020年; Sun等人, 2023a年)。另一种方法是让大语言模型为每个候选文档生成一个相关的查询, 再根据生成实际查询的概率对这些文档重新排序 (Sachan等人, 2022年)。还有一种方法是为大语言模型的表征添加一个输出层, 再利用人工标注的样本通过监督微调来执行重排任务。这一过程会生成一个反映查询与文档相对相关性的标量分数 (Ma等人, 2024年)。

成对比较法。这类方法利用大语言模型来评判一对候选文档与给定查询的相关性。具体而言, 给定一个查询 q 以及一对检索到的文档 $(\mathcal{D}_i, \mathcal{D}_j)$, 我们的目标是让大语言模型确定哪篇文档与给定查询 q 的相关性更高, 这一基本标注单元被记作 $u(q, \mathcal{D}_i, \mathcal{D}_j)$ 。利用这种成对评分单元主要有两种方法: 全配对法和基于排序的方法 (Pradeep等人, 2021年; Qin等人, 2023c年)。全配对法会生成候选文档集合中所有可能的文档对。

document pairs from the candidate documents set. Then, each pair is evaluated independently by the LLM, and a score function is then used to assign a score to each document based on the pairwise comparisons. The final ranking is determined by the total score each document receives. In contrast, the sorting-based method utilizes sorting algorithms, such as heap sort and bubble sort, to leverage efficient data structures for selectively comparing document pairs. This enables the most relevant documents to be quickly identified and placed at the top of the final ranking without the need for exhaustive comparison of all possible pairs. More advanced research, such as PRP-Graph (Luo et al., 2024), draws inspiration from real-world tournaments and incorporates graph structures into pairwise ranking to further improve efficiency.

Listwise. These approaches utilize LLMs to directly rank multiple documents within a single window based on their relevance to the given query (Chen et al., 2025a; Liu et al., 2024; Sun et al., 2023b). Formally, given a query q and a list of retrieved documents $\{D_1, \dots, D_m\}$, we aim to ask the LLM to directly generate an ordered doc-list based on the relevance to the given query q , denoted as the basic labeling unit $u(q, \{D_1, \dots, D_m\})$. Since the Listwise method achieves a good trade-off between effectiveness and efficiency for ranking multiple documents at one time, they are considered as the preferred prompting strategy for LLM ranking. In the following sections, we will detail two Listwise-based ranking methods in practice: RankGPT (Sun et al., 2023b) and TourRank (Chen et al., 2025a).

While effective in straightforward search scenarios, these methods are still based on the traditional paradigm, lacking the flexibility and adaptability necessary to handle more complex interactions, such as multi-turn conversations, intent-driven query decomposition, and answer synthesis, which are increasingly important in modern AI applications.

To address these limitations, recent technological developments have paved the way for a fundamental shift in retrieval paradigms. With the rapid advancements in LLMs, there has been a significant paradigm shift in task execution strategies. As illustrated in Fig. 5, our AI search system moves away from the traditional PLM-based executor towards an integrated approach that leverages both retrieval and reasoning capabilities of LLMs. Instead of solely ranking documents based on a single query, the system now processes multiple, complex queries, potentially across multiple rounds, using LLM-based retrieval and ranking modules. The goal shifts from merely producing a ranked list for a specific user need to recalling a comprehensive document set that can serve as a reference for the LLM. This set provides the

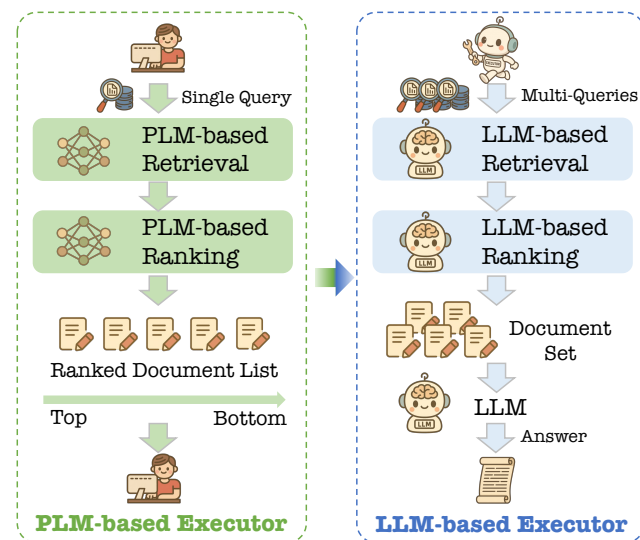


Figure 5 | An illustration of the paradigm shift for the task Executor. The left panel shows the traditional task executor based on PLMs for web search, which primarily addresses a single query by transforming the web search task into a ranking problem and aligning the results with user preferences. The right panel illustrates the paradigm shift in the task execution for AI search. Our AI search system processes multiple queries by re-framing the traditional ranking problem as a recall problem, shifting the objective from aligning user preferences to aligning LLM preferences.

随后，每对文档由大语言模型独立评估，再通过评分函数根据成对比较结果为每篇文档赋予相应分数。最终排名由每篇文档获得的总分决定。相比之下，基于排序的方法则运用堆排序、冒泡排序等排序算法，借助高效的数据结构来选择性地比较文档对。这种方式无需对所有可能的文档对进行穷尽式比较，就能快速识别出相关性最高的文档并将其置于最终排名的顶端。更前沿的研究，如PRP-Graph (Luo等人, 2024年)，从现实世界的锦标赛中汲取灵感，将图结构引入成对排序中，进一步提升了效率。

列表式排序。这类方法利用大语言模型在单一窗口内直接对多个文档按照其与给定查询的相关性进行排序（Chen等人, 2025a; Liu等人, 2024; Sun等人, 2023b）。具体而言，给定一个查询 q 以及一组检索到的文档 $\{D_1, \dots, D_m\}$ ，我们的目标是让大语言模型直接生成按相关性排序的文档列表，这一基本标注单元记为 $u(q, \{D_1, \dots, D_m\})$ 。由于列表式排序方法在同时排序多个文档时能在效果与效率之间取得良好平衡，因此被视为大语言模型排序的首选提示策略。在后续章节中，我们将详细介绍两种基于列表式排序的实际应用方法：RankGPT (Sun等人, 2023b) 和 TourRank (Chen等人, 2025a)。

尽管在简单的搜索场景中这些方法十分有效，但它们仍基于传统范式，缺乏处理更复杂交互所需的灵活性与适应性，比如多轮对话、以意图为导向的查询分解等。

答案合成在现代人工智能应用中正变得越来越重要。

为克服这些局限，近期的技术发展为检索范式的根本性变革铺平了道路。随着大语言模型的快速发展，任务执行策略也发生了显著的范式转变。如图5所示，我们的AI搜索系统摒弃了传统的基于PLM的executor，转而采用一种整合了大语言模型检索与推理能力的全新方法。该系统不再仅依据单一查询对文档进行排序，而是能够利用基于大语言模型的检索与排序模块来处理多个复杂查询，甚至可跨越多个轮次进行处理。其目标也从仅为满足特定用户需求而生成排序列表，转变为召回可作为大语言模型参考的全面文档集。

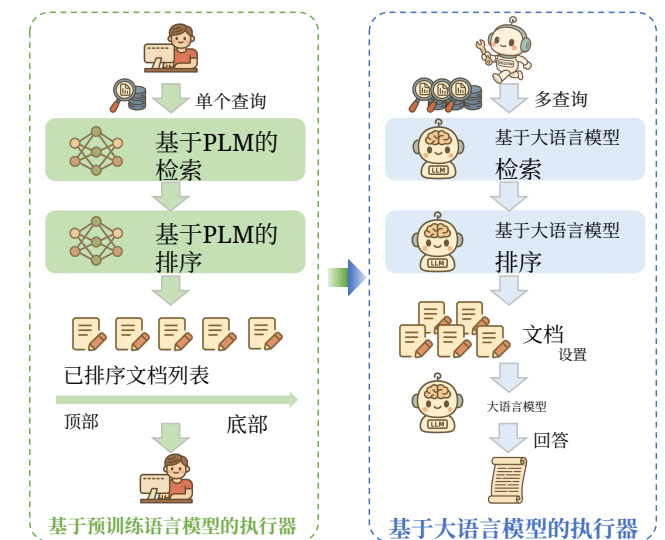


图 5 | 展示了任务执行器范式的转变过程。左侧面板呈现的是基于预训练语言模型的传统网页搜索任务执行器，它主要通过将网页搜索任务转化为排序问题来处理单个查询，并使搜索结果与用户偏好相匹配。右侧面板则描绘了AI搜索中任务执行的范式转变：我们的AI搜索系统通过把传统的排序问题重新定义为召回问题来处理多个查询，其目标也从匹配用户偏好转变为匹配大语言模型的偏好。

contextual foundation for the LLM to generate high-quality, nuanced answers tailored to the user’s evolving intent.

Building upon this motivation, the AI search system focuses on designing a task executor that effectively harnesses LLM capabilities. In this section, we detail the design and motivation behind our task executor, focusing on three core perspectives: *LLM Preference Alignment*, which ensures the system’s targets align with LLMs; *Lightweight System*, aimed at processing multiple complex queries; and *LLM-Augmented Features*, which enhances the retrieval process with LLM-augmented features.

4.2. LLM Preference Alignment

Traditional retrieval and ranking systems primarily rely on manually defined preferences and heuristics, which often fail to fully leverage the deep understanding and nuanced preferences of LLMs. As LLMs demonstrate significant advantages in comprehending complex semantics, handling multi-modal data, and performing sophisticated reasoning, the goal shifts from merely aligning with heuristic-guided preferences to aligning with the LLM-human co-evolutionary preferences. This transition enables the retrieval system to better facilitate high-quality answer generation by guiding the selection towards documents favored by the LLM. Ultimately, aligning with LLM preferences enables more intelligent, autonomous, and effective retrieval strategies, thereby improving the overall performance of the question-answering system.

To achieve this, the AI search system constructs data from three perspectives: *LLM Labeling*, *Reference Selection*, *Generation Reward*. These perspectives guide the training process by capturing different aspects of LLM preferences. Moreover, the AI search system further aligns the preference into the LLM retrieval model through *Distillation of LLM Ranking*, which helps the model internalize the learned preferences and improve its retrieval performance.

4.2.1. LLM Labeling

Compared to manual labeling processes, LLMs can efficiently annotate vast datasets at scale, thereby substantially reducing both the temporal and human resource investments required for training retrieval and ranking models. Furthermore, LLMs deliver highly consistent annotations, mitigating the subjective biases and inter-annotator variability that commonly afflict human-generated labels. By adjusting prompt designs or fine-tuning for specific domains or tasks, LLMs exhibit exceptional flexibility in meeting diverse annotation requirements. As previously noted, the AI search system utilizes listwise methods for document annotation, as these approaches effectively strike a balance between effectiveness and efficiency. In this work, the AI search system primarily utilizes two methods as follows: RankGPT (Sun et al., 2023b) and TourRank (Chen et al., 2025a).

RankGPT. When LLMs come to ranking tasks, especially in the context of passage re-ranking, the limited context window of LLMs poses a significant challenge. Most LLMs have a maximum context length that restricts the number of passages they can process simultaneously. This limitation is particularly problematic in scenarios where a large number of candidate documents need to be ranked, such as in web search or large-scale IR systems. To address this challenge, the AI search system utilizes a strategy called the sliding window approach RankGPT (Sun et al., 2023b). This method allows the efficient re-ranking of large candidate sets without exceeding the context window limitations of the LLM. By breaking down the ranking task into smaller, manageable segments, RankGPT ensures that the LLM can effectively process and rank each subset of documents while maintaining the overall ranking quality.

这一文档集为大语言模型提供了上下文基础，从而能够生成契合用户不断变化的需求、高质量且细腻精准的回答。

基于这一理念，AI搜索系统致力于设计一种能够有效发挥大语言模型能力的任务执行器。在本节中，我们将详细介绍该任务执行器的设计思路与初衷，主要从三个核心维度展开：大语言模型偏好对齐，旨在确保系统的目标与大语言模型的偏好保持一致；轻量级系统，用于高效处理多个复杂查询；以及大语言模型增强功能，通过引入大语言模型增强的功能来优化检索过程。

4.2. 大语言模型偏好对齐

传统的检索与排序系统主要依赖人工设定的偏好规则与启发式方法，这些方法往往无法充分利用大语言模型在深层语义理解、多模态数据处理以及复杂推理方面的优势。由于大语言模型在理解复杂语义、处理多模态数据及进行精细推理方面表现突出，检索系统的目标也从单纯遵循启发式偏好转变为与大语言模型及人类共同演进的偏好相一致。这种转变能够使检索系统更好地引导文档选择，从而选出受大语言模型青睐的文档，进而助力高质量答案的生成。最终，与大语言模型偏好对齐能够实现更智能、自主且高效的检索策略，全面提升问答系统的整体性能。

为实现这一目标，AI搜索系统从三个维度构建数据：*LLM*标注、参考选择以及生成奖励。这些维度通过捕捉LLM偏好的不同方面来指导训练过程。此外，AI搜索系统还通过*LLM*排序蒸馏将这种偏好融入LLM检索模型，从而帮助模型内化所学到的偏好，提升其检索性能。

4.2.1. LLM标注

与人工标注流程相比，LLM能够高效地对大规模数据集进行标注，从而大幅减少训练检索与排序模型所需的时间和人力资源投入。此外，LLM提供的标注结果具有高度一致性，可有效消除人类标注中常见的主观偏差及标注者间的差异。通过调整提示词设计或对特定领域或任务进行微调，LLM能灵活满足各类标注需求。如前所述，AI搜索系统采用列表式方法进行文档标注，因为这类方法在效果与效率之间取得了良好平衡。在本研究中，AI搜索系统主要采用了以下两种方法：RankGPT (Sun等人, 2023b) 和TourRank (Chen等人, 2025a)。

RankGPT. 当大语言模型应用于排序任务，尤其是段落重排场景时，其有限的上下文窗口会带来重大挑战。大多数大语言模型所能处理的最大上下文长度有限，因此无法同时处理大量段落。在需要排序大量候选文档的场景中——比如网络搜索或大规模信息检索系统——这一限制尤为突出。为应对这一挑战，AI搜索系统采用了一种名为滑动窗口法的策略，即RankGPT (Sun等人, 2023b)。该方法能够在不超出大语言模型上下文窗口限制的前提下，高效地对大量候选集进行重排。通过将排序任务拆分成更小、更易处理的片段，RankGPT确保大语言模型能够有效处理并排序每一组文档，同时维持整体的排序质量。

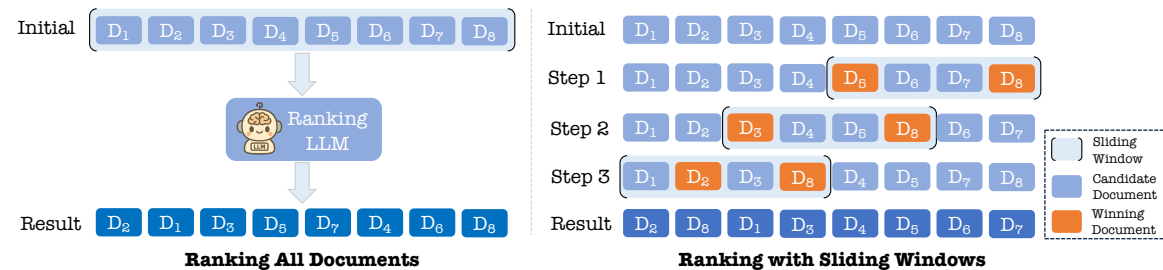


Figure 6 | The illustration of ranking all documents and ranking with sliding windows.

The sliding window strategy employed by RankGPT involves ranking a subset of candidate documents within a sliding window, starting from the bottom of the initial ranking list and moving forward. This approach ensures that each document is compared with multiple others across overlapping windows, maintaining the effectiveness of LLM ranking while remaining computationally feasible for long candidate lists. The process can be broken down into the following stages:

- **Initialization.** Given the retrieved set of M candidate documents from the retrieval model, the sliding window strategy begins with the last w documents, i.e., from the $(M - w + 1)$ -th to the M -th passage. These documents are input to the LLM, which returns a ranked order for this subset. The choice of w (window size) is crucial as it determines the number of documents processed in each iteration. A larger window size allows the LLM to consider more documents at once, potentially improving the ranking quality, but also increases the computational load and the risk of exceeding the context window limitations.
- **Sliding.** Once the initial window is ranked, the window slides backward by s documents. The new window now covers documents from $(M - w - s + 1)$ to $(M - s)$. The LLM is again used to rank this subset, which may overlap with the previous window. The step size s controls the overlap between consecutive windows. A smaller step size increases the overlap, allowing the LLM to compare more documents across windows, but also increases the number of iterations required to cover all documents.
- **Iteration.** The sliding and re-ranking process is repeated until the window reaches the beginning of the passage list and all documents have been covered. This iterative approach ensures that each document is ranked multiple times, allowing the LLM to provide more nuanced and consistent ranking signals across the entire candidate set.

In conclusion, the sliding window strategy employed by RankGPT effectively addresses the context window limitations of LLMs, making it possible to leverage their powerful ranking capabilities for large-scale information retrieval tasks. This approach not only improves the efficiency and effectiveness of passage re-ranking but also offers a scalable solution for various IR applications.

TourRank. LLMs have shown great potential in zero-shot document ranking. However, existing LLM-based ranking methods face several limitations. First, the maximum context length of LLMs restricts the number of documents that can be processed simultaneously, making it challenging to rank a large number of documents at once. Second, the ranking results are highly sensitive to the order in which documents are presented to the LLM, leading to inconsistent outcomes. Third, achieving a balance between computational cost and ranking performance remains a significant challenge. To address these issues, the AI search system introduces a novel approach inspired by the structure of sports tournaments, named TourRank (Chen et al., 2025a).

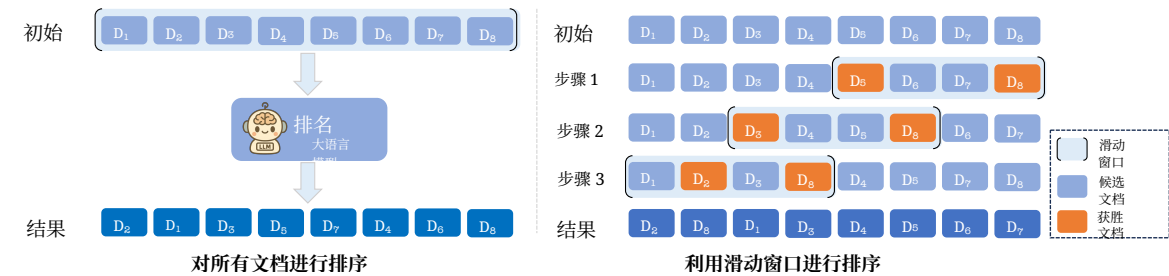


图 6 | 所有文档排序以及滑动窗口排序的示意图。

RankGPT采用的滑动窗口策略是：从初始排序列表的底部开始，在滑动窗口内对一部分候选文档进行排序，并逐步向前推进。这种方式使得每份文档都能在重叠的窗口中与多份其他文档进行比较，从而在保持大语言模型排序有效性的同时，也能适应长候选列表带来的计算需求。整个过程可分为以下几个阶段：

- **初始化。** 给定检索模型返回的 M 个候选文档，滑动窗口策略从最后 w 个文档开始，也就是从第 $M - w + 1$ 段到第 M 段。这些文档会被输入到大语言模型中，由模型为这一子集生成排序结果。 w （窗口大小）的选择至关重要，因为它决定了每次迭代中处理的文档数量。较大的窗口大小能让大语言模型一次性考虑更多文档，从而有望提升排序质量，但也会相应增加计算量，并增大超出上下文窗口限制的风险。
- **滑动。** 初始窗口排序完成后，窗口会向后滑动 s 个文档。此时新窗口覆盖的文档范围变为从 $M - w - s + 1$ 到 $M - s$ 。再次利用大语言模型对这一子集进行排序，该子集可能与前一个窗口存在重叠。步长 s 决定了相邻窗口之间的重叠程度：步长越小，重叠部分越多，大语言模型就能对比更多跨窗口的文档，不过这也意味着覆盖全部文档所需的迭代次数会增加。
- **迭代。** 滑动与重新排序的过程会不断重复，直到窗口抵达段落列表的开头且所有文档均被覆盖为止。这种迭代方式能够确保每份文档都被多次排序，从而使大语言模型在整个候选文档集中给出更为细致且一致的排序信号。

综上所述，RankGPT所采用的滑动窗口策略有效解决了大语言模型在上下文窗口方面的限制，进而得以充分发挥其在大规模信息检索任务中的强大排序能力。该方法不仅提升了段落重新排序的效率与效果，还为各类信息检索应用提供了可扩展的解决方案。

TourRank. 大型语言模型在零样本文档排序任务中展现出了巨大潜力。然而，现有的基于大型语言模型的排序方法仍存在诸多局限。首先，大型语言模型的最大上下文长度限制了可同时处理的文档数量，因此难以一次性对大量文档进行排序。其次，排序结果对文档提交给模型时的顺序极为敏感，从而导致结果不一致。第三，如何在计算成本与排序性能之间取得平衡仍是一项重大挑战。为解决这些问题，AI搜索系统借鉴体育赛事的结构，提出了一种名为 TourRank 的新方法（Chen 等人，2025a）。

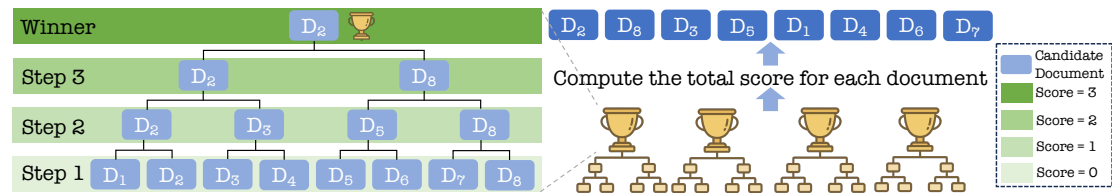


Figure 7 | The illustration of ranking with a tournament strategy.

In sports tournaments, there are often multiple participants, which requires that their rankings be determined as fairly as possible within a short period of time. In many sports tournaments, such as the FIFA World Cup, a group stage format is often adopted. Top-seeded players or teams are distributed across different groups, and multiple group matches are conducted in parallel. This allows for the rapid selection of the top-ranked participants, such as the gold, silver, and bronze medalists.

Similarly, if we liken the document ranking process to a sports tournament, each document can be considered as a participant. By simulating the structure of sports tournaments, we can quickly obtain the ranking of the documents. Specifically, suppose we have M documents to rank. A basic tournament proceeds as follows:

- **Group and Select.** Given a set of M candidate documents, TourRank begins by dividing them into several smaller groups. Each group contains a manageable number of documents that fit within the context window of the LLM. For example, if there exist 100 documents to rank, they might be divided into groups of 20 documents each. Within each group, the LLM evaluates the relevance of the documents to the query and selects the top N most relevant documents to advance to the next stage. This process is analogous to the group stage in sports tournaments, where teams compete within their respective groups to qualify for the next round.
- **Progressive Rounds.** After the initial grouping and selection, the winners from each group are pooled together and re-divided into new groups for the next round. This hierarchical, multi-stage structure continues iteratively, with each round reducing the number of documents and selecting the most relevant ones. For instance, when starting with 100 documents divided into 10 groups of 10, the first round selects 5 documents from each group, thereby reducing the total to 50 documents. These 50 documents are then divided into 5 groups of 10, and the process repeats. Each document earns 1 point whenever it is selected to advance to the next stage.
- **Points System and Multiple Rounds.** The points system is a crucial component of TourRank. Since the scores obtained from a single tournament can be coarse and may contain randomness, TourRank conducts R rounds of tournaments in parallel. Each document's total score is computed by summing its scores from all rounds, denoted as

$$\text{score}_{d_i} = \sum_{r=1}^R \text{score}_{d_i}^r \quad (18)$$

where R is the total number of rounds, and $\text{score}_{d_i}^r$ is the score assigned to document d_i in round r . This approach ensures that the final ranking is more fine-grained and robust, as it aggregates the results from multiple independent tournaments.

The tournament structure of TourRank significantly enhances its efficiency and scalability. By allowing parallel processing within and across groups, TourRank reduces the overall ranking

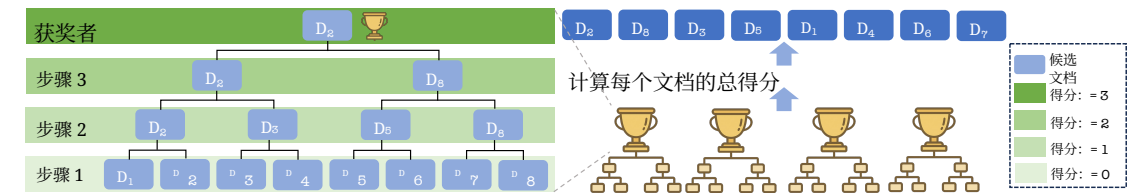


图 7 | : 采用锦标赛策略的排名示意图。

在体育赛事中，通常会有众多参赛者，这就要求能在短时间内尽可能公平地确定他们的排名。在许多体育赛事中，比如国际足联世界杯，往往会采用小组赛的形式。种子选手或队伍会被分配到不同的小组中，各小组内的比赛会同时进行。这种方式能够快速选出排名靠前的参赛者，比如金牌、银牌和铜牌得主。

同样，若将文档排名过程类比为体育赛事，每篇文档便可视作为一名参赛者。通过模拟体育赛事的结构，我们便能快速得出文档的排名结果。具体而言，假设我们需要对 M 篇文档进行排名，基本的锦标赛流程如下：

- **分组与筛选。** 给定一组 M 候选文档后，TourRank首先会将它们划分为若干个较小的组。每组包含的文档数量适中，能够适配大语言模型的上下文窗口容量。例如，若有100篇待排序的文档，可将其分为每组20篇。在每组内，大语言模型会评估文档与查询的相关性，并选出相关性最高的 N 篇文档进入下一阶段。这一过程类似于体育赛事中的小组赛：各小组内的队伍相互竞争，以争夺晋级下一轮的资格。
- **逐步晋级环节。** 完成初步分组与筛选后，各组选出的优胜文档会被汇总起来，再重新划分为新组进入下一轮。这种分层、多阶段的结构会持续迭代下去，每一轮都会减少文档数量并选出最相关的文档。例如，初始的100篇文档被分为10组，每组10篇；第一轮从每组中选出5篇，从而使文档总数缩减至50篇。随后这50篇文档会被分为5组，每组10篇，流程继续重复。每篇文档只要被选入下一阶段，便会获得1个积分。
- **积分系统与多轮比赛。** 积分系统是TourRank的重要组成部分。由于单场赛事产生的分数可能较为粗略且存在随机性，TourRank会同步开展 R 轮赛事。每份文档的总分是通过将其在所有轮次中获得的分数相加得出的，具体表示为：

$$\text{score}_{d_i} = \sum_{r=1}^R \text{score}_{d_i}^r \quad (18)$$

其中 R 代表总轮次数， $\text{score}_{d_i}^r$ 表示文档 d_i 在第 r 轮中获得的分数。采用这种方式能够确保最终排名更加细致且稳健，因为它汇总了多场独立赛事的结果。

TourRank的赛事结构显著提升了其运行效率与可扩展性。通过允许在组内部及组间进行并行处理，TourRank有效降低了整体排名计算耗时。

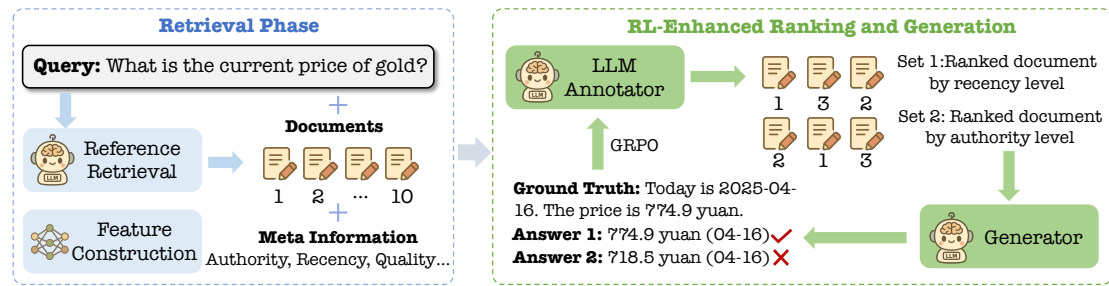


Figure 8 | The illustration of generation reward.

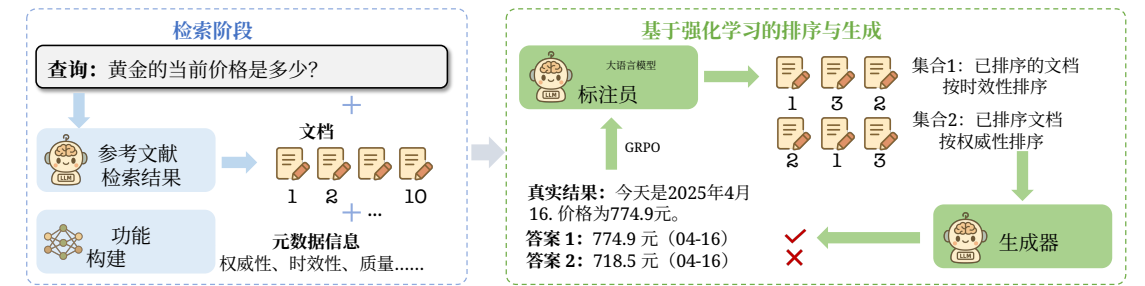


图 8 |：生成奖励示意图。

latency. Specifically, the time complexity of TourRank is $O(K-1)$, where K represents the number of stages in a single tournament. This is a substantial improvement over serial sliding window methods, which have a time complexity of $O(\frac{N-\omega}{s})$. The parallel nature of TourRank makes it particularly suitable for large-scale ranking tasks, where the number of candidate documents can be very large.

4.2.2. Reference Selection

This section highlights the importance of selecting high-quality references in indirectly aligning the LLM preference, ensuring that retrieved documents can enhance the generation of answers. For a complex query, the system retrieves multiple candidate references, which are then incorporated as contextual input to the LLM during the synthesis of an answer. The model's generated response is explicitly linked to supporting references (e.g., "11 of 50 state names are derived from source [1]; Texas is named after a Native American tribe as indicated in source [3], ..."), thereby ensuring that the answer is grounded in verifiable sources. This explicit citation mechanism not only enhances the transparency and trustworthiness of the answer but also facilitates subsequent verification and validation by users and downstream systems. Furthermore, a feedback loop enables iterative refinement of retrieval strategies based on observed LLM usage and outcomes, which in turn optimizes future reference selection for analogous queries. When an answer is generated, the AI search system collects signals related to user satisfaction, answer accuracy, and the relevance of the cited references. These signals are then used to adjust retrieval parameters, ranking functions, or even the candidate source pool in subsequent retrieval cycles. For instance, if the LLM consistently demonstrates a preference for certain types of references (e.g., recent publications, authoritative websites, or data-rich documents), the retrieval system can be adjusted to prioritize these sources for similar future queries.

High-quality reference selection, guided and refined by LLM feedback, not only grounds generated answers in reliable sources but also establishes a virtuous cycle of continuous improvement. This methodology is crucial for building trustworthy, adaptive, and high-performing LLM-based question-answering and information retrieval systems.

4.2.3. Generation Reward

Reference selection indirectly aligns the ranker with LLM preferences, whereas generation rewards achieve this alignment directly. Traditional reference retrieval systems have predominantly relied on supervised learning to optimize the relevance of results. However, the integration of LLMs introduces new opportunities for dynamic and intelligent feedback loops. A promising approach leverages feedback from LLM-generated answers to inform and refine retrieval ranking via reinforcement learning. In this framework, LLM responses serve as direct

具体而言，TourRank的时间复杂度为 $O(K-1)$ ，其中 K 代表单场赛事中的阶段数量。这一数值相比串行滑动窗口方法的 $O(\frac{N-\omega}{s})$ 复杂度有了大幅提升。TourRank的并行特性使其尤其适用于大规模排名任务，这类任务中候选文档的数量往往十分庞大。

4.2.2. 参考文献选择

本节重点阐述了在间接对齐大语言模型偏好时，选取高质量参考文献的重要性，以确保检索到的文档能够助力答案的生成。对于复杂的查询，系统会检索出多个候选参考文献，随后将其作为上下文输入提供给大语言模型，用于答案的合成。模型生成的回答会与对应的参考文献明确关联（例如：“50个州名中有11个源自文献 [1]；得克萨斯州的名称来源于文献 [3]中所提及的某个美洲原住民部落.....”），从而确保答案有可验证的出处作为支撑。这种明确的引用机制不仅提升了答案的透明度和可信度，还便于用户及下游系统后续进行核实与验证。此外，通过反馈循环，系统能够根据观察到的大语言模型使用情况和结果，迭代优化检索策略，进而为同类查询提供更精准的参考文献选择。当答案生成后，人工智能搜索系统会收集与用户满意度、答案准确性以及引用参考文献相关性相关的信号。这些信号随后会被用来调整检索参数、排序函数，甚至后续检索周期中的候选来源池。例如，如果大语言模型始终倾向于某种类型的参考文献（如近期发表的文献、权威网站或数据丰富的文档），检索系统便可被调整为在今后处理类似查询时优先选用这类来源。

由大语言模型反馈引导并优化的高质量参考材料选取机制，不仅能使生成的答案基于可靠的来源，还能建立起持续改进的良性循环。这一方法对于构建可信赖、具备自适应能力且性能卓越的基于大语言模型的问答与信息检索系统而言至关重要。

4.2.3. 生成奖励

参考选择机制能够间接使排序器与大语言模型的偏好保持一致，而生成奖励则能直接实现这种一致性。传统的参考检索系统主要依赖监督学习来优化结果的相关性。不过，大语言模型的引入为构建动态且智能的反馈循环带来了新机遇。一种颇具前景的方法便是利用大语言模型生成的答案所提供的反馈，通过强化学习来指导并优化检索排序。在这一框架中，大语言模型的响应可作为直接的

feedback that guides the optimization of both retrieval and ranking processes.

For each query and its candidate documents, accompanied by metadata such as site name, publication time, and quality score, the ranker applies multiple policies that emphasize different feature combinations (e.g., recency, quality, relevance, and authority). The resulting document rankings are then evaluated using a frozen generator that produces multiple candidate answers. These generated answers are compared with ground truth responses, and a reinforcement reward is assigned to the ranker based on performance. This reward-driven methodology facilitates dynamic policy selection and enhanced document ranking, directly calibrated to generation outcomes. In implementing the reinforcement learning algorithm, the AI search system leverages GRPO. Incorporating generation rewards via reinforcement learning represents a significant advancement toward truly intelligent and adaptive reference retrieval systems. Rather than treating retrieval, ranking, and answer generation as isolated tasks, this integrated feedback loop directly optimizes for both user satisfaction and answer quality.

For example, given a query “What is the current price of gold?”, the AI search system retrieves relevant documents along with their metadata. The ranker may then select a policy, such as $\mathcal{P}_1 : \{recency \succ quality \succ relevance \succ authority\}$, or $\mathcal{P}_2 : \{quality \succ relevance \succ recency \succ authority\}$, to sort these documents accordingly. The sorted results are forwarded to the generator, and feedback from the generator is subsequently used to reward the ranker. This process facilitates document ranking that effectively integrates metadata, thereby optimizing overall retrieval performance.

4.2.4. Distillation of LLM Ranking

Consider a query q and its corresponding final ranking of candidate documents $\{\mathcal{D}_1, \dots, \mathcal{D}_m\}$ obtained from a reasoning-LLM via the methods described above (i.e., *LLM Labeling, Reference Selection, and Generation Reward*). The teacher-generated data are used to construct $m * (m - 1)/2$ document pairs $(\mathcal{D}_i, \mathcal{D}_j)$. Subsequently, an online ranker, which operates as a pointwise ranking LLM, assumes the role of the student model. This student is optimized using the RankNet loss, a pairwise loss function that measures the accuracy of the relative ordering between items, as defined by

$$\mathcal{L} = \sum_{i=1}^n \sum_{j=1}^n \mathbf{1}_{r_i^t < r_j^t} \log(1 + \exp(s_i^s - s_j^s)), \quad (19)$$

where r_i^t and r_j^t denote the teacher’s ranking, and s_i^s and s_j^s represent the corresponding student scores. This distillation process transfers the deep ranking reasoning of the teacher LLM to a more efficient student model suitable for real-time deployment.

4.3. Lightweight System

Modern user queries are increasingly complex, multi-turn, and multifaceted, which poses significant challenges to traditional retrieval systems that are primarily optimized for simple keyword searches. Decomposing such queries into multiple sub-queries, managing iterative interactions, and integrating diverse search intents not only increases computational overhead but also augments system complexity. To overcome these challenges, it is essential to develop a lightweight and efficient retrieval system that leverages the deep comprehension and reasoning capabilities of LLMs. Such a system can streamline system architecture, reduce latency, and effectively manage large-scale, multifaceted queries, thereby enabling faster and more accurate information retrieval in dynamic, real-world scenarios. To achieve this objective, we propose a lightweight system designed to efficiently process massive natural language sub-queries.

反馈信息，用以指导检索与排序过程的优化。

针对每条查询及其对应的候选文档，排序器会结合站点名称、发布时间及质量评分等元数据，应用多种侧重不同特征组合（如时效性、质量、相关性与权威性）的策略。随后，系统利用已训练好的生成模型对生成的文档排名进行评估，该模型会产出多个候选答案。这些生成的答案会与真实参考答案进行对比，再根据表现结果为排序器分配强化奖励。这种基于奖励的机制能够实现动态策略选择并优化文档排名，且排名结果可直接根据生成效果进行调整。在实施强化学习算法时，AI搜索系统采用了GRPO技术。通过强化学习引入生成奖励，标志着向真正智能且自适应的参考检索系统迈出了重要一步。该系统并未将检索、排序与答案生成视为彼此独立的任务，而是通过这一闭环反馈机制直接优化用户满意度与答案质量。

例如，对于查询“当前黄金价格是多少？”，AI搜索系统会先检索出相关文档及其元数据。此时，排序器可能会选用诸如 $\mathcal{P}_1 : \{recency \succ quality \succ relevance \succ authority\}$ 或 $\mathcal{P}_2 : \{quality \succ relevance \succ recency \succ authority\}$ 之类的策略来对文档进行排序。排序后的结果会被送入生成器，而生成器提供的反馈随后会被用来对排序器给予奖励。这一过程能够实现有效整合元数据的文档排序，进而提升整体检索性能。

4.2.4. LLM排序模型的蒸馏

假设有一个查询 q ，以及通过上述方法（即LLM标注、参考项选择和生成奖励）由推理型LLM得到的候选文档最终排序结果 $\{\mathcal{D}_1, \dots, \mathcal{D}_m\}$ 。利用教师模型生成的数据来构建 $m * (m - 1)/2$ 个文档对 $(\mathcal{D}_i, \mathcal{D}_j)$ 。随后，作为点式排序LLM的在线排序器便充当学生模型。该学生模型通过RankNet损失函数进行优化，这是一种用于衡量条目间相对顺序准确性的成对损失函数，其定义如下：

$$\mathcal{L} = \sum_{i=1}^n \sum_{j=1}^n \mathbf{1}_{r_i^t < r_j^t} \log(1 + \exp(s_i^s - s_j^s)), \quad (19)$$

其中 r_i^t 和 r_j^t 表示教师的排序结果， ss_i 与 ss_j 则对应学生模型的得分。这种蒸馏过程将教师LLM的深度排序推理能力迁移到更高效、适合实时部署的学生模型中。

4.3. 轻量级系统

现代用户查询日益复杂、多轮且多样化，这对主要面向简单关键词搜索优化的传统检索系统而言构成了重大挑战。将此类查询拆解为多个子查询、管理迭代式交互以及整合多样的搜索意图，不仅会增加计算开销，还会提升系统复杂度。为克服这些难题，必须开发出一种轻量且高效的检索系统，以充分利用大语言模型所具备的深度理解与推理能力。这样的系统能够简化系统架构、降低延迟，并有效处理大规模、多层面的查询，从而在动态的现实场景中实现更快、更精准的信息检索。为实现这一目标，我们提出了一种专为高效处理海量自然语言子查询而设计的轻量级系统。

Specifically, AI Search paradigm represents a paradigm shift from the traditional hybrid retrieval methods—combining inverted indices with dense retrieval and subsequent PLM-based re-ranking—to a novel framework in which LLMs are directly employed to simplify retrieval and rank relevant documents. This transition not only simplifies the underlying architecture but also fully harnesses the deep contextual reasoning capabilities of LLMs.

4.3.1. Lightweight Retrieval

In the evolution of retrieval systems, various approaches have emerged to enhance accuracy and personalization. Traditional methods initially employed PLMs to generate embeddings for queries and documents, thereby facilitating relevance-based retrieval. However, the increasing demand for a more nuanced understanding and multi-modal integration has spurred the development of advanced retrieval architectures based on LLMs. Fig. 9 provides a visual representation of this progression, outlining the transition from the conventional PLM-based retrieval system to the LLM-driven retrieval system.

PLM-based Retrieval. Early literature on retrieval systems typically employed language models to generate embeddings for both queries and document paragraphs. In these systems, documents are segmented into paragraphs, and each paragraph is represented by aggregated embeddings. Retrieval is then performed by computing the similarity between the query embedding and the embeddings of documents or their paragraphs, with the most relevant results being returned. Although effective for basic textual analysis, this approach is limited in its ability to capture contextual nuances and accommodate personalized representations.

LLM-based Retrieval. With the advent of LLMs, retrieval system architectures have evolved to harness their enhanced language understanding and feature integration capabilities. In the LLM-based retrieval architecture, traditional language models are replaced with an Llama-Tiny-based LLM to generate sophisticated embeddings for both queries and documents. In this scheme, query and document representations are generated by dedicated LLMs: the query LLM incorporates user-specific features alongside the query text, while the document LLM integrates both document content and document-specific attributes. This dual-model approach enables the system to capture personalized and contextual information within a unified embedding space. A key advantage of the LLM-based retrieval approach is its seamless integration of features from multiple sources and modalities. The retrieval architecture is extended to support multi-modal inputs; for example, the document LLM is capable of processing not only textual information but also images and other non-textual data. This results in comprehensive, unified embeddings that more accurately represent the diversity of real-world content.

Overall, the shift toward LLM-based reference retrieval represents a significant milestone in the advancement of intelligent retrieval systems. By leveraging advanced language models and

具体而言，AI搜索范式代表着一种从传统混合检索方法——即结合倒排索引与稠密检索，再进行基于预训练语言模型的重排——向全新框架的转变；在该新框架中，大语言模型被直接用于简化检索流程并对相关文档进行排序。这种转变不仅简化了底层架构，还能充分释放大语言模型所具备的深度上下文推理能力。

4.3.1. 轻量级检索

在检索系统的演进过程中，为了提升检索精度与个性化程度，出现了多种技术路径。传统方法最初利用预训练语言模型来生成查询语句和文档的嵌入向量，从而实现基于相关性的检索。然而，随着对更细致语义理解及多模态信息整合需求的日益增长，基于大语言模型的先进检索架构应运而生。图9直观地展示了这一发展过程，即传统的预训练语言模型检索系统向大语言模型驱动检索系统的转变。

基于预训练语言模型的检索：早期关于检索系统的研究通常借助语言模型为查询语句和文档段落生成嵌入向量。在这类系统中，文档会被切分为多个段落，每个段落由聚合后的嵌入向量来表示。检索时通过计算查询向量与文档或段落嵌入向量之间的相似度，进而返回最相关的结果。尽管该方法在基础文本分析方面表现良好，但在捕捉上下文细微差别以及支持个性化表示方面仍存在局限。

基于LLM的检索。随着大语言模型的出现，检索系统的架构也随之演进，以充分利用其强大的语言理解能力与特征整合能力。在基于LLM的检索架构中，传统的语言模型被基于Llama-Tiny的大语言模型所取代，用于为查询语句和文档生成复杂的嵌入向量。在此方案中，查询语句和文档的表示分别由专用的大语言模型生成：查询LLM会将用户特定特征与查询文本相结合，而文档LLM则会整合文档内容以及文档特有的属性。这种双模型方法使得系统能够在统一的嵌入空间中捕捉个性化及上下文相关信息。基于LLM的检索方法的一大优势在于它能够无缝整合来自多种来源和模态的特征。

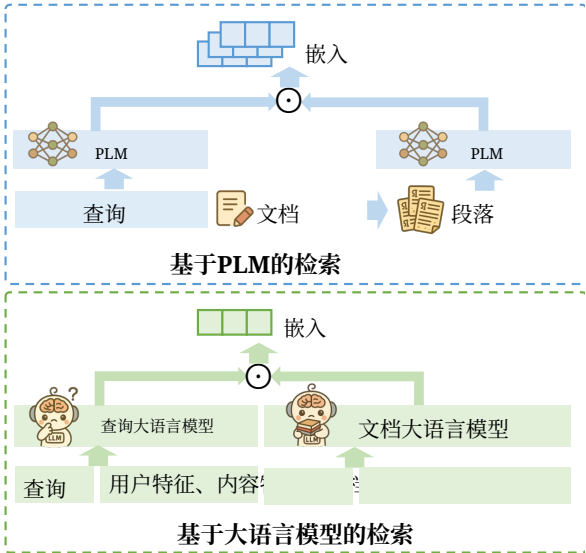


图 9 |：基于PLM的检索与基于LLM的检索之对比示意图。

该检索架构还被扩展以支持多模态输入；例如，文档LLM不仅能处理文本信息，还能处理图像及其他非文本数据。由此生成的统一嵌入向量能够更准确地反映现实世界中内容的多样性。

总体而言，向基于大语言模型的检索方式转变，标志着智能检索系统发展中的重要里程碑。通过运用先进的语言模型以及

integrating multidimensional features, these systems achieve higher retrieval accuracy while delivering a more personalized and comprehensive user experience.

4.3.2. Lightweight Ranking

Building upon advancements in retrieval methods, the ranking component plays a crucial role in refining search results and ensuring that the most relevant documents are delivered to users. Traditionally, ranking strategies relied on PLMs to fuse multiple signals and produce a final relevance score. However, with the rapid development of LLMs, the ranking paradigm has experienced a significant transformation. Fig. 10 elucidates these changes, visually mapping the evolution from conventional PLM-based ranking to the LLM-based ranking.

PLM-based Ranking. In PLM-based ranking systems, traditional end-to-end rankers harness robust PLMs, such as those based on BERT or ERNIE (Li et al., 2023b,c,d,e, 2024b, 2025a,b,c), to encode the primary features of each document, including the query, title, and content. Additional signals such as authority, quality, recency, and relevance features are fused through a neural network fusion layer to generate a final rank score (Li et al., 2025d). This end-to-end framework integrates both semantic and heuristic features, but it is limited by the representational power of the base model.

LLM-based Ranking. In the AI search paradigm, LLM-based rankers use embeddings generated by LLMs to represent the query, title, and content. These semantic embeddings are then combined, again using a neural network fusion layer, with external features such as authority, quality, recency, and relevance features. The use of LLM embeddings enables deeper contextual understanding and more sophisticated feature interactions, leading to more accurate ranking scores. Furthermore, a reasoning LLM is employed directly as the ranking model, processing all available features, including query, title, content, click behavior, recency, authority, quality, and user-specific attributes, in a unified manner to produce the final ranking score. This holistic, end-to-end approach enables effective reasoning over heterogeneous information sources, thereby significantly enhancing overall ranking performance.

The transition from traditional PLM-based rankers to LLM-based ranking architectures represents a paradigm shift in reference retrieval systems. By harnessing the deep semantic comprehension and sophisticated fusion capabilities of large language models, modern ranking systems achieve unprecedented improvements in accuracy and user satisfaction.

4.4. LLM-Augmented Features

Conventional feature engineering relies on handcrafted rules and shallow models to extract key signals, such as authority, recency and relevance. While this approach brought some level of semantic understanding, it suffered from several drawbacks:

整合多维特征，这类系统不仅能实现更高的检索准确率，还能为用户提供更个性化且全面的体验。

4.3.2. 轻量级排序

在检索技术不断进步的背景下，排序环节对于优化搜索结果、确保将最相关的文档呈现给用户起着至关重要的作用。传统上，排序策略依赖预训练语言模型来融合多种信号并生成最终的相关性评分。然而，随着大语言模型的快速发展，排序范式发生了重大变革。图10直观地展示了这一演变过程，从传统的基于预训练语言模型的排序逐步发展为基于大语言模型的排序。

基于PLM的排序。在基于PLM的排序系统中，传统的端到端排序器会利用强大的PLM，例如基于BERT或ERNIE的模型（Li等人，2023b,c,d,e；2024b；2025a,b,c），来编码每份文档的核心特征，包括查询词、标题和内容。诸如权威性、质量、时效性和相关性等额外信号会通过神经网络融合层进行整合，从而生成最终的排序分数（Li等人，2025d）。这种端到端框架能够同时整合语义特征与启发式特征，但其表现能力仍受限于基础模型的代表能力。

基于LLM的排序。在AI搜索范式下，基于LLM的排序器会利用LLM生成的嵌入向量来表示查询词、标题及内容。随后，这些语义嵌入会被进一步组合，同样借助

神经网络融合层会整合权威性、质量、时效性以及相关性等外部特征。借助大语言模型嵌入技术，系统能够更深入地理解上下文信息并实现更复杂的特征交互，从而得出更精准的排序分数。此外，推理型大语言模型可直接作为排序模型，统一处理包括查询词、标题、内容、点击行为、时效性、权威性、质量以及用户专属属性在内的所有可用特征，最终生成排序结果。这种端到端的整体方案能够对异构信息源进行有效推理，进而显著提升整体排序性能。

从传统的基于PLM的排序器向基于大语言模型的排序架构转变，标志着检索系统领域的范式革新。借助大语言模型强大的语义理解能力与复杂特征融合能力，现代排序系统在准确性和用户满意度方面取得了前所未有的提升。

4.4. 大语言模型增强型特征

传统的特征工程依赖人工制定的规则以及浅层模型来提取关键信号，例如权威性、时效性和相关性。虽然这种方法在一定程度上实现了语义理解，但也存在诸多缺陷：

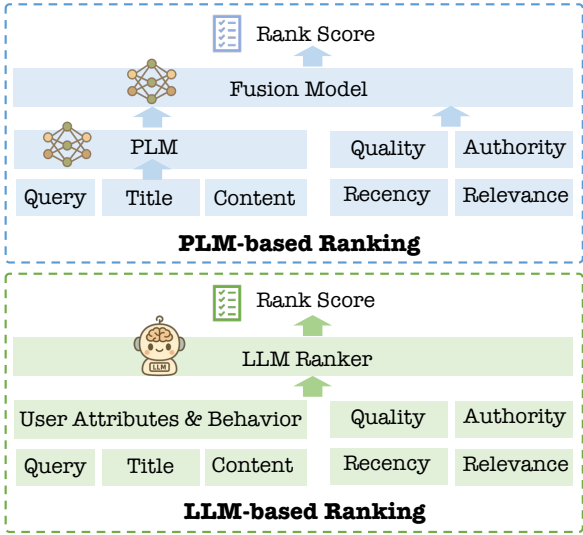


Figure 10 | An illustration of the comparison of PLM-based Ranking and LLM-based Ranking.

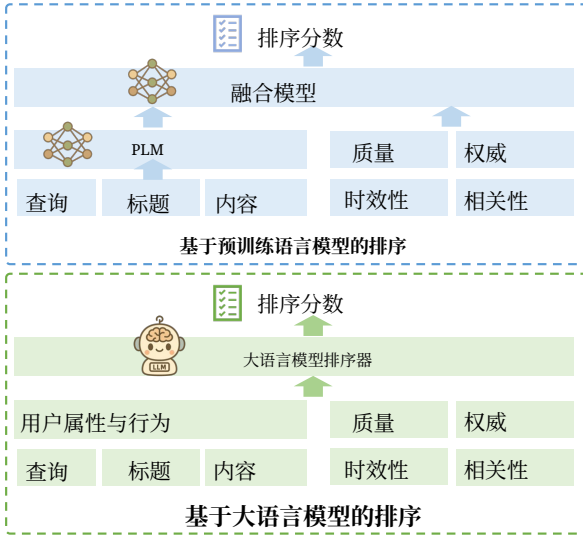


图 10 |：基于PLM的排序与基于LLM的排序对比示意图。

- *Insufficient Coverage*. Limited ability to capture the full breadth of user intent and content diversity.
- *Low Accuracy*. Limited semantic understanding led to suboptimal retrieval results.
- *Difficult Maintenance*. Frequent manual updates and tuning were required as user needs and data evolved.

As LLMs advance in understanding language, context, and multi-modal data, their capability to generate rich semantic representations offers a promising avenue to enhance feature quality. Incorporating LLM-augmented features enables the system to more effectively capture latent user intentions, content quality, and contextual relevance, thereby enhancing retrieval precision. This integration also enables the system to process multimodal information, such as images and videos, thereby expanding the scope and robustness of relevance signals. Ultimately, leveraging LLM-generated features drives the development of more accurate, adaptable, and comprehensive retrieval systems.

Specifically, the AI search system is enhanced with semantic features derived from LLMs. These LLMs directly process both the user query and the document text, yielding richer textual representations and facilitating a more nuanced assessment of factors such as authority, quality, and recency. This approach substantially improves the coverage and accuracy of the retrieval process, as LLMs can infer latent user needs and contextual relevance from the input data. Furthermore, the integration of LLMs with visual language models (VLMs) enables the system to process not only textual information but also multi-modal features, including videos and images. Such multi-modal processing allows the system to capture a broader and more robust set of signals related to authority and quality, thereby enhancing retrieval relevance and user satisfaction. The shift from rule-based and model-based features to those augmented by LLMs and VLMs represents a transformative advancement in reference retrieval systems. By leveraging these powerful language and vision-language models, modern systems achieve broader coverage, higher accuracy, and increased adaptability.

For training with LLM-augmented features, the process begins by mining user session data to capture fine-grained behavioral patterns, including prior queries, user interactions, and content preferences. This comprehensive behavioral dataset, when combined with large-scale pretraining corpora, enables the system to develop a robust understanding of user intent and content relevance. During the training phase, the Search LLM is exposed to hundreds of billions of tokens via next-token prediction and self-supervised learning. This extensive exposure allows the LLM to acquire not only surface-level semantic knowledge but also deeper relational insights, such as authority, recency, and contextual quality, which are central to the LLM-augmented features described above.

By integrating LLM-augmented features into both the training and inference processes, the AI search system achieves significant advancement over traditional retrieval systems. The resulting system is capable of processing and synthesizing multi-modal signals, including text, images, and video, assessing authority and quality from multiple perspectives, and dynamically adapting to user-specific requirements. This comprehensive approach not only enhances the accuracy and robustness of the reference retrieval system but also ensures its scalability and maintainability in real-world, online environments.

- 覆盖范围不足。其捕捉用户意图及内容多样性的能力十分有限。

- 准确率较低。有限的语义理解能力导致了欠佳的检索结果。
- 维护困难。随着用户需求和数据的变化，需要频繁进行手动更新与调优。

随着大语言模型在语言理解、上下文处理以及多模态数据解析方面不断进步，其生成丰富语义表征的能力为提升特征质量提供了广阔前景。引入大语言模型增强的特征后，系统能够更有效地捕捉用户的潜在意图、内容质量及上下文相关性，从而提高检索精度。这种集成还使系统能够处理图像和视频等多模态信息，进而拓展相关性信号的覆盖范围与稳健性。最终，利用大语言模型生成的特征将推动更精准、更具适应性与全面性的检索系统的发展。

具体而言，AI搜索系统通过源自大语言模型的语义特征得到了增强。这些大语言模型直接处理用户查询与文档文本，生成更丰富的文本表征，从而能够更细致地评估权威性、质量和时效性等因素。这种方式显著提升了检索过程的覆盖范围和准确性，因为大语言模型能够从输入数据中推断出用户的潜在需求与上下文相关性。此外，将大语言模型与视觉语言模型相结合，使系统不仅能处理文本信息，还能利用视频和图像等多模态特征。这种多模态处理能力让系统能够获取更广泛且更稳健的权威性与质量相关信号，进而提升检索相关性与用户满意度。从基于规则和模型的特征向由大语言模型及视觉语言模型增强的特征转变，标志着检索系统的重大进步。借助这些强大的语言与视觉语言模型，现代系统实现了更广泛的覆盖、更高的准确性以及更强的适应性。

在利用大语言模型增强的特征进行训练时，首先需挖掘用户会话数据，以捕捉细粒度的行为模式，包括过往查询记录、用户交互行为以及内容偏好。将这一全面的行为数据集与大规模预训练语料库相结合后，系统便能深入理解用户意图及内容相关性。在训练阶段，搜索大语言模型会通过下一个词预测及自监督学习方式，接触数百亿个词元。这种海量的数据输入使大语言模型不仅能掌握表层语义知识，还能获得更深层次的关系性认知，比如权威性、时效性以及上下文质量等，而这些正是上文所述的大语言模型增强特征的核心所在。

通过将大语言模型增强的功能整合到训练和推理过程中，AI搜索系统相较于传统检索系统取得了显著进步。由此形成的系统能够处理和融合文本、图像及视频等多模态信号，从多个维度评估信息的权威性与质量，并能动态适配用户的个性化需求。这种全面的方法不仅提升了参考检索系统的准确性与鲁棒性，还确保了其在现实在线环境中的可扩展性和可维护性。

5. LLM-based Generation

5.1. Background

In the first two stages, AI search paradigm focuses on the design and optimization of the task planner and task executor. Given the execution results and relevant documents retrieved by the Executor, it is critical to design a Writer agent within the AI search system to produce the final, accurate answer. However, the documents retrieved by the Executor frequently contain noise and errors, and addressing these issues to enhance the robustness of the model remains an urgent technical challenge. Furthermore, to provide users with accurate and satisfactory answers, the AI search system is required to align its generated responses with users' preferences. Typically, users prefer that the information provided in the final answer conforms to the **Three H Standard**, *i.e.*, Helpfulness, Harmlessness, and Honesty. Ensuring that the outputs of a RAG system align with user preferences and adhere to the three H standard remains a critical technical challenge that necessitates prompt resolution. Ultimately, the AI search system generates and presents a final answer to the user, which in turn elicits a variety of feedback signals reflective of the user's satisfaction. Explicit signals, such as subsequent query modifications, are readily observable; however, users also provide a wealth of implicit feedback through behaviors like click-through activity and dwell time on the presented results. Effectively leveraging both the explicit and implicit user feedback to refine the AI search system, particularly the Writer, remains a formidable technical challenge.

Building upon the above motivation, we detail the design behind the Writer in the AI search system, focusing on the perspectives of *Robustness*, *RAG Task Alignment*, *Optimization with User Feedback*, and *Multi-Agent Joint Optimization*.

5.2. Robust RAG System

In practical RAG scenarios, the retriever is not always reliable and may retrieve noisy or irrelevant documents. Consequently, these noisy references may misguide the generator, resulting in inaccurate or less robust outputs from the RAG system. To address this issue, the AI search system proposes an adversarial tuning in a multi-agent method *ATM* (Zhu et al., 2024), which leverages a multi-agent system combined with adversarial tuning to enhance the robustness of the generator. An overview of the ATM is illustrated in Fig. 11.

5.2.1. Adversarial Setups for Robust RAG

The ATM system consists of two components: an *Attacker* and a *Generator*. The Attacker aims to introduce perturbations to the documents retrieved by the retriever, while the Generator seeks to withstand these adversarial perturbations and accurately answer the questions.

Specifically, the Attacker perturbs the

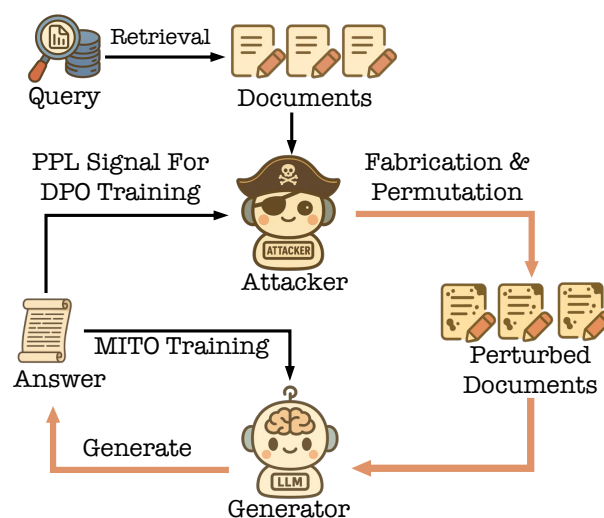


Figure 11 | An illustration of ATM, which leverages a multi-agent system combined with adversarial tuning to enhance the robustness of the generator.

5. 基于大模型的生成技术

5.1. 背景介绍

在前两个阶段，AI搜索范式侧重于任务规划器与任务执行器的设计与优化。鉴于执行器输出的结果及相关文档，在AI搜索系统中设计一名撰写智能体以生成最终且准确的答案至关重要。然而，执行器检索到的文档往往包含噪声与错误，解决这些问题以增强模型的鲁棒性仍是一项紧迫的技术挑战。此外，为了向用户提供准确且令人满意的答案，AI搜索系统必须使其生成的响应符合用户的偏好。通常，用户希望最终答案中所呈现的信息能够遵循三H标准即：有用性、无害性与诚实性。确保RAG系统的输出既符合用户偏好又满足三H标准，仍是一项亟待解决的关键技术难题。最终，AI搜索系统会生成并向用户展示最终答案，进而获取反映用户满意度的各类反馈信号。诸如后续查询修改等显性信号易于观测；不过，用户也会通过点击行为和结果页面上的停留时间等行为提供大量隐性反馈。如何有效利用这些显性和隐性用户反馈来优化AI搜索系统，尤其是撰写智能体，仍是一项极具挑战性的技术课题。

基于上述动机，我们详细阐述了AI搜索系统中Writer模块的设计思路，重点从鲁棒性、RAG任务适配性、基于用户反馈的优化以及多智能体协同优化这几个角度展开说明。

5.2. 鲁棒的RAG系统

在实际的RAG场景中，检索器并不总是可靠的，它可能会返回噪声较大或无关紧要的文档。这些无关的文档可能会误导生成器，从而导致RAG系统产生不准确或鲁棒性较差的输出。为解决这一问题，AI搜索系统提出了一种基于多智能体的对抗性调优方法ATM（Zhu等人，2024），该方法利用多智能体系统并结合对抗性调优技术来增强生成器的鲁棒性。图11展示了ATM的整体架构。

5.2.1. 用于提升RAG鲁棒性的对抗性设置

ATM系统由两个组件构成：攻击者和生成器。攻击者负责对检索器返回的文档施加扰动，而生成器则要抵御这些对抗性扰动并准确回答相关问题。

具体而言，攻击者会篡改

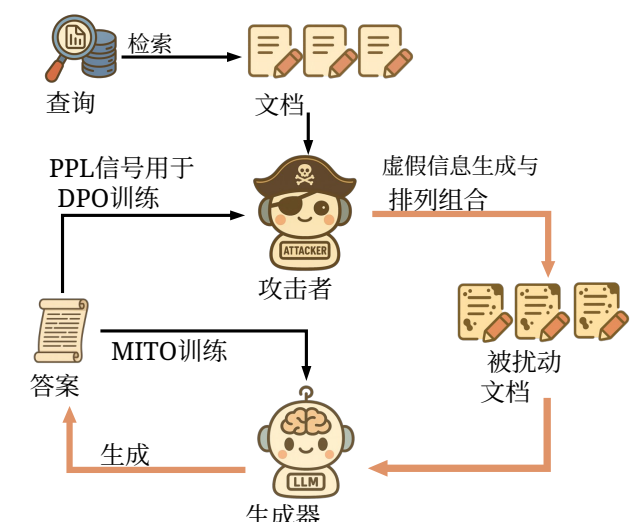


图 11 | 展示了ATM的示意图，该系统利用多智能体框架并结合对抗性调优技术来提升生成器的鲁棒性。

document by fabricating fake knowledge and hiding useful knowledge, which can be divided into the following two steps:

- **Fabrication Generation.** Due to the RAG generator's high sensitivity to noise, the Attacker imitates top-ranked documents by fabricating knowledge that is semantically related to the query but is either useless or misleading, and inserts this fabricated content into the document list.
- **List Permutation.** Because the RAG generator is highly sensitive to the relative position of useful knowledge within the document list, the Attacker could randomly shuffle the document list and duplicate some of its elements. This manipulation brings irrelevant documents to the forefront, effectively hiding the useful knowledge.

Given the question and the list of perturbed documents generated from the Attacker, the Generator takes them as inputs and aims to identify and utilize all useful documents while ignoring noisy ones, thereby maintaining robustness and generating correct answers. The goal of the Generator can be formalized to maximize the objective as

$$G(a | q, D') - \text{dist}[G(a | q, D), G(a | q, D')], \quad (20)$$

where a denotes the ground truth answer, and q represents the question. D and D' refer to the original retrieved documents and perturbed documents from the Attacker, respectively. $G(\cdot)$ signifies the probability that the Generator produces the answer, and $\text{dist}[\cdot]$ quantifies the distance between the two probabilities. Maximizing Eq. 20 implies that the Generator should perform similarly regardless of whether the documents have been perturbed, while still producing the correct answer. In this way, the capability and robustness of the Generator could be enhanced.

5.2.2. Multi-Agent Adversarial Training

The training process for the ATM system comprises two phases: *Initial Training* and *Iterative Adversarial Tuning*.

In the initial training phase, the Generator is trained to acquire fundamental RAG capabilities, providing a better starting point for optimization in the subsequent adversarial tuning. Specifically, the initial training involves four supervised fine-tuning (SFT) tasks: (1) answering the question using the originally retrieved document, (2) answering the question with only one ground-truth document, (3) answering without any document, and (4) extracting the ground-truth document from multiple retrieved documents.

During the iterative adversarial tuning phase, the Attacker and Generator are trained in an alternating manner. The Attacker progressively amplifies the intensity of its adversarial perturbations, while the Generator concurrently refines its generative capabilities to effectively counter these escalating attacks. The training details of the iterative adversarial tuning are introduced as follows.

- For the Attacker, the objective is to synthesize adversarial documents that maximally disrupt the Generator's ability to produce correct answers. Specifically, a fabricated document d' is deemed adversarially effective if it induces high perplexity in the Generator when attempting to generate the correct answer. Formally, given a query q and a fabricated document d' , we quantify the Generator's degradation by computing the perplexity (PPL)

文档内容, 捏造虚假信息。以及隐藏有用的知识, 这一过程可分为以下两个步骤:

- **虚假内容生成.** 由于RAG生成器对噪声极为敏感, 攻击者会模仿排名靠前的文档, 编造出与查询语义相关但毫无用处或具有误导性的知识, 并将这些编造的内容插入到文档列表中。
- **列表重排.** 由于RAG生成器对文档列表中有用知识的相对位置极为敏感, 攻击者可以随机打乱文档列表并重复其中的某些元素。这种操作会将无关文档推至前列, 从而有效隐藏有用的知识。

给定问题以及攻击者生成的扰动文档列表后, 生成器会将其作为输入, 旨在识别并利用所有有用的文档, 同时忽略噪声文档, 以此保持鲁棒性并生成正确答案。生成器的目标可形式化为最大化如下目标函数:

$$G(a | q, D') - \text{dist}[G(a | q, D), G(a | q, D')], \quad (20)$$

其中 a 表示真实答案, q 代表问题。 D 和 D' 分别对应原始检索到的文档以及攻击者生成的扰动文档。 $G(\cdot)$ 表示生成器产生该答案的概率, 而 $\text{dist}[\cdot]$ 则用于量化这两个概率之间的距离。最大化公式20意味着无论文档是否经过扰动, 生成器都应表现出相似的表现, 同时仍能给出正确答案。通过这种方式, 生成器的能力与鲁棒性便能得到提升。

5.2.2. 多智能体对抗训练

ATM系统的训练过程包含两个阶段: 初始训练和迭代式对抗调优。

在初始训练阶段, 生成器被训练以获得基础的RAG能力, 从而为后续对抗调优提供更优的起点。具体而言, 初始训练包含四项监督微调任务: (1) 利用最初检索到的文档回答问题; (2) 仅凭借一份真实文档回答问题; (3) 在不借助任何文档的情况下回答问题; (4) 从多个检索到的文档中筛选出真实文档。

在迭代式对抗调优阶段, 攻击者与生成器以交替方式进行训练。攻击者会逐步增强其对抗性扰动的强度, 而生成器则同步优化自身的生成能力, 以有效应对日益加剧的攻击。迭代式对抗调优的具体训练细节如下所述。

- 对于攻击者而言, 其目标是生成能够最大程度干扰生成器输出正确答案能力的对抗性文档。具体而言, 如果某份伪造文档 d' 在生成器尝试生成正确答案时会导致其困惑度升高, 那么该文档便被视为具有对抗性效果。形式上, 给定一个查询 q 以及一份伪造文档 d' , 我们通过计算困惑度 (PPL) 来衡量生成器性能的下降程度。

of generating the correct answer a as

$$\text{PPL}_G(a | q, \{d'\}) = \exp \left\{ -\frac{1}{T_a} \sum_{t=1}^{T_a} \log P_G(a_t | a_{<t}; q, \{d'\}) \right\}. \quad (21)$$

The Attacker is trained using Direct Preference Optimization (DPO) (Rafailov et al., 2023) with the loss function defined as

$$\mathcal{L}_{\text{DPO}}(A_\theta; A_{\text{ref}}) = -\left[\log \sigma \left(\beta \log \frac{A_\theta(d'_{\text{win}} | q, D)}{A_{\text{ref}}(d'_{\text{win}} | q, D)} - \beta \log \frac{A_\theta(d'_{\text{lose}} | q, D)}{A_{\text{ref}}(d'_{\text{lose}} | q, D)} \right) \right], \quad (22)$$

where A_{ref} is the un-aligned Attacker (reference model), A_θ is the current Attacker to be optimized. d'_{win} and d'_{lose} represent a pair of fabrications generated by the Attacker, win denotes the one with a higher PPL-based reward, and vice versa.

- The Generator aims to produce as many golden answers as possible by leveraging the input documents, while maintaining robustness even when noisy documents are injected. To accomplish this, ATM introduces the Multi-agent Iterative Tuning Optimization (MITO) loss to optimize the Generator as follows:

$$\mathcal{L}_{\text{MITO}} = \mathcal{L}_{\text{SFT}}(a | q, D') + \alpha \mathcal{L}_{\text{KL}}; \quad (23)$$

$$\mathcal{L}_{\text{SFT}}(a | q, D') = -\sum_{t=1}^{T_a} \log P_G(a_t | a_{<t}; q, D'); \quad (24)$$

$$\mathcal{L}_{\text{KL}} = \sum_{t=1}^{T_a} \mathbb{D}_{\text{KL}}[P_G(a_t | a_{<t}; q, D) \| P_G(a_t | a_{<t}; q, D')]. \quad (25)$$

The inclusion of the SFT term enables the Generator to maintain output correctness when processing attacked documents. Additionally, $\mathcal{L}_{\text{KL}}$ serves to reduce the discrepancy in answer generation probabilities between the given normal document list and the attacked document list. α is a pre-set hyper-parameter.

5.3. RAG Task Alignment

To better harness the impressive question-answering capabilities of LLMs within RAG scenarios, it is essential to further align LLMs with the specific requirements of RAG tasks. To address this challenge, this work introduces a preference alignment technique for the RAG scenario (PA-RAG) (Wu et al., 2024), which achieves comprehensive alignment between LLMs and RAG requirements through multi-perspective preference optimization.

5.3.1. RAG Task Requirements

The requirements of RAG tasks can be summarized as follows: (1) *Response informativeness*: the generator must effectively leverage valuable documents to ensure the completeness of the answer. (2) *Response robustness*: the generator should be resilient to interference from noisy documents, maintaining the accuracy and coherence of the responses. (3) *Citation quality*: the claims generated by the generator must be properly cited, ensuring that the answers are traceable to the retrieved documents.

生成正确答案 a 的过程。

$$\text{PPL}_G(a | q, \{d'\}) = \exp \left\{ -\frac{1}{T_a} \sum_{t=1}^{T_a} \log P_G(a_t | a_{<t}; q, \{d'\}) \right\}. \quad (21)$$

攻击者采用直接偏好优化（DPO）方法（Rafailov等人，2023年）进行训练，其损失函数定义如下：

$$\mathcal{L}_{\text{DPO}}(A_\theta; A_{\text{ref}}) = -\left[\log \sigma \left(\beta \log \frac{A_\theta(d'_{\text{win}} | q, D)}{A_{\text{ref}}(d'_{\text{win}} | q, D)} - \beta \log \frac{A_\theta(d'_{\text{lose}} | q, D)}{A_{\text{ref}}(d'_{\text{lose}} | q, D)} \right) \right], \quad (22)$$

其中， A_{ref} 表示未对齐的攻击者模型（参考模型）， A_θ 则表示待优化的当前攻击者模型。 d_{win} 与 d_{lose} 代表由攻击者生成的一对虚构内容， win 表示具有更高基于PPL的奖励值的那个，反之亦然。

- 生成器的目标是通过利用输入文档来产生尽可能多的正确答案，即便在注入噪声文档的情况下也能保持鲁棒性。为实现这一目标，ATM引入了多智能体迭代调优优化（MITO）损失函数来优化生成器，具体方式如下：

$$\mathcal{L}_{\text{MITO}} = \mathcal{L}_{\text{SFT}}(a | q, D') + \alpha \mathcal{L}_{\text{KL}}; \quad (23)$$

$$\mathcal{L}_{\text{SFT}}(a | q, D') = -\sum_{t=1}^{T_a} \log P_G(a_t | a_{<t}; q, D'); \quad (24)$$

$$\mathcal{L}_{\text{KL}} = \sum_{t=1}^{T_a} \mathbb{D}_{\text{KL}}[P_G(a_t | a_{<t}; q, D) \| P_G(a_t | a_{<t}; q, D')]. \quad (25)$$

加入SFT项能够使生成器在处理受攻击的文档时仍保持输出的正确性。此外， $\mathcal{L}_{\text{KL}}$ 用于减小给定正常文档列表与受攻击文档列表之间在答案生成概率上的差异。 α 是一个预设的超参数。

5.3. RAG任务对齐

为了在RAG场景中更好地发挥大语言模型出色的问答能力，必须进一步使大语言模型与RAG任务的具体要求相契合。为应对这一挑战，本研究提出了面向RAG场景的偏好对齐技术（PA-RAG）（Wu等人，2024），该技术通过多视角偏好优化实现了大语言模型与RAG需求之间的全面对齐。

5.3.1. RAG任务要求

RAG任务的要求可归纳如下：(1) 回答的信息丰富度：生成器必须有效利用有价值的文档，以确保答案的完整性。(2) 回答的鲁棒性：生成器应能抵御噪声文档的干扰，从而保持回答的准确性和连贯性。(3) 引用质量：生成器产生的陈述必须有恰当的引用依据，确保答案能够追溯到所检索到的文档。

5.3.2. PA-RAG Optimization Objectives

The optimization goal is to enable the generator to fully utilize valuable documents (Correctness) and accurately cite the references corresponding to claims (Citation quality).

Fully utilizing valuable documents means all answers contained within the documents should be included in the generator's output. Formally, let G represent the generator, x denotes the input, $D = \{d_1, d_2, \dots, d_n\}$ represent the documents retrieved by the retriever, $A = \{a_1, a_2, \dots, a_n\}$ represent the short answers contained in the documents, and y denote the response generated by the generator. The correct response by the generator can be expressed as:

$$\begin{aligned} y &= G(x, D), \\ \text{s.t. } \forall a_i \in A, C(y, a_i) &= \text{True} \end{aligned} \quad (26)$$

where $C(y, a_i) = \text{True}$ implies that the answer a_i is included in y .

Accurate citation means that each claim is fully supported by the cited documents, and irrelevant citations are avoided. Formally, the statement, claim, and citation could be denoted as s , c , and t , respectively. The correct citation of references corresponding to claims can be expressed as follows:

$$y = \{s_1, s_2, \dots, s_n\}, \quad (27)$$

$$s_i = \{\text{"claim": } c_i, \text{"citation": } t_i = [t_{i1}, t_{i2}, \dots, t_{in}]\}, \quad (28)$$

$$\forall s_i \in y, \phi(\text{concat}(t_i), c_i) = 1 \quad (29)$$

$$\forall s_i \in y, \forall t_{ij} \in t_i, \phi(t_{ij}, c_i) = 1 \vee \phi(\text{concat}(t_i \setminus t_{ij}), c_i) = 0 \quad (30)$$

Here, $\text{concat}(t_i)$ denotes the concatenation of all cited documents, and $\phi(t, c) = 1$ indicates that the claim c is fully supported by the citation t .

5.3.3. Methodology of PA-RAG

The training of PA-RAG consists of two phases. The first phase involves instruction fine-tuning to equip the generator with fundamental RAG capabilities. The second phase employs multi-perspective preference optimization to further enhance the generator's response informativeness, response robustness, and citation quality. An overview of the PA-RAG is illustrated in Fig. 12.

Instruction Fine-tuning for Basic RAG Capabilities. The instruction fine-tuning phase focuses on enabling the generator to equip the fundamental abilities to utilize and cite documents. The training data construction contains these three key steps as follows:

- **Construct questions and high-quality documents.** Questions and corresponding short answers are sourced from existing datasets. To construct high-quality documents, the

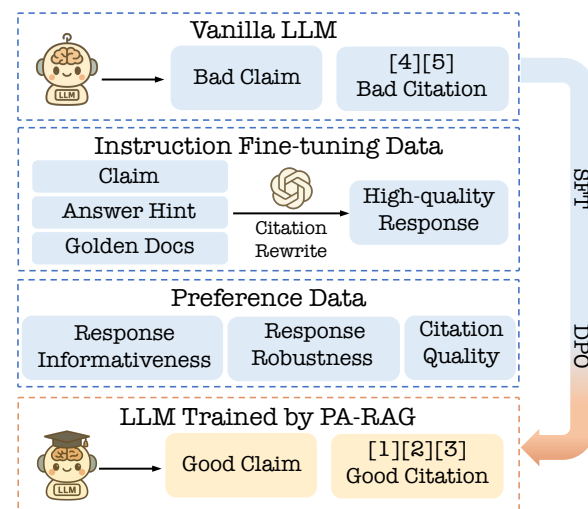


Figure 12 | An illustration of PA-RAG, which achieves comprehensive alignment between LLMs and RAG requirements through multi-perspective preference optimization.

5.3.2. PA-RAG 优化目标

优化的目标是使生成器能够充分利用有价值的文档（即保证正确性），并准确引用与陈述相对应的参考文献（即提升引用质量）。

充分利用有价值的文档意味着文档中包含的所有答案都应被纳入生成器的输出结果中。正式而言，设 G 表示生成器， x 代表输入内容， $D = \{d_1, d_2, \dots, d_n\}$ 表示检索器获取的文档， $A = \{a_1, a_2, \dots, a_n\}$ 表示文档中包含的简短答案，而 y 则表示生成器生成的响应。生成器给出的正确响应可表示为：

$$\begin{aligned} y &= G(x, D), \\ \text{s.t. } \forall a_i \in A, C(y, a_i) &= \text{True} \end{aligned} \quad (26)$$

其中 $C(y, a_i) = \text{True}$ 意味着答案 a_i 被包含在 y 中。

准确的引用意味着每条主张都能得到所引文献的充分支持，且避免无关引用。形式上，陈述、主张和引用可分别表示为 s 、 c 和 t 。与主张相对应的参考文献的正确引用可表述如下：

$$y = \{s_1, s_2, \dots, s_n\}, \quad (27)$$

$$s_i = \{\text{"claim": } c_i, \text{"citation": } t_i = [t_{i1}, t_{i2}, \dots, t_{in}]\}, \quad (28)$$

$$\forall s_i \in y, \phi(\text{concat}(t_i), c_i) = 1 \quad (29)$$

$$\forall s_i \in y, \forall t_{ij} \in t_i, \phi(t_{ij}, c_i) = 1 \vee \phi(\text{concat}(t_i \setminus t_{ij}), c_i) = 0 \quad (30)$$

此处， $\text{concat}(t_i)$ 表示所有被引用文档的拼接，而 $\phi(t, c) = 1$ 则表示即主张 c 得到了所引文献的充分支持——

该文献
为 t_i 。

5.3.3. PA-RAG 的方法论

PA-RAG 的训练分为两个阶段。第一阶段通过指令微调，使生成器具备基础的 RAG 能力；第二阶段采用多视角偏好优化技术，进一步提升生成器的响应信息量、响应鲁棒性以及引用质量。图 12 展示了 PA-RAG 的整体架构。

用于培养基础 RAG 能力的指令微调。 指令微调阶段旨在让生成器掌握利用文档并引用文档的基本能力。训练数据的构建包含以下三个关键步骤：

- **构建问题与高质量文档。** 问题及其对应的简短答案来源于现有数据集。为了构建高质量的文档，

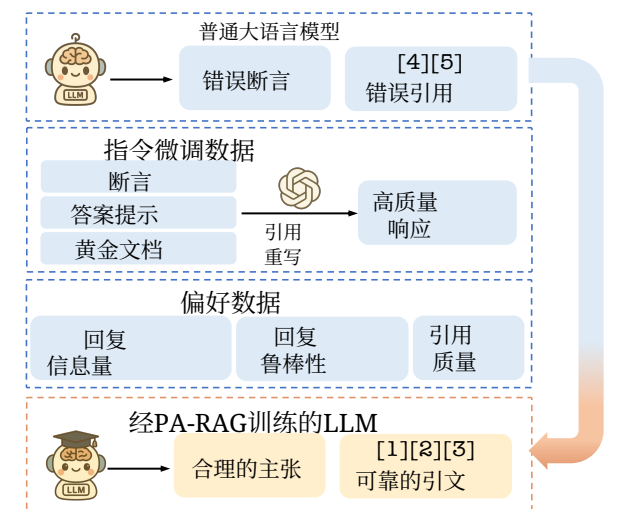


图 12 | PA-RAG 示意图，它通过多视角偏好优化实现了大语言模型与 RAG 需求之间的全面对齐。

RAG retriever first retrieves the top 100 most relevant documents from the corpus for each question, then filters them to retain only those containing the short answers (referred to as “golden documents”). Subsequently, up to five documents are selected as prompt documents, ensuring coverage of all required answers and relevance to the query.

- **Construct high-quality responses.** Responses are generated using GPT-4o, guided by prompts that include instructions, selected documents, and short answer hints. The detailed prompt can be illustrated as follows:

Prompt for GPT-4o

Instruction: Write an accurate, engaging, and concise answer for the given question using only the provided search results (some of which might be irrelevant) and cite them properly. Use an unbiased and journalistic tone. Always cite for any factual claim. When citing several search results, use [1][2][3]. Cite at least one document and at most three documents in each sentence. If multiple documents support the sentence, only cite a minimum sufficient subset of the documents.

Question: {Question}

The final answer should contain the following short answers: {Short answers}

Documents: {Documents}

Answer:

- **Citation rewrite.** While GPT-4o produces coherent answers, direct outputs often exhibit inconsistent or incomplete citations, necessitating further refinement to meet quality standards. To address flawed citations, a three-step validation process is applied:
(1) *Verification:* A Natural Language Inference (NLI) model evaluates whether cited documents logically support the associated claims. (2) *Correction:* If citations fail verification, traverse the power set of all prompt documents as citations to explore a feasible citation scheme. (3) *Simplification:* Redundant or irrelevant citations are pruned from valid subsets to retain only essential references.

Preference Optimization of the Generator through DPO. After the generator acquires the basic ability to utilize and cite documents, it further enhances response informativeness, response robustness, and citation quality sequentially through preference optimization. This phase requires creating preference data containing an input, a superior response (chosen output), and an inferior response (rejected output).

- **Response Informativeness** refers to the completeness of the answer in the response. The optimization objective is to ensure the generator fully uses the document containing the short answer. The construction of preference data includes: (1) *Input Construction:* Like the input for instruction fine-tuning data, the input includes the instruction, the question, and high-quality prompt documents containing all short answers from up to 5 golden documents. (2) *Chosen Output Construction:* The chosen output is consistent with the output component of the instruction fine-tuning data, using GPT-4o and the *citation rewrite* mechanism to construct a response that includes all short answers and the accurate citation. (3) *Rejected Output Construction:* To simulate cases where the generator misses parts of the golden documents, some golden documents are removed from the prompt, and generate inferior answers with the unoptimized generator.
- **Response robustness** refers to the generator’s ability to resist interference. The optimization objective is to enable the generator to avoid disruption from noisy documents. The construction of preference data includes: (1) *Noisy document construction:* Noisy documents are divided into two types. The first type includes documents related to the question

RAG检索器会针对每个问题从语料库中检索出前100篇相关性最高的文档，随后筛选出包含这些简短答案的文档（这类文档被称为“黄金文档”）。之后，系统会从中选取最多五篇文档作为提示文档，以确保涵盖所有所需答案且与查询内容相关。

- **生成高质量的回复。**这些回复由GPT-4o生成，其生成过程受到包含指令、精选文档以及简短答案提示的提示词引导。详细的提示词示例如下：

GPT-4o的提示词

指令：仅依据所提供的搜索结果（其中部分内容可能无关）来撰写准确、有趣且简洁的答案，并正确引用相关来源。表述需保持客观公正，具备新闻稿风格。凡涉及事实性陈述之处均需注明出处。若引用多个搜索结果，请使用 [1][2][3]进行标注；每句话中引用的文档数量应至少为一个，最多不超过三个。如果多个文档能够支撑同一句话的内容，仅需引用最少数量的必要文档即可。

问题： {Question}

最终答案应包含以下简短答案： {Short answers}

文档： {Documents}

答案：

- **引文重写。**尽管GPT-4o能够生成连贯的答案，但其直接输出的引文往往存在不一致或不完整的情况，因此需要进一步调整才能符合质量标准。为解决引文缺陷问题，我们采用了三步验证流程：
(1) 验证：借助自然语言推理模型来判定所引用的文档是否在逻辑上支持相关论断。(2) 修正：若引文未通过验证，则遍历所有提示文档构成的幂集作为候选引文，从而找出可行的引用方案。(3) 简化：从有效的引文子集中剔除冗余或无关的引文，仅保留必要的参考文献。

通过DPO优化生成器的偏好。当生成器掌握了利用和引用文档的基本能力后，便会通过偏好优化依次提升回复的信息量、回复的稳健性以及引文质量。这一阶段需要构建包含输入内容、更优回复（被选输出）及较差回复（被拒输出）的偏好数据。

- **回复信息量**指的是回复中答案的完整程度。优化的目标是确保生成器能充分利用包含简短答案的文档。偏好数据的构建过程如下：(1) 输入构建：与指令微调数据的输入类似，输入内容包含指令、问题以及最多来自5份黄金文档且包含全部简短答案的高质量提示文档。(2) 被选输出构建：被选输出与指令微调数据中的输出部分一致，借助GPT-4o及引文重写机制来生成包含全部简短答案及准确引文的回复。(3) 被拒输出构建：为模拟生成器遗漏部分黄金文档内容的情况，会从提示中移除一些黄金文档，再利用未经优化的生成器来产生较差的答案。
- **响应鲁棒性**指的是生成器抵御干扰的能力。其优化目标是让生成器能够避免受到含噪文档的干扰。偏好数据的构建包括：(1) 含噪文档构建：含噪文档分为两类。第一类是与问题相关的文档，

but lacking the answer; two such documents are randomly selected from the top 100 relevant retrieved documents without short answers. The second type consists of irrelevant documents to the question, with two randomly selected documents without short answers taken from retrieval results of other questions. (2) *Input construction*: The input consists of the instruction, the question, and low-quality prompt documents combining up to five golden documents with four noisy documents. (3) *Chosen output construction*: To simulate the generator ignoring all noisy documents, the chosen output is generated without including these documents, which is the same as the chosen output in the response informativeness part.

- **Citation quality** refers to the generator’s ability to accurately cite relevant documents. The optimization goal is to ensure correct citation of documents related to the claim while avoiding references to irrelevant sources. Data construction involves two steps: first, generating responses and filtering to retain those containing all short answers; second, applying a *citation rewrite* mechanism to identify incorrect citations that fail the NLI model verification or cite irrelevant documents as rejected output and then correct them as chosen output.

The preference optimization stage adheres to a strict optimization sequence, training the model sequentially and independently on response informativeness, response robustness, and citation quality. This sequence is arranged according to the increasing difficulty of preference optimization, similar to curriculum learning (Bengio et al., 2009). Modifying this order can result in performance degradation.

5.4. Optimization with User Feedback

Although LLMs demonstrate powerful capabilities in understanding human instructions and generating high-quality responses by pre-training on extensive corpora and subsequently fine-tuning on meticulously designed instruction-following datasets, it is often observed that fine-tuned LLMs produce unexpected and even harmful responses that significantly deviate from human expectations. To address this, most LLMs incorporate alignment strategies that aim to guide the models to meet predetermined dimensions, such as generating harmless, useful, and honest answers. Typically, these alignment methods (Touvron et al., 2023; Zheng et al., 2023) first assign preference signals to answers generated by various models and then employ techniques, such as RLHF or DPO, to train the models based on these preference signals. However, these approaches usually incur high labor and time costs, as well as significant sampling inference overhead. Moreover, some studies (Bai et al., 2022; Chang et al., 2023) have explored the use of AI-assisted annotation or AI-guided feedback as alignment signals to mitigate the reliance on human effort. Nevertheless, the feedback distributions from these sources generally do not align with online user behaviors and tend to be time-varying and heterogeneous. To tackle the above problems, the AI search system first explores the direct alignment of LLMs with online human behavior and proposes an LLM alignment method Reinforcement Learning with Human Behaviors (*RLHB*). Specifically, RLHB employs a multi-model simultaneous training mechanism with the target LLM as the generator and another auxiliary LLM as the discriminator. They are trained adversarially, ensuring that <query, response, feedback> originates from real online interactions. In inference, the aligned generator accepts the user query along with the most preferred behavior as input signals to generate responses. Compared to RLHF, RLHB eliminates the need for extensive annotation and is adaptable to evolving user behaviors by updating behavior modeling in natural-language form.

The user behavior used in RLHB includes *explicit* and *implicit behavior*. More specifically,

但其中不包含答案；这类文档从前100篇相关检索文档中随机选取两篇不含简短答案的文档。第二类是与问题无关的文档，同样从其他问题的检索结果中随机选取两篇不含简短答案的文档。(2) 输入构建：输入内容由指令、问题以及最多包含五篇优质文档和四篇含噪文档的低质量提示文档组成。(3) 优选输出构建：为了模拟生成器忽略所有含噪文档的情况，优选输出在生成时不会纳入这些文档，该输出与响应信息量部分中的优选输出一致。

- **引用质量**指的是生成器准确引用相关文档的能力。其优化目标是确保对与陈述相关的文档进行正确引用，同时避免引用无关来源。

数据构建包含两步：首先生成回复并筛选，保留包含所有简短答案的回复；其次应用引用重写机制，识别出未通过自然语言推理模型验证或引用无关文档的错误引用，将其作为拒绝输出，随后将其修正为选定输出。

偏好优化阶段遵循严格的优化顺序，依次且独立地对模型进行回复信息量、回复鲁棒性及引用质量方面的训练。这一顺序是根据偏好优化的难度递增程度来安排的，类似于课程学习（Bengio等人，2009）。若更改此顺序，可能会导致性能下降。

5.4. 基于用户反馈的优化

尽管大语言模型通过在海量语料上进行预训练，随后在精心设计的指令跟随数据集上微调，展现出理解人类指令及生成高质量回复的强大能力，但人们常发现微调后的大语言模型会产生出乎意料甚至有害的回复，这些回复与人类期望相去甚远。为解决这一问题，大多数大语言模型采用了对齐策略，旨在引导模型满足预设标准，例如生成无害、有用且诚实的答案。通常，这类对齐方法（Touvron等人，2023；Zheng等人，2023）会先为不同模型生成的答案赋予偏好信号，随后利用RLHF或DPO等技术基于这些偏好信号对模型进行训练。然而，这些方法往往需要耗费大量人力与时间，还会产生显著的自采样推理开销。此外，一些研究（Bai等人，2022；Chang等人，2023）探索了将AI辅助标注或AI引导的反馈用作对齐信号，以减少对人工劳动的依赖。不过，这些来源的反馈分布通常与在线用户行为不符，且往往具有时变性和异质性。为解决上述问题，AI搜索系统率先探索了大语言模型与在线人类行为的直接对齐，提出了一种名为“基于人类行为的强化学习”（*RLHB*）的大语言模型对齐方法。具体而言，RLHB采用多模型同步训练机制，以目标大语言模型作为生成器，另一辅助大语言模型作为判别器，二者通过对抗训练确保 <查询、回复、反馈> 源自真实的在线交互。在推理阶段，经过对齐的生成器会将用户查询以及最偏好的行为作为输入信号，以此生成回复。与RLHF相比，RLHB无需大量标注工作，且能通过更新自然语言形式的行为模型来适应不断变化的用户行为。

RLHB中所使用的用户行为包括显式行为和隐式行为。更具体地说：

explicit behavior refers to users' proactive feedback behavior, such as the click behavior on *Like* and *Dislike* buttons. The implicit behavior usually involves indirect and non-perceived interactions, such as *Page View* (PV) and *Clicks* (i.e., the number of times an answer is clicked). In particular, *Page View*, *Clicks*, and *Like* are positive indicators, and *Dislike* is negative. At the implementation level, these indicators are first smoothed using a logarithmic transformation and then discretized into equal intervals. Subsequently, reward shaping is performed during reinforcement learning by computing $(Like - Dislike) / (PV + Clicks)$, integrating multiple reward signals into a holistic scalar. Next, we present the details of RLHB as follows.

The LLM alignment with online human behaviors can be formulated as a Markov Decision Process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$, transition dynamics $\mathcal{P}$, reward function $\mathcal{R}$ and discount factor γ , denoted as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. The AI search system considers human interactions and behaviors as the environment E . At each time step t , the Writer observes the current state $s_t \in \mathcal{S}$ from E (the query triggered by the customer), and takes actions $a_t \in \mathcal{A}$ according to a policy $\pi : \mathcal{S} \mapsto p(\mathcal{A})$ that maps the states to a probability distribution over the actions. After finishing the generation, the Writer receives a reward $r_t = \mathcal{R}(s_t, a_t)$ from the trained reward model and transitions to a new state $s_{t+1} \in \mathcal{S}$ determined by the environment E . The return of the interactive trajectory $\tau = \{s_1, a_1, \dots, s_T, a_T\}$ is defined as the cumulative γ -discounted rewards: $R(\tau) = \sum_{t=1}^T \gamma^t r_t$, where T denotes the episode horizon. The objective is to optimize the policy π by maximizing the expected returns from the initial state.

To guide the policy toward behavior that aligns with real-world demonstrations, RLHB incorporates a conditional discriminator $D(s_t, a_t; b_t)$ to determine whether a given state-action pair (s_t, a_t) , conditioned on feedback b_t , originates from real online environments. Here, s_t denotes the state (query), a_t denotes the action (response), and b_t represents the behavioral feedback at time t . The discriminator is optimized by maximizing

$$\mathcal{L}_D = \mathbb{E}_{(s_t, a_t, b_t) \in \mathcal{M}_e} [\log(D(s_t, a_t; b_t))] + \mathbb{E}_{(s_t, a_t, b_t) \in \mathcal{M}_g} [1 - \log(D(s_t, a_t; b_t))], \quad (31)$$

where $\mathcal{M}_e$ and $\mathcal{M}_g$ denote the expert and generated demonstration sets, respectively. The policy $\pi(a_t|s_t, b_t)$ is optimized to generate feedback-conditioned responses using the clipped Proximal Policy Optimization (PPO) (Schulman et al., 2017) surrogate objective:

$$\mathcal{L}(\theta) = \mathbb{E}_t [\min(\ell_t \hat{A}_t, \text{clip}(\ell_t, 1 - \epsilon, 1 + \epsilon) \hat{A}_t)], \quad (32)$$

where the importance ratio is defined as

$$\ell_t = \frac{\pi_\theta(a_t|s_t, b_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t, b_t)}. \quad (33)$$

Here, θ represents the current policy parameters, and ϵ is the clipping threshold. The advantage estimates $\hat{A}_t$ are computed using Generalized Advantage Estimation:

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}, \quad (34)$$

with the temporal difference error

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}; b_{t+1}) - V_\phi(s_t; b_t), \quad (35)$$

where γ is the discount factor, λ is the GAE parameter, V_ϕ is the value function with parameters ϕ . The reward signal r_t includes the discriminator output, regularized with a KL divergence penalty as

$$r_t - \eta \text{KL}(\pi_{\theta_{\text{old}}}(a_t|s_t, b_t), \pi_\theta(a_t|s_t, b_t)), \quad (36)$$

显式行为指的是用户的主动反馈行为，例如在点赞和点踩按钮上的点击行为。隐式行为通常涉及间接且不易察觉的交互，比如页面浏览量（PV）以及点击次数（即：某个答案被点击的次数）。其中，页面浏览量、点击次数和点赞数属于正向指标，而点踩数则属于负向指标。在具体实现层面，这些指标会先通过对数变换进行平滑处理，随后被离散化为等间隔的数值。接着，在强化学习过程中通过计算(点赞数 - 点踩数)/(页面浏览量 + 点击次数)来进行奖励塑形，从而将多种奖励信号整合为一个统一的标量值。接下来，我们将详细介绍RLHB的相关内容。

大语言模型与在线人类行为的对齐过程可被建模为一个马尔可夫决策过程（MDP），该过程包含状态空间 $\mathcal{S}$ 、动作空间 $\mathcal{A}$ 、转移动态 $\mathcal{P}$ 、奖励函数 $\mathcal{R}$ 以及折扣因子 γ ，整体表示为一个元组 $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ 。人工智能搜索系统将人类的互动与行为视为环境 E 。在每个时间步 t ，写作模块会从 E （由用户触发的查询）中观测到当前状态 $s_t \in \mathcal{S}$ ，并根据策略 $\pi : \mathcal{S} \mapsto p(\mathcal{A})$ 来选取动作 $a_t \in \mathcal{A}$ ——该策略将状态映射为动作上的概率分布。生成内容完成后，写作模块会从训练好的奖励模型中获得奖励 $r_t = \mathcal{R}(s_t, a_t)$ ，随后根据环境 E 转移到新的状态 $s_{t+1} \in \mathcal{S}$ 。交互轨迹 $\tau = \{s_1, a_1, \dots, s_T, a_T\}$ 的回报被定义为累积的 γ 折损奖励之和： $R(\tau) = \sum_{t=1}^T \gamma^t r_t$ ，其中 T 表示剧集的时长。最终目标是优化策略 π ，从而最大化从初始状态出发的期望回报。

为使策略能够产生与现实世界演示相一致的行为，RLHB引入了一个条件判别器 $D(s_t, a_t; b_t)$ ，用于判断在给定反馈 b_t 的条件下，某一状态-动作对 (s_t, a_t) 是否源自真实的在线环境。此处， s_t 表示状态（查询）， a_t 表示动作（响应），而 b_t 则代表时间 t 时的行为反馈。该判别器通过最大化目标函数来优化。

$$\mathcal{L}_D = \mathbb{E}_{(s_t, a_t, b_t) \in \mathcal{M}_e} [\log(D(s_t, a_t; b_t))] + \mathbb{E}_{(s_t, a_t, b_t) \in \mathcal{M}_g} [1 - \log(D(s_t, a_t; b_t))], \quad (31)$$

其中， $\mathcal{M}_e$ 和 $\mathcal{M}_g$ 分别表示专家演示集与生成的演示集。策略 $\pi(a_t|s_t, b_t)$ 会通过截断的近端策略优化（PPO）(Schulman等人, 2017)替代目标函数来进行优化，以生成受反馈条件约束的响应。

$$\mathcal{L}(\theta) = \mathbb{E}_t [\min(\ell_t \hat{A}_t, \text{clip}(\ell_t, 1 - \epsilon, 1 + \epsilon) \hat{A}_t)], \quad (32)$$

此处，重要性比率的定义如下：

$$\ell_t = \frac{\pi_\theta(a_t|s_t, b_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t, b_t)}. \quad (33)$$

这里， θ 代表当前的策略参数， ϵ 为裁剪阈值。优势估计值 $\hat{A}_t$ 则是通过广义优势估计方法计算得出的。

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}, \quad (34)$$

对应的时序差分误差如下所示：

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}; b_{t+1}) - V_\phi(s_t; b_t), \quad (35)$$

其中， γ 为折扣因子， λ 为GAE参数， V_ϕ 则是带有参数 ϕ 的价值函数。奖励信号 r_t 包含判别器输出，并通过KL散度惩罚项进行正则化。

$$r_t - \eta \text{KL}(\pi_{\theta_{\text{old}}}(a_t|s_t, b_t), \pi_\theta(a_t|s_t, b_t)), \quad (36)$$

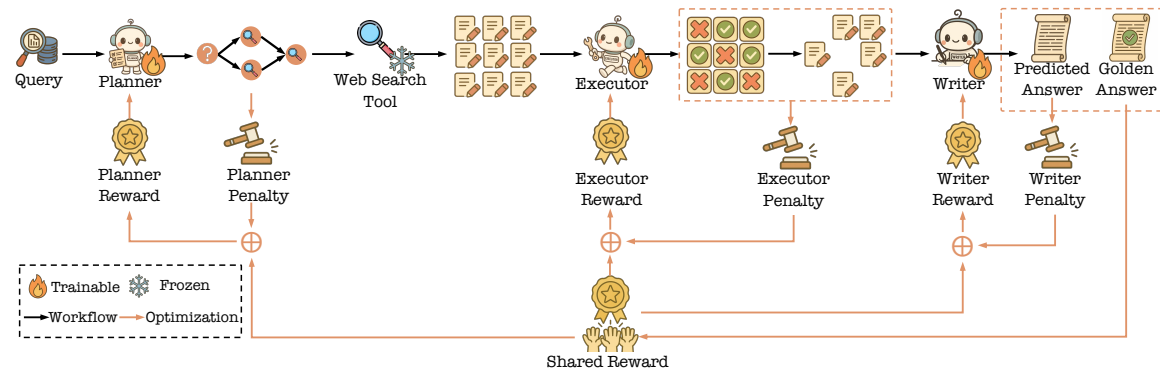


Figure 13 | The illustration of MMOA-RAG, which leverages the Multi-Agent PPO algorithm to align the individual objectives of all modules with the shared goal of maximizing the quality and accuracy of generated answers.

with η as a scaling coefficient. The value function V_ϕ is updated by minimizing the squared error:

$$\mathcal{L}(\phi) = \mathbb{E}_t [\|V_\phi(s_t; b_t) - \hat{R}_t\|^2], \quad (37)$$

where $\hat{R}_t = \hat{A}_t + V_\phi(s_t; b_t)$ represents expected returns. To improve training stability, a fraction κ of high-reward demonstrations is bootstrapped as synthetic negative samples. Additionally, instead of numerical rewards, behavioral feedback is directly introduced as natural language instructions, leveraging the LLM's capability to understand and respond to textual guidance.

5.5. Multi-Agent Joint Optimization

LLMs have revolutionized various applications, including question answering (Asai et al., 2023), information retrieval (Xiong et al., 2024), and various forms of reasoning (Hao et al., 2023); however, their reliance on static pre-training inhibits timely updates to internal knowledge, often resulting in outdated or fabricated outputs (Zhao et al., 2023b). RAG addresses this limitation by integrating current external knowledge into the generation process, thereby mitigating hallucinations and enhancing response accuracy. Contemporary RAG systems are structured as complex pipelines composed of interdependent modules, such as query rewriting (Jiang et al., 2024), first-stage retrieval (Lewis et al., 2020a), re-ranking (Salemi and Zamani, 2024b), document processing (Ke et al., 2024), and answer generation (Shao et al., 2023b), that are traditionally optimized independently via supervised fine-tuning (SFT) on human-annotated data. This modular approach can lead to a misalignment between individual module objectives (e.g., training retrieval modules to maximize metrics like NDCG (Järvelin and Kekäläinen, 2002)) and the overarching goal of generating accurate answers, since improved relevance does not necessarily yield higher answer quality (Cuconasu et al., 2024). Prior efforts (Cuconasu et al., 2024; Guu et al., 2020; Izacard and Grave, 2020; Ke et al., 2024; Lee et al., 2019; Salemi and Zamani, 2024a; Zamani and Bendersky, 2024) to overcome these challenges have employed end-to-end optimization techniques, including the propagation of rewards from the final answer through attention distributions or generation probabilities, as well as reinforcement learning methods like DPO and PPO; however, these strategies are typically confined to simplified two-module pipelines and fail to capture the intricate interdependencies and cooperative dynamics present in more sophisticated multi-module systems.

To address the above limitations, the AI search system proposes a multi-module joint optimization algorithm for RAG (MMOA-RAG) (Chen et al., 2025b). By modeling each intermediate

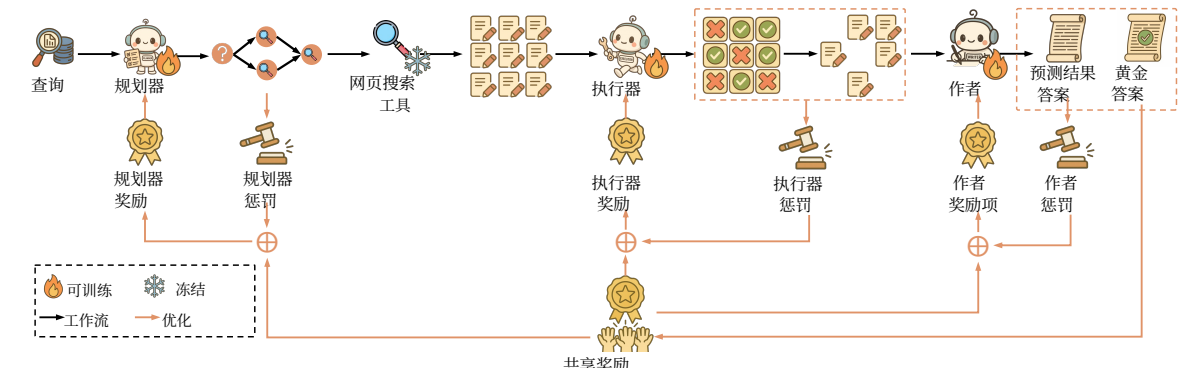


图 13 | 展示了MMOA-RAG的结构示意图，该框架利用多智能体PPO算法来协调各模块的个体目标，以实现提升生成答案的质量与准确性的共同目标。

此处 η 作为缩放系数。价值函数 V_ϕ 的更新是通过最小化平方误差来实现的。

$$\mathcal{L}(\phi) = \mathbb{E}_t [\|V_\phi(s_t; b_t) - \hat{R}_t\|^2], \quad (37)$$

其中， $\hat{R}_t = \hat{A}_t + V_\phi(s_t; b_t)$ 表示预期回报。为提升训练稳定性，会将一部分高奖励样本 κ 作为合成负样本加以利用。此外，并非采用数值型奖励，而是直接引入自然语言指令作为行为反馈，从而充分发挥大语言模型对文本指导的理解与响应能力。

5.5. 多智能体联合优化

大语言模型彻底革新了各类应用场景，包括问答系统 (Asai等人, 2023年)、信息检索 (Xiong等人, 2024年) 以及各种推理任务 (Hao等人, 2023年)。然而，由于它们依赖静态预训练机制，导致内部知识无法及时更新，常常产生过时或虚构的输出内容 (Zhao等人, 2023b年)。RAG技术通过将最新的外部知识融入生成过程，有效缓解了这一问题，从而减少了幻觉现象并提升了回答的准确性。现代RAG系统由多个相互依赖的模块构成复杂的流水线，例如查询重写 (Jiang等人, 2024年)、第一阶段检索 (Lewis等人, 2020a)、结果重排 (Salemi和Zamani, 2024b)、文档处理 (Ke等人, 2024年) 以及答案生成 (Shao等人, 2023b)。这些模块通常借助基于人工标注数据的监督微调技术来独立优化。但这种模块化方式可能导致各模块的目标与生成准确答案这一总体目标之间出现偏差——比如训练检索模块时往往以NDCG等指标为优化目标 (Järvelin和Kekäläinen, 2002年)，而相关性的提升并不必然带来更高的答案质量 (Cuconasu等人, 2024年)。此前已有研究 (Cuconasu等人, 2024年; Guu等人, 2020年; Izacard和Grave, 2020年; Ke等人, 2024年; Lee等人, 2019年; Salemi和Zamani, 2024a年; Zamani和Bendersky, 2024年) 尝试通过端到端优化技术来解决这些难题，比如利用注意力分布或生成概率将最终答案的奖励反向传播，或是采用DPO、PPO等强化学习方法。不过，这类策略通常仅适用于简化的双模块流水线，无法反映更复杂多模块系统中存在的复杂相互依赖关系与协作机制。

为克服上述局限，该AI搜索系统提出了一种面向RAG的多模块联合优化算法 (MMOA-RAG) (Chen等人, 2025b)。通过将该算法中的每个中间模块建模为独立智能体，从而实现优化过程的精细化控制。

module, such as the Planner, *Web Search Tool*, the Executor, and the Writer, as an independent agent, the optimization process is formulated as a multi-agent collaborative RL task. MMOA-RAG leverages the multi-agent PPO (MAPPO) algorithm (Yu et al., 2022) to align the individual objectives of all modules with the shared goal of maximizing the quality and accuracy of generated answers, as evaluated through metrics like the F1 score against ground-truth responses. MMOA-RAG offers notable flexibility in its choice of reward functions and pipeline configurations, allowing it to accommodate diverse system designs while maintaining the cooperative alignment required for efficient optimization. Unlike earlier approaches, which often focus on optimizing simplified one-module or two-module RAG systems or fail to account for the interdependencies among components, MMOA-RAG provides a robust and generalizable framework for optimizing complex, multi-module RAG pipelines. In contrast to methods based on DPO or single-agent PPO, MMOA-RAG excels at fostering effective collaboration among modules, ensuring that the entire system operates cohesively to produce high-quality responses to knowledge-intensive tasks. Next, we will detail the proposed framework through the following algorithmic description.

As shown in Figure 13, MMOA-RAG consists of four components. Specifically, the Planner refines the initial query q , breaking down complex or ambiguous questions into manageable sub-questions $subq$ to improve retrieval efficiency. *Web Search Tool* then gathers relevant documents for each sub-question, producing a candidate document set D . Given the candidate document set, the Executor contained in the Executor then identifies a refined subset D_{selected} that most effectively supports generating the final answer. Using this filtered subset, the Writer produces the predicted response Ans_{predict} to the original query. The Planner, Executor, and Writer are implemented using LLMs, enabling them to operate as RL agents that update their parameters based on reward signals. For efficiency, these modules share a single LLM architecture. As *Web Search Tool* is not modeled as an RL agent due to its unique requirements, it is treated as part of the external environment in reinforcement learning.

The core focus of the MMOA-RAG framework lies in the collaborative optimization of the multiple modules in the pipeline. Its objective is to align the individual optimization goals of each component with the overarching task of producing high-quality responses. To this end, the framework uses a shared reward, R_{shared} , which measures the system's overall performance. A common choice for R_{shared} is the F1 score of the Generator's predicted answer, Ans_{predict} , calculated against the ground-truth response.

Given the inherently cooperative nature of the modules in the RAG pipeline, this shared reward, R_{shared} , serves as a unified signal to guide the optimization of all RL agents. This approach builds on established practices in multi-agent reinforcement learning (MARL) literature (Chen et al., 2022; Rashid et al., 2020; Yu et al., 2022), where shared rewards are leveraged to promote collaboration among agents. To ensure training stability and enhance convergence during the joint optimization process, the framework incorporates penalty mechanisms for individual agents. These penalties refine agent behaviors and discourage inefficient actions described as follows:

- P_{QR} refers to the penalty term for the Planner, designed to prevent the generation of an excessive number of sub-questions.
- P_{S} refers to the penalty term for the Executor, used to avoid redundant or invalid document selections.
- P_{G} refers to the penalty term for the Writer, discouraging the production of lengthy or verbose answers.

这些模块包括规划器、网页搜索工具、执行器和写作器。优化过程被定义为多智能体协作式强化学习任务。MMOA-RAG利用多智能体PPO（MAPPO）算法（Yu等人，2022）来协调各模块的个体目标，使其与最大化生成答案的质量与准确性这一共同目标保持一致；这一目标的评估指标包括对真实答案的F1分数。MMOA-RAG在奖励函数及流程配置的选择上具有显著的灵活性，既能适配各类系统设计，又能维持高效优化所需的协作一致性。与以往仅针对简化版单模块或双模块RAG系统进行优化、或未考虑组件间相互依赖关系的方法不同，MMOA-RAG为复杂的多模块RAG流程提供了一个稳健且具备泛化能力的优化框架。相较于基于DPO或单智能体PPO的方法，MMOA-RAG更擅长促进各模块间的有效协作，确保整个系统能够协同运作，从而高质量地完成知识密集型任务。接下来，我们将通过后续的算法描述来详细阐述所提出的框架。

如图13所示，MMOA-RAG由四个组件构成。具体而言，规划器会对初始查询 q 进行细化处理，将复杂或模糊的问题拆分成易于处理的子问题 $subq$ ，以此提升检索效率。网页搜索工具随后会针对每个子问题收集相关文档，从而生成候选文档集 D 。在获得候选文档集后，执行器会从其中筛选出最能有效辅助生成最终答案的精炼子集 D 。借助这一筛选后的子集，撰写器便会针对原始查询生成预测响应 Ans 。规划器、执行器和撰写器均基于大语言模型实现，因此能够作为强化学习智能体运行，并根据奖励信号更新自身参数。为提高效率，这些模块共享同一套大语言模型架构。由于网页搜索工具具有独特的使用要求，无法被建模为强化学习智能体，因此在强化学习中被视作外部环境的一部分。

MMOA-RAG框架的核心在于对流程中多个模块进行协同优化。其目标是让各组件的独立优化目标与生成高质量响应的整体任务保持一致。为此，该框架采用了共享奖励 R_{shared} 来衡量系统的整体表现。 R_{shared} 的常见选择是生成器所预测的答案 Ans_{predict} 与真实答案之间的F1分数。

鉴于RAG流程中各模块本身具有天然的协作特性，这种共享奖励 R_{shared} 便成为指导所有强化学习智能体优化的统一信号。该方法借鉴了多智能体强化学习领域的成熟做法（Chen等人，2022年；Rashid等人，2020年；Yu等人，2022年），这些研究均利用共享奖励来促进智能体之间的协作。为确保训练过程的稳定性并提升联合优化阶段的收敛效果，该框架为各智能体设置了惩罚机制。这些惩罚机制能够规范智能体的行为，抑制低效动作的产生，具体说明如下：

- P_{QR} 指的是规划器的惩罚项，其作用是防止生成数量过多的子问题。
- P_{S} 指的是执行器的惩罚项，用于避免出现冗余或无效的文件选择情况。
- P_{G} 代表针对生成器的惩罚项，用于抑制其产生冗长或啰嗦的回答。

The interplay of the shared reward R_{shared} and these penalty terms P_{QR} , P_{S} , and P_{G} ensures coordinated optimization of all modules, promoting synergy and effectiveness across the pipeline.

Detailed Configuration. MMOA-RAG models three key modules, the Planner (QR), the Executor (S), and the Writer (G), as RL agents. These agents, defined within the tuple $\langle \mathcal{G}, O, A, R \rangle$, operate collaboratively to maximize the shared reward R_{shared} . $\mathcal{G} = \{\text{QR}, \text{S}, \text{G}\}$ is the set of agents. For each agent $i \in \mathcal{G}$, O_i denotes its observation, A_i represents its action space, and R_i is its reward.

Concretely, for the Planner, the observation space $O_{\text{QR}} = \{\text{Prompt}_{\text{QR}}, q\}$ comprises a guiding prompt $\text{Prompt}_{\text{QR}}$ and the initial query q , while the action space A_{QR} corresponds to the LLM’s vocabulary space. The reward function R_{QR} combines the shared reward R_{shared} with a penalty term P_{QR} , designed to discourage generating an excessive number of sub-questions. More specifically, $P_{\text{QR}} = -0.5$ when the number of sub-questions exceeds 4, and is 0 otherwise. The reward of QR can be represented as follows:

$$R_{\text{QR}} = R_{\text{shared}} + P_{\text{QR}}, \quad (38)$$

Moreover, the observation of the Executor can be represented as $O_{\text{S}} = \{\text{Prompts}, q, D\}$, where Prompts are guiding instructions, q is the query, and D is a set of K candidate documents. Its action space $A_{\text{S}} = \{"0", "1", \dots, "K-1", "Document", ", "\}$ is constrained to document selection, reducing the exploration space for efficient training. The reward function R_{S} incorporates the penalty P_{S} , which is set to -1 when the selected document IDs are invalid (e.g., duplicates or incorrect formats) and 0 otherwise. R_{S} can be represented as follows:

$$R_{\text{S}} = R_{\text{shared}} + P_{\text{S}}, \quad (39)$$

Similarly, the observation of the Writer can be represented as $O_{\text{G}} = \{\text{Prompt}_{\text{G}}, q, D_{\text{selected}}\}$, where Prompt_{G} guides the generation process, q is the query, and D_{selected} is the filtered set of documents provided by the Executor. The Writer shares the same action space as the Planner, $A_{\text{G}} = A_{\text{QR}}$. Its reward function can be formulated as

$$R_{\text{G}} = R_{\text{shared}} + P_{\text{G}}, \quad (40)$$

where R_{G} introduces a penalty term P_{G} , which discourages excessively verbose answers by setting $P_{\text{G}} = -0.5$ when the generated content exceeds a predefined length, and 0 otherwise. Overall, these components ensure that all agents are aligned with the overarching goal of generating high-quality answers while maintaining computational efficiency and training stability.

Warm Start with SFT. To facilitate joint optimization across multiple modules in MMOA-RAG, it is essential to perform SFT as a warm start for the trainable modules: the Planner, Executor, and Writer. The purpose of this warm start is to improve the modules’ adherence to task-specific instructions, reduce the exploration space during MARL, and enhance the efficiency of exploration and exploitation during training.

Specifically, for the Planner, public datasets from Rewrite-Retrieve-Read pipelines (Ma et al., 2023) are utilized for SFT by training the module to reformulate an input query q into a series of sub-questions $\text{subq} = \{q_{\text{subq}_1}, q_{\text{subq}_2}, \dots\}$, making retrieval more effective.

As for the Executor, its task is to identify a subset of candidate documents D_{selected} from a given pool $D = \{d_1, d_2, \dots, d_K\}$ that are most relevant to answering q . To construct the Executor’s SFT dataset, a heuristic approach is employed. First, the stop words and punctuation are removed from q_i , retaining only its essential keywords to form a filtered query set Set_{q_i} . These

共享奖励 R_{shared} 与这些惩罚项 P_{QR} 、 P_{S} 以及 P_{G} 之间的相互作用，能够确保所有模块实现协同优化，进而提升整个流程中的协同效应与整体效率。

详细配置。MMOA-RAG将三个关键模块——规划器（QR）、执行器（S）和书写器（G）——建模为强化学习智能体。这些智能体定义在元组 $\mathcal{G}, O, A, R$ 中，通过协同工作来最大化共享奖励 R_{shared} 。 $\mathcal{G} = \{\text{QR}, \text{S}, \text{G}\}$ 即为智能体集合。对于每一个智能体 $i \in \mathcal{G}$ ， O_i 表示其观测值， A_i 代表其动作空间，而 R_i 则是其获得的奖励。

具体而言，对于规划器而言，其观测空间 $O_{\text{QR}} = \{\text{Prompt}_{\text{QR}}, q\}$ 包含引导性提示 $\text{Prompt}_{\text{QR}}$ 以及初始查询 q ，而其动作空间 A_{QR} 则对应着大语言模型的词汇空间。奖励函数 R_{QR} 将共享奖励 R_{shared} 与一个惩罚项 P_{QR} 相结合，该惩罚项旨在抑制生成过多子问题的行为。更确切地说，当子问题数量超过4个时， P_{QR} 取值= -0.5 ；否则取值为0。QR 的奖励可表示如下：

$$R_{\text{QR}} = R_{\text{shared}} + P_{\text{QR}}, \quad (38)$$

此外，执行器的观测结果可表示为 $O_{\text{S}} = \{\text{Prompts}, q, D\}$ ，其中 Prompts 代表引导指令， q 为查询内容， D 则是一组 K 候选文档。其动作空间 $A_{\text{S}} = \{"0", "1", \dots, "K-1", "Document", ", "\}$ 被限定为文档选择操作，从而缩小探索范围以实现高效训练。奖励函数 R_{S} 包含了惩罚项 P_{S} ：当所选文档 ID 无效（例如存在重复或格式错误）时，该惩罚项取值为 -1 ，否则为 0。 R_{S} 的具体形式如下：

$$R_{\text{S}} = R_{\text{shared}} + P_{\text{S}}, \quad (39)$$

类似地，写作器的观测结果可表示为 $O_{\text{G}} = \{\text{Prompt}_{\text{G}}, q, D_{\text{selected}}\}$ ，其中 Prompt_{G} 用于指导生成过程， q 为查询内容， D_{selected} 则是执行器提供的经过筛选的文档集合。写作器与规划器共享相同的动作空间 $A_{\text{G}} = A_{\text{QR}}$ 。其奖励函数可表述为：

$$R_{\text{G}} = R_{\text{shared}} + P_{\text{G}}, \quad (40)$$

其中 R_{G} 引入了惩罚项 P_{G} ，用于抑制过于冗长的回答：当生成内容超出预设长度时， $P_{\text{G}} = -0.5$ 会被设为惩罚值，否则该惩罚项取值为 0。总体而言，这些组件确保了所有智能体都能与生成高质量答案这一总体目标保持一致，同时兼顾计算效率与训练稳定性。

通过SFT实现热启动。为了在MMOA-RAG中促进多个模块间的联合优化，必须对可训练模块——规划器、执行器和写作器进行SFT以作为热启动。这种热启动的目的在于提升这些模块对特定任务指令的遵循程度，缩小多智能体强化学习过程中的探索空间，并提高训练期间的探索与利用效率。

具体而言，针对规划器而言，会利用Rewrite-Retrieve-Read流程中的公开数据集（Ma等人，2023）来进行SFT；该过程旨在训练该模块将输入查询 q 重构为一系列子问题： $\text{subq} = \{q_{\text{subq}_1}, q_{\text{subq}_2}, \dots\}$ ），从而提升检索效果。

至于执行器，其任务是从给定的候选文档池 $D = \{d_1, d_2, \dots, d_K\}$ 中筛选出与回答 q 最相关的子集 D 。为了构建执行器的监督微调数据集，采用了一种启发式方法：首先移除 q_i 中的停用词和标点符号，仅保留其核心关键词，从而形成过滤后的查询集 Set_{q_i} 。

keywords are then compared against the content of each candidate document $d_{i,j}$ of q_i to compute a subset $\text{Set}_{d_{i,j}}$. The document $d_{i,j}$ is included in D_{selected} if $\text{Set}_{q_i} \cap \text{Set}_{d_{i,j}} \neq \emptyset$, providing the ground-truth labels for training.

Meanwhile, the Writer is fine-tuned to synthesize the final answer $\text{Ans}_{\text{predict}}$ based on D_{selected} and the query q , using $\text{Ans}_{\text{golden}}$ as the ground-truth output. The supervised loss for the Writer, as well as for other modules, can be formulated as

$$L_{\text{SFT}}(\theta) = -\frac{1}{N} \sum_{n=1}^N \log P(Y_i | X_i; \theta), \quad (41)$$

where N is the number of training samples, X_i represents the input (e.g., q and D_{selected}), Y_i is the ground-truth output (e.g., $\text{Ans}_{\text{golden}}$), and θ are the model parameters. By leveraging these carefully constructed SFT datasets for each module, MMOA-RAG ensures effective initialization, enabling the modules to work in synergy during joint multi-agent optimization.

Multi-Agent Optimization. Following the warming start with SFT, the LLM is better equipped to follow precise task-specific instructions, enabling improved performance across three agents. To further enhance system performance, MMOA-RAG employs a fully cooperative MARL strategy based on MAPPO. In this setup, the modules collectively optimize their policies under a shared global reward, R_{shared} , reflecting the overall quality of answers produced by the system (e.g., F1 score of $\text{Ans}_{\text{predict}}$). The modules, the Planner (QR), the Executor (S), and the Writer (G), share parameters through a common LLM backbone, effectively reducing computational overhead while promoting efficient collaboration between agents. The optimization involves three key models: the Actor model (with parameters θ), which generates outputs like subq , D_{selected} , and $\text{Ans}_{\text{predict}}$ based on observations O_i ; the Critic model (with parameters ϕ), which estimates the state-value function V_ϕ for stability in optimization; and the pre-trained SFT model (with parameters θ_{SFT}), which serves as the baseline reference. The overall loss function integrates the Actor and Critic losses as

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{Actor}}(\theta) + \alpha \cdot \mathcal{L}_{\text{Critic}}(\phi), \quad (42)$$

where α is a hyperparameter balancing the two terms. The Actor loss $\mathcal{L}_{\text{Actor}}$ incorporates advantage estimates $\hat{A}_t^i$, which is computed using Generalized Advantage Estimation (GAE) (Schulman et al., 2015), and the importance sampling ratio r_t^i to perform stable policy updates:

$$\mathcal{L}_{\text{Actor}}(\theta) = \sum_i \sum_t \min [r_t^i \hat{A}_{\pi_\theta}^{i,t}, \text{clip}(r_t^i, 1 - \epsilon, 1 + \epsilon) \hat{A}_{\pi_\theta}^{i,t}], \quad (43)$$

where $r_t^i = \pi_\theta(a_t^i | s_t^i) / \pi_{\theta_{\text{old}}}(a_t^i | s_t^i)$. Meanwhile, the Critic loss $\mathcal{L}_{\text{Critic}}$ minimizes the state-value prediction error as:

$$\mathcal{L}_{\text{Critic}}(\phi) = \sum_i \sum_t \max \left[(\Delta V_{i,t})^2, \left(\text{clip} \left(V_\phi^{i,t}, V_{\phi_{\text{old}}}^{i,t} \pm \epsilon \right) - V_{\text{target}}^{i,t} \right)^2 \right], \quad (44)$$

Here, $\Delta V_{i,t} = V_\phi^{i,t} - V_{\text{target}}^{i,t}$, where $V_\phi^{i,t} = V_\phi(s_t^i)$. The term $V_{\text{target}}^{i,t}$ represents the cumulative return and s_t^i is the state-value function.

The reward function $R(s_t^i, a_t^i)$ combines R_{shared} , agent-specific penalties P_i , and a regularization term penalizing deviations from the SFT-informed policy as:

$$R(s_t^i, a_t^i) = \begin{cases} R_{\text{shared}} + P_i - \beta \log \frac{\pi_\theta(\text{Answer}_i | O_i)}{\pi_{\theta_{\text{SFT}}}(\text{Answer}_i | O_i)}, & \text{if } t = T, \\ 0, & \text{otherwise.} \end{cases} \quad (45)$$

随后, 将这些关键词与每份候选文档 $d_{i,j}$ 的内容 q_i 进行对比, 以此计算出子集 $\text{Set}_{d_{i,j}}$. 若某文档 $d_{i,j}$ 满足 $\text{Set}_{q_i} \cap \text{Set}_{d_{i,j}} \neq \emptyset$ 的条件, 就会被纳入 D 所选的子集之中, 这些即为训练所需的真实标签。

同时, **Writer**模型会经过微调, 以便根据 D 选定的内容以及查询 q 来生成最终答案 $\text{Ans}_{\text{predict}}$, 并以 $\text{Ans}_{\text{golden}}$ 作为基准输出。**Writer**模型以及其他模块的监督损失可表述为:

$$L_{\text{SFT}}(\theta) = -\frac{1}{N} \sum_{n=1}^N \log P(Y_i | X_i; \theta), \quad (41)$$

其中 N 表示训练样本的数量, X_i 代表输入 (例如 q 和 D 选定的内容), Y_i 为基准输出 (例如 $\text{Ans}_{\text{golden}}$), θ 则是模型参数。通过利用为各模块精心构建的SFT数据集, MMOA-RAG能够实现有效的初始化, 从而让各模块在多智能体联合优化过程中协同工作。

多智能体优化。在完成基于SFT的预热阶段后, 大语言模型能更好地遵循特定的任务指令, 进而提升三个智能体的性能。为了提高系统性能, MMOA-RAG采用了基于MAPPO的全协作式多智能体强化学习策略。在此框架下, 各模块会在共享全局奖励 R 的驱动下共同优化自身策略, 该奖励反映了系统生成答案的整体质量 (例如 $\text{Ans}_{\text{predict}}$ 的F1分数)。规划器 (QR)、执行器 (S) 以及**Writer** (G) 这些模块通过共用同一个大语言模型骨干网络来共享参数, 这不仅有效降低了计算开销, 还促进了智能体之间的高效协作。优化过程涉及三个关键模型: **Actor**模型 (参数为 θ), 它基于观测结果 O_i 生成诸如 subq 、 D 选定内容以及 $\text{Ans}_{\text{predict}}$ 等输出; **Critic**模型 (参数为 ϕ), 它负责估算状态价值函数 V_ϕ 以保障优化的稳定性; 还有预训练的SFT模型 (参数为 θ_{SFT}), 它作为基准参考。整体损失函数整合了**Actor**模型和**Critic**模型的损失项:

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{Actor}}(\theta) + \alpha \cdot \mathcal{L}_{\text{Critic}}(\phi), \quad (42)$$

其中, α 是一个用于平衡这两项的超参数。**Actor**损失 $\mathcal{L}_{\text{Actor}}$ 包含了优势估计值 $\hat{A}_t^i$, 该值是通过广义优势估计 (GAE) (Schulman等人, 2015年) 计算得出的; 同时, 重要性采样比率 r_t^i 被用于实现稳定的策略更新:

$$\mathcal{L}_{\text{Actor}}(\theta) = \sum_i \sum_t \min [r_t^i \hat{A}_{\pi_\theta}^{i,t}, \text{clip}(r_t^i, 1 - \epsilon, 1 + \epsilon) \hat{A}_{\pi_\theta}^{i,t}], \quad (43)$$

此处有 $r_t^i = \pi_\theta(a_t^i | s_t^i) / \pi_{\theta_{\text{old}}}(a_t^i | s_t^i)$ 。与此同时, **Critic**损失 $\mathcal{L}_{\text{Critic}}$ 旨在最小化状态价值预测误差:

$$\mathcal{L}_{\text{Critic}}(\phi) = \sum_i \sum_t \max \left[(\Delta V_{i,t})^2, \left(\text{clip} \left(V_\phi^{i,t}, V_{\phi_{\text{old}}}^{i,t} \pm \epsilon \right) - V_{\text{target}}^{i,t} \right)^2 \right], \quad (44)$$

这里, $\Delta V_{i,t} = V_\phi^{i,t} - V_{\text{target}}^{i,t}$, 其中 $V_\phi^{i,t} = V_\phi(s_t^i)$ 。项 $V_\phi^{i,t}$ 目标值代表累积回报, 而 s_t^i 则是状态价值函数。

奖励函数 $R(s_t^i, a_t^i)$ 结合了 R 共享的、针对特定智能体的惩罚项 P_i , 以及用于惩罚偏离SFT引导策略程度的正则化项, 其表达式如下:

$$R(s_t^i, a_t^i) = \begin{cases} R_{\text{shared}} + P_i - \beta \log \frac{\pi_\theta(\text{Answer}_i | O_i)}{\pi_{\theta_{\text{SFT}}}(\text{Answer}_i | O_i)}, & \text{if } t = T, \\ 0, & \text{otherwise.} \end{cases} \quad (45)$$

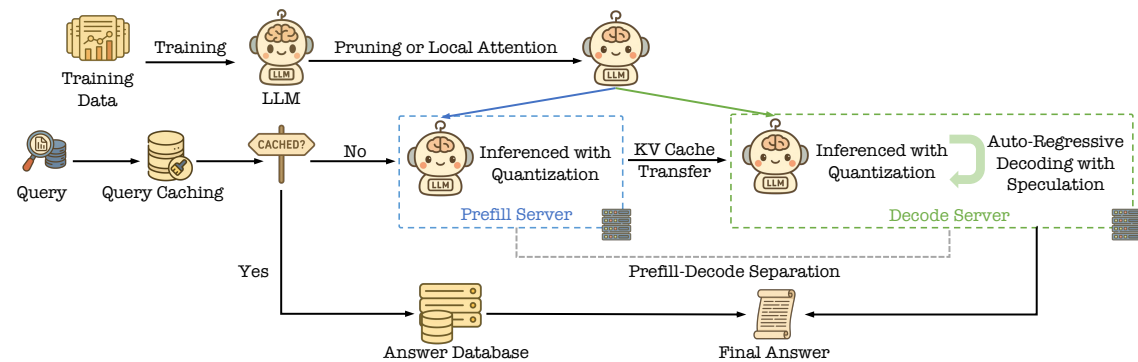


Figure 14 | The technical pipeline of lightning LLM's generation.

The entire training process involves collecting rollouts for each agent, estimating advantages with GAE, and updating the Actor and Critic parameters by optimizing $\mathcal{L}(\theta, \phi)$. Additionally, to speed up the entire training process, we can perform a minibatch in the rollout process. Finally, we can obtain a well-trained Actor model used for the next inference and evaluation.

6. Light-Weighting LLM

6.1. Background

AI-powered search applications have experienced a substantial increase in user traffic. For example, prominent platforms such as Perplexity are reported to manage over 10 million daily visits (Perplexity, 2023). The core technology driving these applications consists of LLMs, which predominantly utilize the Transformer architecture (Vaswani et al., 2017). Although the Transformer model has demonstrated notable effectiveness, its computational complexity, which scales quadratically with the input sequence length, introduces significant challenges during inference. Moreover, in adherence to scaling laws (Kaplan et al., 2020), LLMs have evolved to encompass an exceptionally large number of parameters. For instance, models such as LLaMA-4, DeepSeek-V3, and Qwen-3 possess parameter counts ranging from 100 billion to 1,000 billion, thereby requiring considerable storage capacity.

The computational requirements of LLMs necessitate considerable hardware resources, and their inference latency constitutes a significant portion of the overall response time in AI search applications. For example, a state-of-the-art LLM for AI search, DeepSeek-R1 (DeepSeek-AI et al., 2025), which reportedly incurs high deployment costs (requiring 32 GPUs, each with 80 GiB memory per instance) and suffers from latency bottlenecks (a first token latency of 2.91 seconds and a decoding speed of approximately 30 tokens/second). Search applications are inherently latency-sensitive, as users typically have limited tolerance for waiting times.

Consequently, optimizing LLM inference has emerged as a critical imperative to reduce latency, minimize operational costs, and enhance overall system efficiency. This section examines lightweighting methodologies that have gained widespread adoption in both industrial and academic settings due to their demonstrated effectiveness, feasibility of deployment, cost efficiency, and ability to deliver measurable performance improvements beyond theoretical gains. Lightweighting techniques for LLMs in AI search can be systematically categorized into two principal approaches: (1) algorithmic-level optimizations, which focus on reducing model parameter counts and computational complexity, and (2) infrastructure-level optimizations, which enhance inference efficiency through improved computational workflows and resource

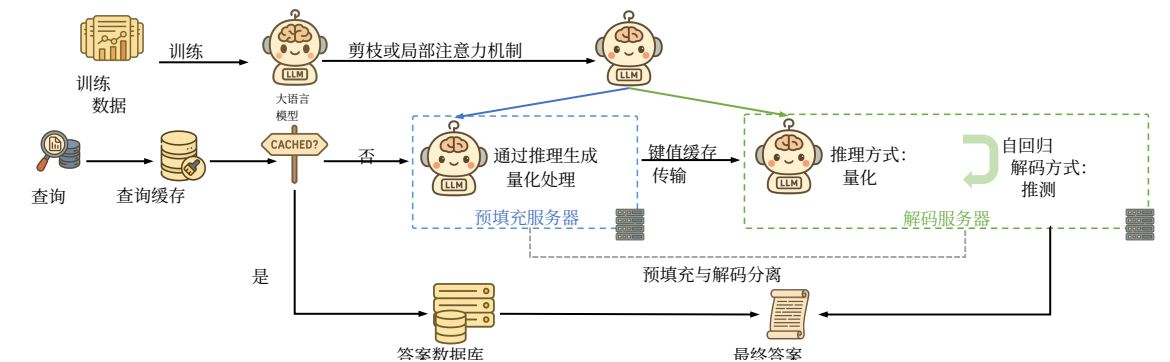


图 14 | : Lightning LLM的生成技术流程。

整个训练过程包括为各智能体收集轨迹数据、利用GAE估算优势值，并通过优化 $\mathcal{L}(\theta, \phi)$ 来更新Actor与Critic参数。此外，为了加快训练进程，我们可以在轨迹生成阶段采用小批量处理方式。最终，我们便能得到经过充分训练的Actor模型，用于后续的推理与评估。

6. 轻量化大语言模型

6.1. 背景

由人工智能驱动搜索应用，其用户流量呈现出显著增长态势。例如，据称像Perplexity这样的知名平台每日访问量便超过了1000万次（Perplexity, 2023年）。支撑这类应用的核心技术便是大语言模型，它们主要采用Transformer架构（Vaswani等人，2017年）。尽管Transformer模型展现出了卓越的效能，但其计算复杂度会随着输入序列长度的增加而呈平方级上升，这给推理过程带来了巨大挑战。此外，遵循缩放定律（Kaplan等人，2020年），大语言模型的参数规模也日益庞大。比如LLaMA-4、DeepSeek-V3和Qwen-3等模型，其参数数量介于1000亿到1万亿之间，因此亟需巨大的存储容量。

大语言模型对计算资源的需求十分庞大，其推理延迟在人工智能搜索应用的总响应时间中占据了很大比例。以用于人工智能搜索的最先进大语言模型DeepSeek-R1为例（DeepSeek-AI等人，2025年），该模型的部署成本极高（需要32个GPU，每个实例配备80 GiB内存），且存在延迟瓶颈（首个词元生成延迟达2.91秒，解码速度约为每秒30个词元）。搜索应用对延迟极为敏感，因为用户对等待时间的容忍度通常很低。

因此，优化大语言模型推理过程已成为降低延迟、减少运营成本并提升系统整体效率的关键任务。本节将探讨那些因成效显著、易于部署、成本低廉且能带来超出理论预期的实际性能提升而被工业界和学术界广泛采用的轻量化方法。在人工智能搜索领域，针对大语言模型的轻量化技术可系统划分为两大主要方向：（1）算法层面的优化，旨在减少模型参数数量与计算复杂度；（2）基础设施层面的优化，通过改进计算流程与资源利用方式来提升推理效率。

utilization strategies.

6.2. Algorithmic-Level Lightweighting

Algorithmic-level lightweighting primarily targets the reduction of an LLM’s parameter count and inherent computational complexity. Reducing the number of parameters directly lessens storage requirements, while decreasing computational load alleviates the demand on computing resources. In the following paragraphs, we introduce two highly effective and low-deployment-cost techniques for reducing both computational and parameter load: local attention and pruning.

6.2.1. Local Attention

The internal computations of LLMs are predominantly characterized by matrix multiplications involving input context tokens and model parameter matrices. A prominent strategy to mitigate this computational burden is to restrict the scope of input tokens attended to by each token, a technique termed **local attention**. It is well-established that the computational complexity of the standard full attention mechanism exhibits quadratic scaling ($O(N^2)$) with respect to the input sequence length N . However, a substantial body of research (Xiao et al., 2023b) suggests that full attention is not always necessary, as only a subset of contextual tokens typically exerts a significant influence on the model’s output. Consequently, local attention mechanisms can often serve as an effective substitute for full attention.

While various lightweight attention mechanisms have been proposed, such as linear attention (MiniMax et al., 2025), RWKV (Peng et al., 2023), and Mamba (Gu and Dao, 2024), their adoption typically requires training models *de novo*, incurring substantial costs. In contrast, local attention techniques can often be integrated into pre-trained full-attention models with minimal overhead. This adaptation can be accomplished either through lightweight fine-tuning (Zhang et al., 2025b) or, in some cases, applied directly at inference time without any retraining, as demonstrated by methods like StreamingLLM (Xiao et al., 2023b).

Recent works have explored various strategies to adapt pre-trained models. For instance, LightTransfer (Zhang et al., 2025b) addresses the high computational cost of full attention by identifying and modifying “lazy layers” where attention patterns are highly localized. Its innovation lies in selectively replacing full attention in these layers with a more efficient mechanism like streaming attention. This approach significantly reduces the KV cache footprint and computational load while preserving long-context capabilities, and can be applied either training-free or with minimal fine-tuning.

More advanced approaches aim to linearize pre-trained Transformers to achieve linear-time inference. A key challenge in this conversion is integrating recurrent mechanisms, such as gates, without costly retraining. Liger (Lan et al., 2025) tackles this problem by repurposing existing weights from the pre-trained model’s key projection to construct these gating mechanisms, thereby avoiding new parameters. Its hybrid attention, combining local sliding window attention with a linear recurrent model, effectively balances local context processing with long-range dependency modeling.

Similarly, LoLCATs (Zhang et al., 2025a) focuses on converting standard softmax attention into a subquadratic alternative, with a strong emphasis on scalability to extremely large models. The core innovation is a two-stage framework: an “Attention Transfer” step first trains a subquadratic mechanism to mimic the original attention, followed by a “Low-rank Linearizing” step that uses LoRA to correct residual errors. This parameter-efficient method has successfully

利用策略。

6.2. 算法层面的轻量化

算法层面的轻量化主要针对减少大语言模型的参数数量及其固有的计算复杂度。参数数量的减少可直接降低存储需求，而计算负荷的减轻则能缓解对计算资源的压力。在接下来的段落中，我们将介绍两种极为有效且部署成本低的、能够同时降低计算量与参数规模的技术：局部注意力机制与剪枝。

6.2.1. 局部注意力机制

大语言模型的内部计算主要以输入上下文词元与模型参数矩阵之间的矩阵乘法为主。为减轻这一计算负担，一种常用策略是限制每个词元可关注的输入词元范围，该技术被称为**局部注意力机制**。众所周知，标准全注意力机制的计算复杂度会随着输入序列长度 N 的增加而呈平方级增长($O(N^2)$)。然而大量研究 (Xiao等人, 2023b) 表明，全注意力机制并非总是必需的，因为通常只有部分上下文词元会对模型输出产生显著影响。因此，局部注意力机制往往能够有效替代全注意力机制。

尽管已经提出了多种轻量级注意力机制，例如线性注意力机制 (MiniMax等人, 2025)、RWKV (Peng等人, 2023) 以及Mamba (Gu和Dao, 2024)，但采用这些机制通常需要从头开始训练模型*de novo*，成本极高。相比之下，局部注意力技术往往能够以极小的开销集成到已预训练的全注意力模型中。这种适配既可以通过轻量级微调来实现 (Zhang等人, 2025b)，在某些情况下甚至无需重新训练，就能在推理阶段直接应用，StreamingLLM (Xiao等人, 2023b) 等方法便证明了这一点。

近期的研究探讨了多种适配预训练模型的策略。例如，LightTransfer (Zhang等人, 2025b) 通过识别并修改那些注意力模式高度局部化的“惰性层”，来解决全注意力机制带来的高计算成本问题。其创新之处在于有选择地用流式注意力这类更高效的机制来替换这些层中的全注意力模块。这种方法在保留长上下文处理能力的同时，能显著减少键值缓存占用的空间及计算量，且既可在无需训练的情况下应用，也可仅需极少量的微调。

更先进的方法旨在将预训练的Transformer模型线性化，以实现线性时间的推理。此类转换中的一个关键挑战是如何集成门控等循环机制，同时避免昂贵的重新训练过程。Liger (Lan等人, 2025) 通过复用预训练模型中键投影层的现有权重来构建这些门控机制，从而无需新增参数。其混合注意力机制结合了局部滑动窗口注意力与线性循环模型，能够有效平衡局部上下文处理与长距离依赖建模的需求。

同样，LoLCATs (Zhang等人, 2025a) 致力于将标准的softmax注意力机制转化为次二次复杂度的替代方案，尤其注重其在超大规模模型中的可扩展性。其核心创新在于采用两阶段框架：首先通过“注意力迁移”步骤训练一种次二次机制来模拟原始的注意力过程；随后利用“低秩线性化”步骤借助LoRA来修正残留误差。这种参数效率极高的方法已经取得了显著成果。

linearized models as large as 405B parameters, achieving up to a 3× throughput increase while maintaining performance comparable to the original Transformer.

Furthermore, models adapted with local attention can leverage existing, highly optimized attention libraries, such as FlashAttention (Dao et al., 2022), during deployment, thereby ensuring computational efficiency in practice.

6.2.2. Model Pruning

Pruning offers another powerful strategy for model lightweighting by directly removing redundant parameters from a model’s weight matrices. This process reduces both the model’s storage footprint and its computational demands during inference. Pruning techniques can be broadly categorized into unstructured, semi-structured, and structured pruning. While unstructured pruning removes individual weights, leading to sparse models that often require specialized hardware or libraries for acceleration, *structured pruning* removes entire structural components, such as attention heads, neuron columns, or even entire layers. This results in a smaller, dense model that can be directly accelerated on standard hardware, making it highly practical for deployment. Similar to local attention, many advanced structured pruning methods can be applied to pre-trained LLMs with minimal fine-tuning or even in a training-free manner, offering a cost-effective path to model compression.

Recent works have developed various innovative approaches to structured pruning. For instance, Layer Collapse (LaCo) (Yang et al., 2024) introduces a training-free method that directly removes entire layers from a pre-trained model. Its core innovation is the *Reserving-Differences-while-Seeking-Common (RDSC) Layer Merge* principle, which calculates the parameter differences between adjacent layers and merges this differential into a subsequent layer. This “collapses” layers, effectively pruning them while preserving the model’s overall function. A key advantage of LaCo is its ability to remove 30% to 50% of layers while maintaining performance and, crucially, preserving the model’s internal architecture. This allows the pruned model to serve as a drop-in replacement in existing systems without code modifications.

Other methods focus on more granular structures. The CoFi (Coarse-to-Fine) pruning method (Xia et al., 2022) demonstrates that structured pruning can achieve compression and speedup comparable to distillation but at a much lower computational cost. CoFi’s innovation lies in simultaneously pruning both coarse-grained structures (e.g., layers) and fine-grained units (e.g., attention heads, hidden dimensions) using masks of varying granularity. To maintain accuracy, it integrates a dynamic layerwise distillation method that learns the optimal layer mapping between the teacher and the pruned student model, achieving over 95% sparsity and 10× speedups while retaining high accuracy.

Building on classic compression frameworks, SlimGPT (Ling et al., 2024) adapts the Optimal Brain Surgeon (OBS) technique for the unique challenges of structured pruning in LLMs. To overcome the limitations of OBS, SlimGPT introduces a *Batched Greedy Pruning* algorithm that enables low-cost and rapid structured pruning by processing groups of parameters. It employs a grouped Cholesky decomposition for attention heads and a dynamic group size for FFNs to achieve near-optimal pruning efficiently. A key advantage of SlimGPT is its accessibility: it is a task-agnostic method requiring only a small calibration dataset and can complete the entire pruning process in about an hour on a single GPU.

Another line of work targets semi-structured pruning, which is critical for hardware acceleration but can often degrade model accuracy. A novel plug-and-play post-training pruning (PTP) method (Zhang et al., 2024) addresses this by introducing two key components. First, a

该方法成功应用于参数量高达405B的线性化模型，在保持与原始Transformer相当的性能水平的同时，实现了高达 3× 的吞吐量提升。

此外，采用局部注意力机制调整的模型在部署时能够利用现有的、经过高度优化的注意力库，例如FlashAttention（Dao等人，2022年），从而确保实际应用中的计算效率。

6.2.2. 模型剪枝

剪枝是另一种实现模型轻量化的有效策略，其原理是直接从模型的权重矩阵中移除冗余参数。这一过程既能减小模型的存储占用，又能降低推理时的计算需求。剪枝技术大致可分为非结构化剪枝、半结构化剪枝和结构化剪枝三类。非结构化剪枝会删除单个权重，生成的稀疏模型往往需要借助专用硬件或库才能高效运行；而结构化剪枝则是移除整个结构组件，比如注意力头、神经元列甚至整个层。这样得到的模型体积更小且为密集型，可直接在标准硬件上加速运行，因此非常适合实际部署。与局部注意力机制类似，许多先进的结构化剪枝方法只需对预训练的大语言模型进行少量微调，甚至无需微调即可应用，为模型压缩提供了一种成本低廉的解决方案。

近期的研究提出了多种用于结构化剪枝的创新方法。例如，Layer Collapse (LaCo)（Yang等人，2024年）提出了一种无需训练的方法，可直接从预训练模型中移除整个层。其核心创新在于保留差异以寻求共性（RDSC）的层合并原理，该原理通过计算相邻层之间的参数差异，并将这种差异合并到后续层中。这种方式能够“合并”各层，从而在实现剪枝的同时保留模型的整体功能。LaCo的一个关键优势在于：它能在保持性能的前提下移除30%至50%的层，且尤为重要，能完整保留模型的内部架构。这使得剪枝后的模型可直接替换现有系统中的模型，无需修改代码。

其他方法则侧重于更细粒度的结构。CoFi（粗到细）剪枝方法（Xia等人，2022年）表明，结构化剪枝所能实现的压缩与加速效果可与知识蒸馏相媲美，但计算成本却低得多。CoFi的创新之处在于它同时剪枝粗粒度结构（如层）和细粒度单元（如注意力头、隐藏维度），并使用不同粒度的掩码来实现这一目的。为保证准确性，该方法采用了动态逐层蒸馏技术，用以学习教师模型与剪枝后的学生模型之间的最优层映射关系，最终实现了超过95%的稀疏度以及10×加速效果，同时仍能保持较高的准确率。

基于经典的压缩框架，SlimGPT（Ling等人，2024年）将最优脑外科医生（OBS）技术应用用于大语言模型中结构化剪枝所面临的独特挑战。为克服OBS的局限性，SlimGPT提出了一种批量贪心剪枝算法，该算法通过批量处理参数来实现低成本且快速的结构化剪枝。它针对注意力头采用分组乔列斯基分解方法，针对前馈神经网络则采用动态分组大小策略，从而高效实现近乎最优的剪枝效果。SlimGPT的一大优势在于其易用性：这是一种与任务无关的方法，仅需少量校准数据集即可运行，且在单张GPU上大约一小时就能完成整个剪枝过程。

另一类研究方向聚焦于半结构化剪枝技术，该技术对硬件加速至关重要，但往往会导致模型精度下降。Zhang等人（2024年）提出了一种新颖的即插即用式训练后剪枝（PTP）方法，该方法包含两个关键组件。首先，

new metric, Relative Importance and Activation (RIA), jointly considers weight magnitude and activation data to prevent the inadvertent removal of important weight channels, a drawback of previous metrics. Second, it employs an efficient channel permutation algorithm that reorganizes input channels to maximally preserve important weights before enforcing the N:M sparsity pattern. This training-free approach is scalable to models over 70B parameters and makes sparse LLM inference more practical by converting models to a hardware-friendly N:M format with minimal performance loss.

6.3. Infrastructure-Level Lightweighting

Infrastructure-level lightweighting encompasses a diverse range of techniques aimed at optimizing the computational and memory efficiency of large-scale AI systems, particularly LLMs. These methods can be broadly categorized based on their applicability: some exploit the unique characteristics inherent in specific applications like AI search, while others offer general-purpose improvements to the underlying LLM inference pipeline. This section will delve into several key strategies. Specifically, we will first examine optimizations tailored to the distinct operational patterns of AI search applications, including those that leverage *input-output similarity*, address *output length reduction*, and capitalize on *query semantic similarity*. Subsequently, we will discuss more general infrastructure enhancements applicable to a broader range of LLM use cases, such as *prefill-decode separation*, *quantization*, and *speculative decoding*, which target fundamental aspects of the inference process.

6.3.1. Output Length Reduction

A further distinguishing characteristic is the pronounced asymmetry in length between user input queries and LLM-generated outputs, where the latter are typically substantially longer. For instance, a concise query such as “Explain the history of Hawaii” may elicit an extensive response.

Numerous methods aim to mitigate the high inference costs associated with excessively long outputs. A primary strategy involves reducing the length of the model’s generated responses. This can be achieved through several approaches:

- *Prompt-based Length Reduction.* Some techniques leverage prompting to curtail output length. For example, TALE-EP (Han et al., 2024) estimates token budgets by first prompting the LLM to determine a reasonable token limit for a given task, then incorporating this estimate into a constraint prompt that guides the model to generate concise yet accurate responses. Chain-of-Draft (CoD) (Xu et al., 2025) addresses the verbosity common in step-by-step reasoning by instructing models to “Think step by step, but only keep a minimum draft for each thinking step, with at most five words,” thereby preserving essential intermediate reasoning while significantly reducing token usage. Lee et al. (Lee et al., 2025) conducted a systematic study of different compression instructions, revealing a universal trade-off between reasoning length and accuracy, and proposing the concept of “token complexity” as the minimum number of tokens required for successful problem-solving.
- *Training-based Length Reduction.* Other methods achieve output length reduction through modifications during the training phase. Demystifying (Yeo et al., 2025) implements a Cosine Reward function based on Dirichlet principles, incorporating an “exceed length penalty” to control Chain-of-Thought (CoT) reasoning length while maintaining performance. Aggarwal et al. (Aggarwal and Welleck, 2025) modify the training data by incorpo-

一种名为“相对重要性及激活度”（RIA）的新指标，能够同时考量权重大小与激活数据，从而避免重要权重通道被误删——这正是以往指标存在的缺陷。其次，该方法采用高效的通道重排算法，在强制实施N:M稀疏模式之前，先重新组织输入通道以最大程度保留重要权重。这种无需额外训练的方法可应用于参数规模超过700亿的模型，它能将模型转换为适合硬件处理的N:M格式，且性能损失极小，从而让稀疏大语言模型的推理应用更具实用性。

6.3. 基础设施层面的轻量化

基础设施层面的轻量化涵盖了一系列旨在优化大规模人工智能系统，尤其是大语言模型的计算与内存效率的技术。这些技术可根据其适用范围大致分类：有些技术利用特定应用场景（如人工智能搜索）所固有的特性，而另一些则能为底层的大语言模型推理流程提供通用性改进。本节将深入探讨几种关键策略。具体而言，我们首先会剖析针对人工智能搜索应用独特运行模式的优化措施，包括那些利用输入输出相似性、致力于实现输出长度缩减以及借助查询语义相似性的优化方法。随后，我们会讨论适用于更广泛大语言模型用例的通用基础设施增强技术，例如针对推理过程关键环节的预填充-解码分离、量化以及投机性解码。

6.3.1. 输出长度缩减

另一个显著特点是用户输入查询与大语言模型生成的输出之间存在着明显的长度差异，后者通常要长得多。例如，像“请解释夏威夷的历史”这样简短的查询，可能会引出篇幅较长的回答。

大量方法旨在降低因输出过长而产生的高推理成本。主要策略之一是缩短模型生成的回复长度。可通过以下几种途径实现这一目标：

- 基于提示词的长度缩减。部分技术利用提示词来压缩输出长度。例如，TALE-EP（Han等人，2024年）会先通过提示词让大语言模型为特定任务确定合理的词元数量上限，再将这一估算值纳入约束性提示词中，从而引导模型生成简洁且准确的回复。Chain-of-Draft（CoD）（Xu等人，2025年）针对逐步推理中常见的冗长问题，要求模型“逐步思考，但每个思考步骤仅保留最少草稿内容，且不超过五个词”，如此既能保留关键的中间推理过程，又能大幅减少词元使用量。Lee等人（Lee等人，2025年）对不同压缩指令进行了系统性研究，发现了推理长度与准确性之间的普遍权衡关系，并提出了“词元复杂度”这一概念，即成功解决问题所需的最少词元数量。

- 正在解决中。
- 基于训练的长度缩减方法。其他方法则是在训练阶段通过相应调整来实现输出长度的缩减。Demystifying（Yeo等人，2025年）采用了基于狄利克雷原理的余弦奖励函数，并引入了“超出长度惩罚”机制，以此在保持性能的同时控制思维链推理的长度。Aggarwal等人（Aggarwal和Welleck，2025年）则通过对训练数据进行调整来.....

rating explicit length constraint instructions, such as “Think for N tokens,” before applying policy optimization to pre-trained reasoning models. O1-Pruner (Luo et al., 2025) introduces a Length-Harmonizing Reward that calculates the ratio between reference model and predicted CoT lengths, incorporating accuracy constraints to prevent performance degradation while shortening reasoning processes. DAST (Shen et al., 2025a) employs SimPO to fine-tune models using a specially constructed length-preference dataset based on a self-defined token-length budget measurement. Kimi (Team et al., 2025) incorporates a length penalty into its policy optimization approach to improve long CoT activations and facilitate model merging.

- *Compression of Intermediate States.* A distinct category of techniques focuses on compressing intermediate data generated during the model’s inference process. For instance, Coconut (Chain of Continuous Thought) (Hao et al., 2024) treats final-layer hidden states as “continuous thought” to replace discrete tokens, reusing these hidden states as subsequent input embeddings and gradually adding latent CoT tokens during training, thereby improving accuracy while reducing intermediate “thinking” tokens. CODI (Shen et al., 2025b) employs self-distillation where the model serves as both teacher and student, jointly learning explicit and implicit CoT while aligning hidden activations, enabling internal reasoning without generating explicit tokens. CCOT (Cheng and Van Durme, 2024) condenses long reasoning into short, contentful contemplation tokens by precomputing full CoT, selecting important hidden states, and training LoRA modules to predict and decode these key tokens.

6.3.2. Semantic Caching

AI search applications frequently encounter a high degree of semantic similarity among queries submitted by different users. Consequently, a considerable fraction of distinct input queries can yield identical or substantially similar answers (e.g., “Tell me about the history of Hawaii” versus “What is the history of Hawaii?”). This redundancy is leveraged by various optimization strategies, including query caching, result caching, and query rewriting (Gill et al., 2025).

For instance, MeanCache (Gill et al., 2025) exemplifies a user-side semantic caching system tailored for LLM-powered web services. When a query is submitted to a MeanCache-enabled LLM service, the system computes an embedding vector for the query. This embedding is then compared against cached query embeddings using cosine similarity to identify potential matches. For each semantically similar query identified in the cache, MeanCache analyzes its associated context chain, comparing it with the dialogue history of the current query. If a match is found—that is, a cached query exhibits both semantic similarity and a congruent context chain—the corresponding response is retrieved from the local cache and served to the user. Otherwise, MeanCache forwards the query to the LLM service to elicit a response; this response, along with the query and its embedding, is subsequently stored in the cache. To preserve user privacy, MeanCache employs Federated Learning (FL) to train a compact embedding model, thereby obviating the need to store user data on a central server. The embedding model is trained using a multi-task learning approach that incorporates both contrastive loss and multiple-negatives ranking loss. This process also involves determining an optimal cosine similarity threshold, denoted as τ , to strike a balance between true positives (cache hits) and false positives. To mitigate storage and computational overheads, MeanCache applies Principal Component Analysis (PCA) for dimensionality reduction of the embedding vectors. Empirical studies indicate that approximately 31% of user queries exhibit similarity to previously submitted ones, underscoring the potential of user-side caching to significantly reduce LLM inference costs.

在对预训练推理模型实施策略优化之前，需先设定明确的长度约束指令，例如“思考N个词元”。O1-Pruner (Luo等人，2025年)提出了一种长度协调奖励机制，该机制会计算参考模型与预测出的思维链长度之比，同时纳入准确性约束，以便在缩短推理过程的同时避免性能下降。DAST (Shen等人，2025a)采用SimPO方法，利用基于自定义词元长度预算度量标准构建的特殊长度偏好数据集来对模型进行微调。Kimi (Team等人，2025年)则在其策略优化方法中加入了长度惩罚项，以此提升长思维链激活效果并促进模型合并。

- 中间状态的压缩。有一类独特的技术专注于压缩模型推理过程中生成的中间数据。例如，Coconut (连续思维链) (Hao等人，2024年)将最后一层的隐藏状态视为“连续思维”，用以替代离散的标记；这些隐藏状态会被重新用作后续输入的嵌入向量，且在训练期间会逐渐加入潜在的思维链标记，从而在提升准确性的同时减少中间“思考”标记的数量。CODI(Shen等人，2025b年)采用自蒸馏机制，让模型同时充当教师和学生角色，共同学习显式和隐式的思维链，同时对齐隐藏激活值，进而实现无需生成显式标记的内部推理。CCOT(Cheng和Van Durme，2024年)通过预先计算完整的思维链、选取重要的隐藏状态，并训练LoRA模块来预测和解码这些关键标记，从而将冗长的推理过程浓缩为简短且富有内容的沉思标记。

6.3.2. 语义缓存

AI搜索应用常常会遇到不同用户提交的查询在语义上高度相似的情况。因此，相当比例的不同输入查询可能会得到相同或极为相似的答案（例如，“请介绍一下夏威夷的历史”与“夏威夷的历史是怎样的？”）。这种冗余现象被各类优化策略所利用，包括查询缓存、结果缓存以及查询重写 (Gill等人，2025年)。

例如，MeanCache (Gill等人，2025年)便是一种专为基于大语言模型的网络服务设计的用户端语义缓存系统。当用户向支持MeanCache的大语言模型服务提交查询时，系统会先为该查询计算一个嵌入向量。随后，该系统会通过余弦相似度将这一嵌入向量与缓存中的查询嵌入向量进行对比，以找出潜在的匹配项。对于在缓存中识别出的每个语义相似的查询，MeanCache会分析其关联的上下文链，并将其与当前查询的对话历史进行比较。一旦找到匹配项——也就是说，缓存中的查询既具备语义相似性，又拥有一致的上下文链——相应的响应就会从本地缓存中提取并提供给用户。否则，MeanCache会将查询转发给大语言模型服务以获取响应；随后，该响应会与查询及其嵌入向量一同存入缓存。为保护用户隐私，MeanCache采用联邦学习技术来训练一个轻量级的嵌入模型，从而无需将用户数据存储在中央服务器上。该嵌入模型采用多任务学习方法进行训练，其中结合了对比损失与多负样本排序损失。这一过程还包括确定一个最佳的余弦相似度阈值，即 τ ，以便在真阳性（缓存命中）与假阳性之间取得平衡。为降低存储与计算开销，MeanCache运用主成分分析法对嵌入向量进行降维处理。实证研究表明，约31%的用户查询与之前提交过的查询具有相似性，这凸显了用户端缓存技术在大幅降低大语言模型推理成本方面的巨大潜力。

6.3.3. Quantization

Quantization is a widely adopted infrastructure-level lightweighting technique. It involves reducing the numerical precision of the model's weights and/or activations (e.g., from 32-bit floating-point, FP32, to 8-bit floating-point, FP8, or even lower bit integers). This reduction in data representation bit-width decreases memory footprint, memory bandwidth requirements, and can accelerate computations on hardware supporting lower precision arithmetic. Existing quantization methods for Large Language Models (LLMs) can be divided into two categories: weight-only quantization and weight+activation quantization.

- *Weight-only Quantization.* In weight-only quantization, models like ZeroQuant-V2 (Yao et al., 2023b) apply uniform quantization to OPT and BLOOM, finding that 8-bit weights with 16-bit activations preserve performance while 4-bit weights cause significant degradation. They introduce Low-Rank Compensation (LoRC) to approximate quantization error with a storage-efficient low-rank matrix. GLM-130B (Zeng et al., 2022) demonstrates successful 4-bit quantization using row-wise symmetric quantization, attributed to its well-shaped weight distributions. Non-uniform methods include LUT-GEMM/nuQmm (Park et al., 2022), which extends binary-coding quantization with bias terms and group-wise quantization. SqueezeLLM (Kim et al., 2023) uses sensitivity-based k-means centroids for non-uniform quantization, optimizing for memory-bound LLM inference. QLoRA (Dettmers et al., 2023) introduces NormalFormat datatype based on Quantile Quantization, normalizing weights to standard normal distribution. GPTQ (Frantar et al., 2023) builds on Optimal Brain Quantization, quantizing rows in parallel with lazy batch updates and Cholesky reformulation for stability. QuIP (Chee et al., 2023) defines optimal adaptive rounding methods using LDL decomposition and incoherence processing, achieving viable 2-bit quantization. AWQ (Lin et al., 2023) preserves significant activation channels in FP16 while quantizing others, using activation-aware scaling to better represent salient weights.
- *Weight+Activation Quantization.* For weight+activation quantization, LLM.int8 (Dettmers et al., 2022) identifies systematic outliers in activations, preserving outlier dimensions in high-precision while quantizing average values to INT8. RPTQ (Yuan et al., 2023) clusters and reorders activation dimensions based on min/max values, fusing operations to reduce latency. Low-bit floating-point formats like FP4/FP8 are explored in MoFQ (Zhang et al., 2023b) and ZeroQuant-FP (Wu et al., 2023), providing better handling of activation outliers than integer formats. SmoothQuant (Xiao et al., 2023a) suppresses outliers by scaling down activation dimensions and scaling up corresponding weight dimensions with a controllable hyperparameter. Outlier Suppression (Wei et al., 2022b) uses inverse gamma values from LayerNorm as scaling factors, while Outlier Suppression+ (Wei et al., 2023) adds shifting factors to remove asymmetry in activations. FPTQ (Li et al., 2023a) introduces logarithmic activation equalization for non-linear moderation of activation distributions. OmniQuant (Shao et al., 2023a) learns optimal clipping ranges for extreme values, and QLLM (Liu et al., 2023) splits outlier channels into sub-channels while merging similar ones.

6.3.4. Prefill-Decode Separating Deployment

The LLM inference process can be divided into two distinct phases: the *prefill* phase, which processes the input prompt and is computationally intensive (compute-bound), and the *decode* phase, which generates subsequent tokens autoregressively and is memory bandwidth-intensive (memory-bound). Recognizing these differing characteristics, mature LLM inference frameworks often implement separate deployment and resource allocation for these phases. For instance,

6.3.3. 量化

量化是一种被广泛采用的底层基础设施级轻量化技术。它涉及降低模型权重和/或激活值的数值精度（例如，从32位浮点数FP32降至8位浮点数FP8，甚至更低位的整数）。这种数据表示位宽上的缩减能够减少内存占用与内存带宽需求，同时还能在支持低精度算术运算的硬件上加速计算。针对大语言模型现有的量化方法可分为两类：仅权重量化和权重+激活值量化。

- 仅权重量化。在仅权重量化中，诸如ZeroQuant-V2 (Yao等人, 2023b) 之类的模型对OPT和BLOOM模型应用均匀量化，研究发现采用8位权重与16位激活值时能够保持原有性能，而4位权重则会导致显著的性能下降。他们引入了低秩补偿 (LoRC) 技术，利用存储效率高的低秩矩阵来近似量化误差。GLM-130B (Zeng等人, 2022) 通过行对称量化实现了成功的4位量化，这得益于其形状良好的权重分布。非均匀量化方法包括LUT-GEMM/nuQmm (Park等人, 2022)，该方法在二进制编码量化的基础上加入了偏置项和分组量化机制。SqueezeLLM (Kim等人, 2023) 采用基于敏感度的k均值聚类中心进行非均匀量化，旨在优化受内存限制的大型语言模型推理过程。QLoRA (Dettmers等人, 2023) 引入了基于分位数量化的NormalFormat数据类型，将权重归一化到标准正态分布。GPTQ (Frantar等人, 2023) 以最优脑量化为基础，通过并行量化各行数据，并结合延迟批量更新与乔列斯基重构来保证稳定性。QuIP (Chee等人, 2023) 利用LDL分解与非相干处理技术定义了最优的自适应舍入方法，从而实现了可行的2位量化。AWQ (Lin等人, 2023) 在FP16精度下保留重要的激活通道，同时对其他通道进行量化，借助激活感知缩放技术来更精准地表示关键权重。
- 权重+激活值量化。在权重与激活值量化方面，LLM.int8 (Dettmers等人, 2022年) 能够识别出激活值中的系统性异常值，保留高精度表示的异常维度，同时将平均值量化为INT8格式。RPTQ (Yuan等人, 2023年) 根据最小值和最大值对激活维度进行聚类与重排，通过融合运算来降低延迟。MoFQ (Zhang等人, 2023b年) 以及ZeroQuant-FP (Wu等人, 2023年) 探索了FP4/FP8这类低比特浮点格式，相较于整数格式，它们能更好地处理激活值中的异常值。SmoothQuant (Xiao等人, 2023a年) 通过缩小激活维度的数值范围、放大对应权重的数值范围来抑制异常值，这一过程可通过一个可控的超参数进行调整。Outlier Suppression (Wei等人, 2022b年) 利用LayerNorm输出的逆伽马值作为缩放因子；而Outlier Suppression+ (Wei等人, 2023年) 则引入偏移因子以消除激活值中的不对称性。FPTQ (Li等人, 2023a年) 采用对数形式的激活值均衡技术来对激活值分布进行非线性调节。OmniQuant (Shao等人, 2023a年) 能够学习到针对极端值的最优截断范围，QLLM (Liu等人, 2023年) 则将异常通道拆分为子通道，同时合并相似的通道。

6.3.4. 预填充与解码分离部署

大语言模型推理过程可分为两个截然不同的阶段：预填充阶段负责处理输入提示，该阶段计算量极大（属于计算密集型）；解码阶段则用于自回归地生成后续词元，对内存带宽的需求较高（属于内存密集型）。鉴于这两阶段的特性差异，成熟的大语言模型推理框架通常会为它们分别部署并分配资源。例如：

systems like Mooncake (Qin et al., 2024), or as implied by the architecture of models like DeepSeek-R1 (DeepSeek-AI et al., 2025), dynamically adjust the ratio of servers or resources dedicated to prefill and decode tasks based on user load and request patterns. This dynamic allocation ensures optimal utilization of computational resources, minimizing bottlenecks and improving throughput. For example, a system might allocate a certain number of nodes/GPUs for parallel prompt processing (prefill) and a different set for token generation (decode), scaling each pool independently.

6.3.5. Speculative Decoding

Speculative decoding represents a significant advancement in accelerating large language model inference through parallel processing, and existing methods can be categorized into generation and refinement strategies. Generation approaches include predefined fill tokens, which use random initialization but require multiple refinement iterations; retrieval-based methods like REST (He et al., 2023) that leverage exact suffix matching from datastores; and N-gram techniques such as ANPD (Ou et al., 2024) that adapt predictions based on contextual patterns.

Auto-regressive generation represents the most common approach, split between independent and dependent drafters. Independent drafters like SpecDec (Xia et al., 2023) use smaller models to generate tokens verified by larger target models, with improvements like BiLD (Kim et al., 2024) adding confidence thresholds for verification. Dependent drafters reduce computation by sharing resources between draft and target models. Layer skipping methods like Draft&Verify (Zhang et al., 2023a) selectively skip intermediate layers, while dependent head approaches like EAGLE (Li et al., 2024a) add lightweight prediction heads to the target model’s hidden states. Multi-token prediction techniques generate multiple tokens simultaneously. Medusa (Cai et al., 2024) introduced parameter-efficient lightweight decoding heads on pre-trained models, while Amphista (Li et al., 2024c) enhanced this with bi-directional self-attention to model inter-token relationships.

For refinement strategies, single-pass verification dominates the field. Linear verification methods sequentially validate draft tokens against the target model’s distributions, with Fast Inference (Leviathan et al., 2023) introducing speculative sampling to improve acceptance rates. Tree-based verification approaches like SpecInfer (Miao et al., 2023) construct trees of possible completions for parallel exploration, with optimizations like Sequoia’s (Chen et al., 2024) hardware-aware tree optimizer maximizing computational efficiency. Iterative decoding methods permit multiple refinement iterations, with approaches like Jacobi (Teng et al., 2024) reframing auto-regressive generation as an iterative optimization problem that can produce identical output to traditional decoding under greedy sampling.

7. Evaluation

To further investigate the effectiveness of the AI search system in a real-world environment, we perform extensive experiments with real-world traffic and compare it with the legacy system (*i.e.*, web search system) at Baidu Search.

7.1. Human Evaluation

To comprehensively evaluate the impact of the proposed AI search system, we conduct a series of human assessments using a side-by-side comparison approach. First, we log a large set of

像Mooncake (Qin等人, 2024年) 这样的系统, 或是DeepSeek-R1 (DeepSeek-AI等人, 2025年) 这类模型的架构所体现的思路, 都会根据用户负载和请求模式动态调整用于预填充和解码任务的服务器或资源比例。这种动态分配方式能够确保计算资源得到最优利用, 从而最大限度减少瓶颈并提高吞吐量。举例来说, 系统可能会分配一定数量的节点/GPU用于并行处理提示(预填充), 再另外分配一组资源用于词元生成(解码), 且这两个资源池可独立进行扩容。

6.3.5. 投机解码

投机解码是通过并行处理来加速大语言模型推理的一项重要技术突破, 现有方法可分为生成策略和优化策略两类。生成类方法包括使用预定义填充词的方法——这类方法采用随机初始化, 但需要多次迭代优化; 基于检索的方法如REST (He等人, 2023年), 利用数据存储中的精确后缀匹配机制; 以及N-gram技术, 例如ANPD (Ou等人, 2024年), 可根据上下文模式调整预测结果。

自回归生成是最常用的方式, 可分为独立式起草器和依赖式起草器两类。独立式起草器如SpecDec (Xia等人, 2023年) 利用较小规模的模型生成令牌, 再由规模较大的目标模型进行验证; BiLD (Kim等人, 2024年) 等改进方案还引入了置信度阈值以辅助验证过程。依赖式起草器则通过共享草稿模型与目标模型之间的资源来减少计算量。Draft&Verify (Zhang等人, 2023a) 等层跳过方法会选择性地跳过中间层; 而EAGLE (Li等人, 2024a) 等依赖式头结构方法则在目标模型的隐藏状态上添加轻量级的预测头。多令牌预测技术能够同时生成多个令牌: Medusa (Cai等人, 2024年) 在预训练模型上引入了参数效率高的轻量级解码头; Amphista (Li等人, 2024c) 则通过双向自注意力机制来建模令牌间的关联关系, 进一步提升了性能。

在优化策略方面, 单轮验证占据主导地位。线性验证方法会依次根据目标模型的分布来验证生成的词元, 其中Fast Inference (Leviathan等人, 2023年) 引入了推测采样技术以提升接受率。SpecInfer (Miao等人, 2023年) 等基于树结构的验证方法会构建可能的补全序列树以实现并行探索; Sequoia (Chen等人, 2024年) 提出的硬件感知树优化器等优化手段则能最大化计算效率。迭代解码方法允许进行多次优化迭代, 例如Jacobi (Teng等人, 2024年) 将自回归生成重新定义为迭代优化问题, 在贪婪采样条件下可产出与传统解码方式完全一致的输出。

7. 评估

为了进一步探究该AI搜索系统在真实环境中的有效性, 我们利用真实的网络流量开展了大量实验, 并将其与百度搜索中原有的系统(即网页搜索系统)进行了对比。

7.1. 人工评估

为了全面评估所提出的AI搜索系统的影响, 我们采用并列比较的方式开展了一系列人工评估。首先, 我们记录大量查询及其对应的搜索结果, 这些结果分别由AI搜索系统和传统系统返回。

Table 1 | Relative improvements of the proposed AI search system (*AI Search*) to the legacy system (*Web Search*) across query complex categories in the manual evaluation.

Model	NWR		
	Simple	Moderately Complex	Complex
Web Search	-	-	-
AI Search	0.00%	5.00%*	13.00%*

* indicates the statistically significant
(*t*-test with $p < 0.05$ over the legacy system).

queries and their corresponding search results, which are returned by both the AI search system and the legacy system. Then, we employ professional annotators to judge the superiority of the results from the AI search system compared to those from the legacy system. The manual evaluation results are measured with *Win vs. Tie vs. Lose*, which is a metric measured by the professional annotators’ judgment. For a query, the annotators are provided with a pair ($Result_1$, $Result_2$) whereby one result is returned by system A, and the other is generated by a competitor system B. The annotators, who do not know which system the result is from, are then required to independently rate among *Win* ($Result_1$ is better), *Lose* ($Result_2$ is better), and *Tie* (they are equally tied). In order to quantify the human evaluation, we aggregate these three indicators mentioned above as a unified metric, denoted as Normalized Win Rate (NWR):

$$NWR = \frac{\#Win - \#Lose}{\#Win + \#Tie + \#Lose}, \tag{46}$$

where $\#Win$ (or $\#Lose$) is the number of results generated from the AI search system better (or worse) than the legacy system, and $\#Tie$ indicates two results are equally good or bad. In the manual evaluation, we consider the influence of query complexity on the results for the evaluation. Therefore, we sample three categories of simple, moderately complex, and complex queries to comprehensively compare the two systems. Specifically, the test dataset is constructed in advance by sampling real user queries through two distinct selection strategies as follows:

- *Targeted Sampling*. We employ DeepSeek-R1 prompts to assess the complexity of queries and classify them into three categories: simple queries, moderately complex queries, and complex queries.
- *Random Sampling*. The test queries are randomly selected, which consist of high-frequency queries, random queries, and long-tail queries.

As shown in Table 1, the AI search system delivers substantial improvements to the legacy system, which reveals that the AI search system could better meet user satisfaction. Specifically, for simple queries, the AI search system performs comparably to the legacy system. However, for moderately complex and complex queries, the AI search system demonstrates significant improvements, achieving a relative enhancement of WTL by 13% for complex queries. These human evaluation results indicate that the AI search system is capable of delivering accurate and comprehensive responses in complex query scenarios, thereby effectively satisfying user search requirements and validating its effectiveness.

表 1 |：在人工评估中，所提出的AI搜索系统（AI搜索）相较于传统系统（网页搜索）在不同查询复杂度类别下的相对性能提升。

模型	NWR		
	简单	中等复杂度	复杂
网页搜索	-	-	-
AI搜索	0.00%	5.00%*	13.00%*

* 表示具有统计学意义。
(*t*检验，显著性水平为 $p < 0.05$ ，针对传统系统进行比较)。

随后，我们聘请专业标注员来判断AI搜索系统产生的结果相较于传统系统的结果是否更优。人工评估的结果通过胜/平/负这一指标来衡量，该指标由专业标注员的判断得出。针对每条查询，标注员会收到一对结果（ $Result_1$ 与 $Result_2$ ），其中一份结果由系统A返回，另一份则由竞争对手系统B生成。标注员在不知晓结果来源系统的情况下，需独立评定其属于胜（ $Result_1$ 更优）、负（ $Result_2$ 更优）还是平（两者相当）。为了量化人工评估结果，我们将上述三个指标整合为一个统一指标，即归一化胜率（NWR）：

$$NWR = \frac{\#Win - \#Lose}{\#Win + \#Tie + \#Lose}, \tag{46}$$

其中， $\#Win$ （或 $\#Lose$ ）代表AI搜索系统生成的结果中优于（或逊于）传统系统的数量，而 $\#Tie$ 则表示两者效果相当。在人工评估中，我们考虑了查询复杂度对结果的影响。因此，我们选取简单、中等复杂和复杂三类查询样本，以全面对比这两个系统。具体而言，测试数据集是通过采用两种不同的选取策略，从真实用户查询中提前采集而成的：

- 定向采样。我们利用DeepSeek-R1提示词来评估查询的复杂度，并将其分为简单查询、中等复杂查询和复杂查询三类。
- 随机采样。测试查询是随机选取的，其中包括高频查询、随机查询以及长尾查询。

如表1所示，AI搜索系统相较于传统系统有显著改进，这表明AI搜索系统能够更好地满足用户需求。具体而言，对于简单查询，AI搜索系统的表现与传统系统相当；但对于中等复杂及复杂的查询，AI搜索系统则展现出明显优势，在复杂查询场景下WTL指标的相对提升幅度达到了13%。这些人工评估结果表明：AI搜索系统能够在复杂查询场景中给出准确且全面的回答，从而有效满足用户的搜索需求，也验证了其实际效用。

Table 2 | Improvements of the proposed AI search system (*AI Search*) compared with the online legacy system (*Web Search*) on user-side metrics in online A/B Test.

Model	CQR	PV	DAU	Dwell Time
<i>Web Search</i>	-	-	-	-
<i>AI Search</i>	-1.45%	1.04%	1.85%	0.52%

7.2. Online A/B Test

To evaluate the real-world effectiveness of the AI search system, we deploy it at Baidu Search and conduct an A/B Test comparing it with the legacy search system. When deploying the AI search system at Baidu Search, specific user-side metrics serve as critical indicators of the effectiveness of search systems in enhancing users’ search satisfaction. In the online A/B test, we conduct the experiments with 1.00% real-world web traffic of Baidu search, focusing on metrics directly related to user engagement and reporting the improvements of the proposed AI search system over the online legacy system. As shown in Table 2, compared to the online legacy system, the AI search system achieves the following relative improvements: the change query rate (CQR) decreases by 1.45%; the number of page views (PV) increases by 1.04%; the number of daily active users (DAU) increases by 1.85%; dwell time (Dwell Time) increases by 0.52%. All reported values are statistically significant with $p < 0.05$. These results suggest that the proposed AI search system substantially enhances user engagement within the search engine context.

7.3. Case Study

As depicted in Fig. 15, we present two representative case studies. One case study compares the outcomes of the AI search system with those of the legacy system for a simple query, while the other examines their respective performance for a complex query. Specifically, for a simple query such as “*How tall is Mount Tai?*”, both the AI search system and the legacy search system promptly provided the correct answer—that “Mount Tai is 1,545 meters tall”. This case study demonstrates that for straightforward queries, the AI search system can effectively respond by either invoking the web search tool or leveraging its internalized knowledge, while traditional web search systems achieve the correct response through a series of retrieval and ranking processes. However, for complex queries that require multi-step reasoning, traditional web search methods exhibit notable limitations in fulfilling user requirements and often fail to produce correct answers. For instance, when faced with the complex query “Who is older, Emperor Han-Wu or Emperor Caesar, and by how many years?”, the legacy search system cannot directly yield the final answer through simple retrieval and ranking procedures. By contrast, the proposed AI search system employs a *Master*-assigned *Planner*-Enhanced configuration. In this approach, *Planner* first decomposes the main query into sub-tasks and dynamically adjusts the capability boundary by invoking the appropriate tools. Subsequently, *Executor* concurrently carries out two sub-tasks—retrieving the ages of Emperor Han-Wu and Emperor Caesar. Based on the results of these sub-tasks, a calculation tool is then leveraged to compute the age difference, and finally, *Writer* synthesizes the findings to present the final answer. This case study clearly demonstrates the advantages and necessity of the AI search system in effectively addressing complex queries. According to these two cases, we find that the AI search system has an enhanced capability to deliver more accurate results for user satisfaction-oriented searches than the legacy system.

表 2 |：所提出的AI搜索系统（*AI搜索*）与在线传统系统（网页搜索）在在线A/B测试中用户端指标的对比改进情况。

模型	CQR	光 伏	日活跃 用户数	停留时间
网页搜索	-	-	-	-
<i>AI搜索</i>	-1.45%	1.04%	1.85%	0.52%

7.2. 在线A/B测试

为了评估AI搜索系统在现实场景中的实际效果，我们将其部署于百度搜索平台，并与传统搜索系统开展A/B测试。将AI搜索系统应用于百度搜索时，特定的用户端指标可作为衡量搜索系统提升用户搜索满意度成效的关键依据。在本次在线A/B测试中，我们选取了百度搜索1.00%的真实网页流量进行实验，重点考察与用户参与度直接相关的指标，以反映所提出的AI搜索系统相较于在线传统系统的改进情况。如表2所示，与在线传统系统相比，AI搜索系统取得了以下相对提升：查询变更率（CQR）下降了1.45%；页面浏览量（PV）增加了1.04%；日活跃用户数（DAU）增长了1.85%；停留时间（Dwell Time）延长了0.52%。所有报告数据在 $p < 0$ 水平上均具有统计学意义。这些结果说明：所提出的AI搜索系统能够显著提升搜索引擎环境下的用户参与度。

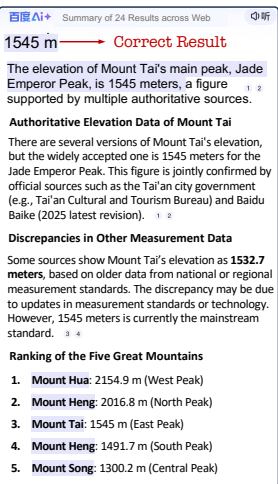
7.3. 案例研究

如图15所示，我们展示了两个具有代表性的案例研究。其中一个案例对比了AI搜索系统与传统系统在简单查询下的表现，另一个则分析了它们在复杂查询中的性能差异。具体而言，对于像“泰山有多高？”这样的简单查询，AI搜索系统和传统搜索系统都能迅速给出正确答案——即“泰山海拔1,545米”。这一案例表明，对于简单的查询，AI搜索系统既能通过调用网页搜索工具来响应，也能利用自身内置的知识库；而传统的网页搜索系统则需经过一系列检索和排序步骤才能得到正确结果。然而，对于需要多步推理的复杂查询，传统网页搜索方法在满足用户需求方面存在明显局限，往往无法给出正确答案。例如，面对“汉武帝和凯撒大帝谁年纪更大？大多少岁？”这类复杂查询时，传统搜索系统无法通过简单的检索和排序过程直接得出最终答案。相比之下，我们所提出的AI搜索系统采用了由*Master*指定的*Planner*增强配置。在此机制下，*Planner* 首先把主查询拆解为多个子任务，并通过调用合适的工具来动态调整能力边界；随后，*Executor*会同时执行两个子任务——分别获取汉武帝和凯撒大帝的年龄信息。基于这些子任务的结果，系统会利用计算工具算出年龄差，最后由*Writer* 整合所有信息并给出最终答案。这一案例清楚地体现了AI搜索系统在高效处理复杂查询方面的优势与必要性。综合这两个案例可知，相较于传统系统，AI搜索系统在提供更符合用户期望的精准结果方面具备更强的能力。

Query: How tall is Mount Tai?



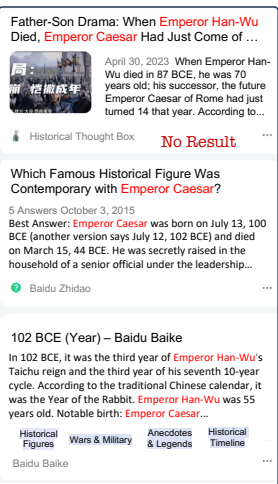
Legacy Search System



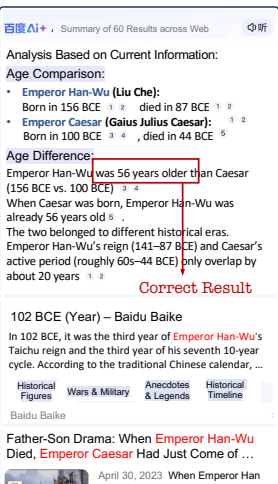
AI Search System

(a) A case study of the simple query.

Query: Who is elder, Emperor Han-Wu or Emperor Caesar, by how many years?



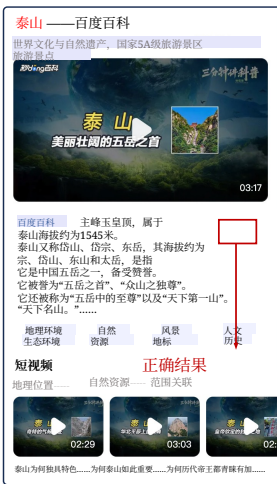
Legacy Search System



AI Search System

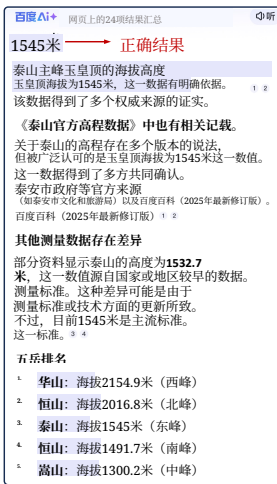
(b) A case study of the complex query.

查询: 泰山有多高?



传统搜索系统

(a) 简单查询的案例分。



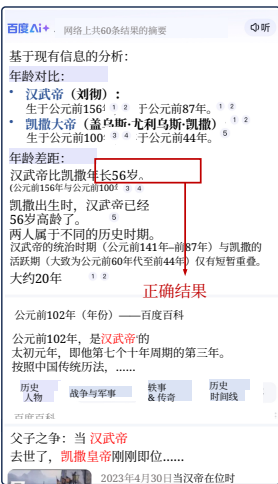
人工智能搜索系统

查询: 汉武帝比凯撒大帝年长多少岁?



传统搜索系统

(b) 复杂查询的案例研究。



人工智能搜索系统

Figure 15 | Online case comparisons of the AI search system to the legacy search system.

8. Conclusion

In this paper, we propose a novel AI search paradigm that fundamentally rethinks the traditional information-seeking process. Our proposed AI search system leverages a modular, multi-agent architecture to emulate and enhance human-like problem solving. Specifically, the Master Agent dynamically assesses query complexity and orchestrates the formation of an agent team, while the Planner Agent decomposes complex queries into a structured DAG of sub-tasks with explicit interdependencies. The Executor Agent then carries out these sub-tasks through the coordinated invocation of diverse tools, and the Writer Agent synthesizes the collected results to generate comprehensive answers. This multi-agent framework addresses the limitations of classical IR and state-of-the-art RAG systems, which typically provide only a linear, document-retrieval oriented response. By enabling proactive planning, dynamic tool integration, and iterative reasoning, our system can effectively handle complex, multi-step queries, thereby reducing the cognitive load on users and improving overall search quality. The empirical evaluations, as discussed in the experimental sections, further validate the system's effectiveness in delivering accurate, context-aware responses and enhanced user engagement. Overall, our work contributes a structured and detailed blueprint that integrates best practices from both industry and academia. It lays the groundwork for future research in AI-driven information seeking and highlights several promising directions for optimizing the performance of collaborative agents and seamless tool integration.

Acknowledge

We sincerely appreciate all the engineers of Baidu Search for their contributions to the construction of the AI search system.

8. 结论

在本文中，我们提出了一种全新的AI搜索范式，从根本上重新思考了传统的信息检索流程。我们所设计的AI搜索系统采用模块化的多智能体架构，以模拟并增强人类的问题解决能力。具体而言，Master智能体会动态评估查询的复杂程度，并协调组建相应的智能体团队；Planner智能体则将复杂查询拆解为具有明确依赖关系的结构化DAG子任务；Executor智能体通过协同调用各类工具来完成这些子任务；Writer智能体则负责整合所得结果并生成全面的回答。这种多智能体框架弥补了传统信息检索系统以及当前先进的RAG系统在响应方式上的不足——后者通常仅提供线性的、以文档检索为导向的结果。通过实现主动规划、动态工具集成以及迭代推理，该系统能够有效处理复杂的多步查询，从而减轻用户的认知负担并提升整体搜索质量。实验部分所呈现的评估结果进一步证实了该系统在提供准确、符合语境的回答以及增强用户参与度方面的有效性。总体而言，我们的工作提供了一套结构清晰且详实的设计蓝图，融合了业界与学术界的最佳实践，为未来AI驱动的信息检索研究奠定了基础，同时也指出了优化协作智能体与无缝工具集成性能的几个有前景的方向。

致谢

我们由衷感谢百度搜索的所有工程师为人工智能搜索系统的建设所做出的贡献。

References

P. S. Abril and R. Plant. The patent holder’s dilemma: Buy, sell, or troll? *Communications of the ACM*, 50(1):36–44, Jan. 2007. doi: 10.1145/1188913.1188915. URL <http://doi.acm.org/10.1145/1219092.1219093>.

J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

P. Aggarwal and S. Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.

Anthropic. Introducing the model context protocol. <https://www.anthropic.com/news/model-context-protocol>, Nov. 2024. Accessed 14 Jun 2025.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2023.

Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024.

Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, 2007.

C.-M. Chan, C. Xu, R. Yuan, H. Luo, W. Xue, Y. Guo, and J. Fu. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*, 2024.

J. D. Chang, K. Brantley, R. Ramamurthy, D. Misra, and W. Sun. Learning to generate better than your LLM. *CoRR*, abs/2306.11816, 2023. doi: 10.48550/ARXIV.2306.11816. URL <https://doi.org/10.48550/arXiv.2306.11816>.

参考文献

附言：Abril和R. Plant所著的《专利持有者的困境：购买、出售还是进行专利诉讼？》《ACM通讯》，第50卷第1期，第36–44页，2007年1月。doi: 10.1145/1188913.1188915。网址：<http://doi.acm.org/10.1145/1219092.1219093>。

J. Achiam、S. Adler、S. Agarwal、L. Ahmad、I. Akkaya、F. L. Aleman、D. Almeida、J. Altenschmidt、S. Altman、S. Anadkat等。GPT-4技术报告。*arXiv预印本arXiv:2303.08774*，2023年。P. Aggarwal和S. Welleck. L1：利用强化学习控制推理模型的思考时长。*arXiv预印本arXiv:2503.04697*，2025年。

Anthropic公司。模型上下文协议介绍。<https://www.anthropic.com/news/model-context-protocol>，2024年11月。访问于2025年6月14日。

D. Arthur和S. Vassilvitskii. k-means++：精心选择初始值所带来的优势。斯坦福大学技术报告，2006年。

A. Asai、Z. Wu、Y. Wang、A. Sil和H. Hajishirzi。Self-rag：通过自我反思学习检索、生成与评判。发表于第十二届国际学习表征会议，2023年。

Y. Bai、S. Kadavath、S. Kundu、A. Askell、J. Kernion、A. Jones、A. Chen、A. Goldie、A. Mirhoseini、C. McKinnon等。Constitutional AI：基于AI反馈实现无害性。*arXiv预印本arXiv:2212.08073*，2022年。

Y. Bengio、J. Louradour、R. Collobert和J. Weston。课程学习。收录于第26届年度国际机器学习会议论文集，第41–48页，2009年。

S. Brin和L. Page。大规模超文本网络搜索引擎的结构剖析。计算机网络与ISDN系统，第30卷(1-7期)：107–117页，1998年。

S. Bubeck、V. Chandrasekaran、R. Eldan、J. Gehrke、E. Horvitz、E. Kamar、P. Lee、Y. T. Lee、Y. Li、S. Lundberg等人合著的《人工智能通用智能的萌芽：基于GPT-4的早期实验》，*arXiv预印本arXiv:2303.12712*，2023年。

C. Burges、T. Shaked、E. Renshaw、A. Lazier、M. Deeds、N. Hamilton和G. Hullender撰写的《利用梯度下降进行排序学习》，收录于第22届国际机器学习会议论文集，第89–96页，2005年。

T. Cai、Y. Li、Z. Geng、H. Peng、J. D. Lee、D. Chen和T. Dao提出的《Medusa：一种采用多个解码头的简单LLM推理加速框架》，*arXiv预印本arXiv:2401.10774*，2024年。Z. Cao、T. Qin、T.-Y. Liu、M.-F. Tsai和H. Li所著的《排序学习：从成对方法到列表方法》，收录于第24届国际机器学习会议论文集，第129–136页，2007年。

陈志明、徐晨、袁睿、罗浩、薛伟、郭阳、付杰。Rq-rag：学习为检索增强生成优化查询。*arXiv预印本arXiv:2404.00610*，2024年。

张建东、布兰特利、拉马穆尔蒂、米斯拉、孙伟。学会生成优于现有大语言模型的内容。*CoRR*，abs/2306.11816，2023年。doi: 10.48550/ARXIV.2306.11816。URL <https://doi.org/10.48550/arXiv.2306.11816>。