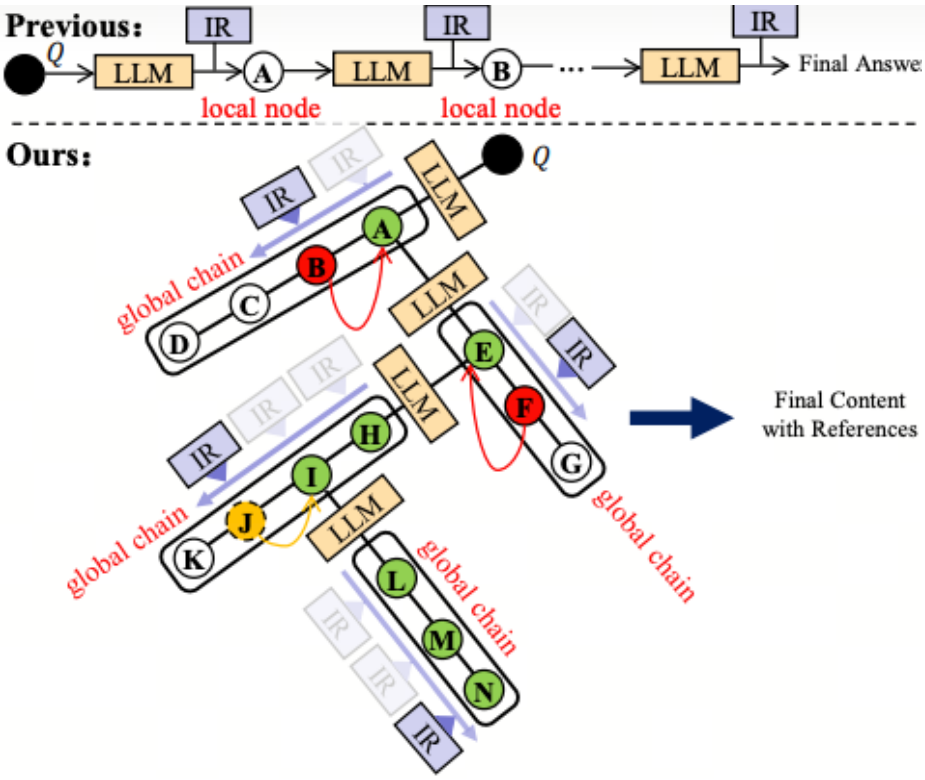
知乎



中国科学院大学2024：Search-in-the-Chain——揭秘大型语言模型如何赋能交互式搜索技术

SmartMindAI
专注搜索、广告、推荐、大模型和人工智能最新技术，欢迎关注我
已关注

3 人赞同了该文章

Introduction

当前LLM在组合推理和处理长尾及实时信息方面存在缺陷，(1) 复杂知识的组合推理能力不足；(2) 无法有效记忆长尾和实时信息；(3)生成与事实不符的；影响其在复杂任务中的准确性和可信度。此外，仅依赖上下文生成缺乏证据支持，降低了内容的可追溯性，使人对LLM生成内容的信任度降低。检索增强方法因其能结合模型知识与外部知识库，有潜力解决这些问题。

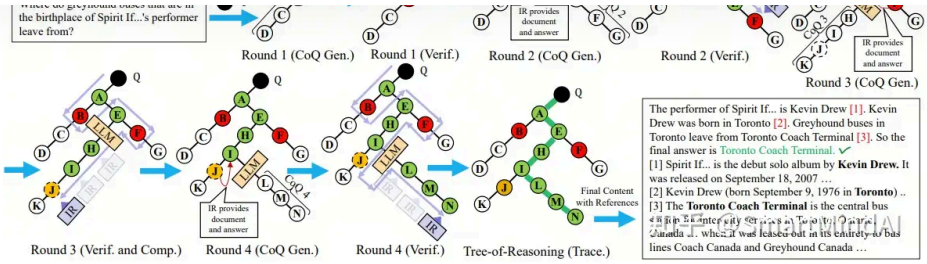
本研究致力于开发'在链中搜索'（SearChain）框架，解决这些问题，以提升LLM在知识密集型任务中的表现，增强内容的可信度和可追溯性。

论文提出在链中搜索（SearChain）框架，通过多轮LLM与IR交互来改进大型语言模型*的性能。SearChain区别于传统方法，它构建了Chain-of-Query（CoQ），这是一种链式查询策略，用于解决复杂问题，而非仅处理单个节点。每个节点包含LLM生成的答案和对额外知识的需求标志。通过LLM在情境学习中构建CoQ，避免了IR中断推理链（克服挑战$\mathcal{C}$-1）。

在交互过程中，IR对LLM的答案进行验证，如果发现不一致且IR认为重要，会提供修正建议，这有助于减少LLM因错误信息而误导（$\mathcal{C}$-2）。通过IR对LLM知识的校验，提高了内容的可信度。最后，SearChain跟踪整个推理过程，生成可追溯的生成链，增加了用户对LLM生成内容的信任。

Our Method

知乎



本节介绍SearChain，一个融合检索系统（IR）和大语言模型✦（LLM）的交互设计。LLM作为主导，构建全局推理链------Chain-of-Query（CoQ），每个节点包含查询、答案及LLM需确认标志。IR与CoQ节点互动，提供修正信息，避免误导，仅当必要时才补充LLM所缺知识。通过迭代交互，LLM逐渐完善CoQ，生成精确且可信的解决方案。SearChain还附加追踪功能，记录整个推理过程并附带文档引用，以提升内容的可追溯性，推理路径转化为推理树（Tree-of-Reasoning，ToR）上的节点深度优先搜索✦，使LLM能动态调整路径。

Comparison with Previous Methods

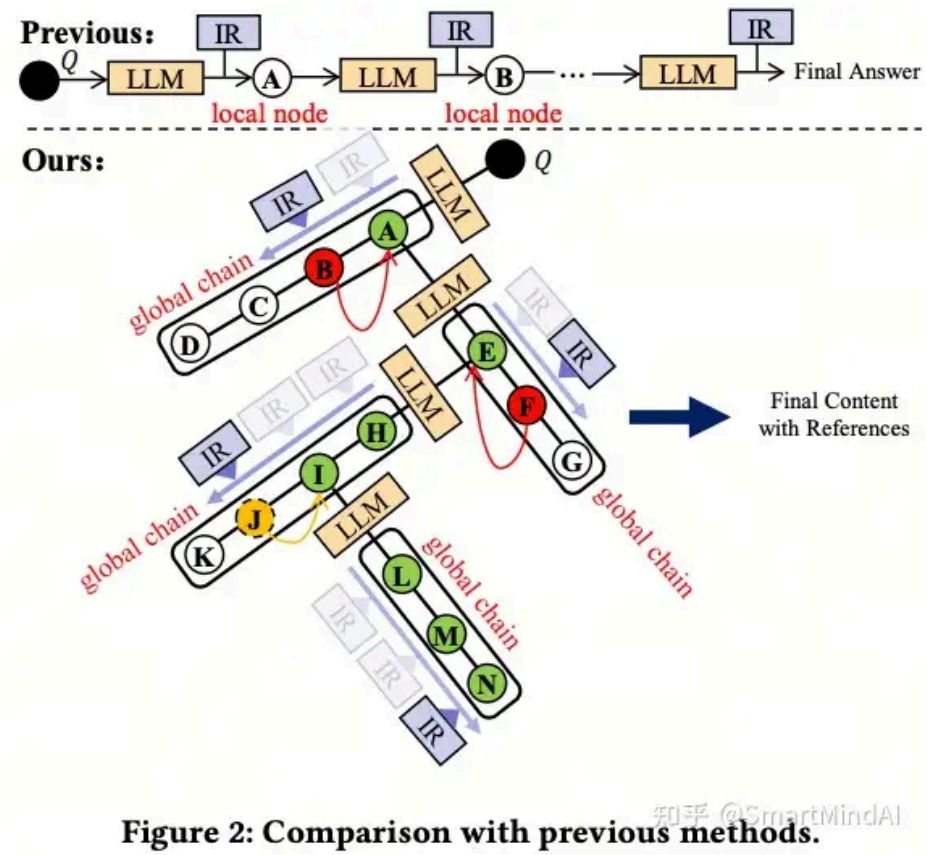


Figure 2: Comparison with previous methods.

图展示比较了本文的方法与基于检索增强技术（如Self-Ask、React、DSP等）在解决复杂知识密集问题上的区别。主要差异包括： 1. 全局与局部： 传统方法将IR直接嵌入多步推理，导致LLM局限于局部节点（如A□），而你们的方法通过Chain-of-Query，提供全局交互界面，保持推理链条连贯，解决了$\mathcal{C}$-1。

1. 信息提供与验证： 以往直接传递检索信息，可能误导LLM。你们的方法在IR确信时仅修正不一致信息，通过标记未知，减少了错误传递，解决了$\mathcal{C}$-2。
2. 路径调整： 前者路径固定，不能动态变化。你们的方法将路径转化为树状结构✦，允许LLM根据IR验证和补充动态调整推理路径，解决了$\mathcal{C}$-3。

Chain-of-Query Generation

在SearChain中，我们借助上下文学习，促使LLM为复杂问题Q构造全局推理链，即Chain-of-Query（CoQ）。

知乎

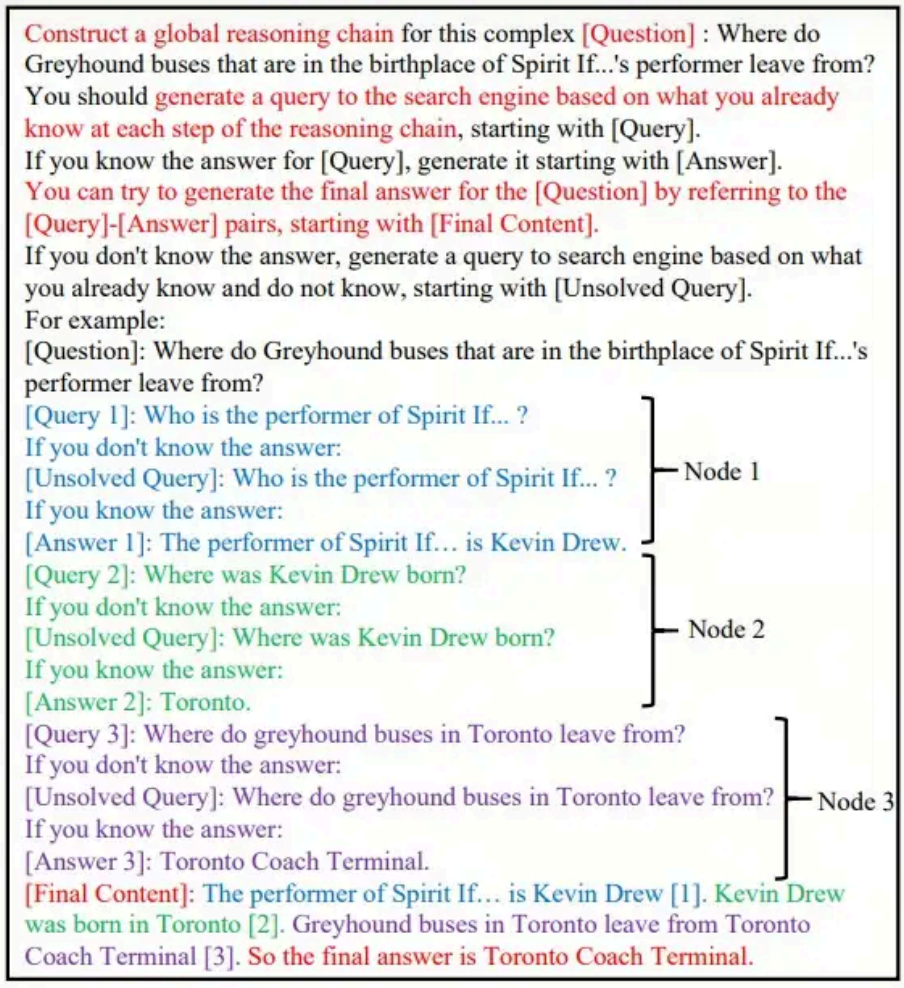


Figure 3: Prompt to make LLM generate Chain-of-Query.

在SearChain中，我们利用上下文学习策略，驱动LLM构建Chain-of-Query（CoQ），这是一个处理复杂问题Q的全局推理结构。每个CoQ节点由IR引导的查询和答案构成，代表解决子问题的步骤。初始，LLM通过Construct a global reasoning chain生成整个链条，确保从整体角度规划问题解决。节点标记为$UnsolvedQuery$表示未知领域。随着交互，LLM根据IR反馈调整CoQ，如需修正或补充信息，这使得推理过程动态且深入。实验显示，CoQ能促使LLM处理更复杂的子问题，通过多次推理步骤，而基线可能在遇到难题时就终止，因为它们通常只关注局部而不顾全局。CoQ的全局视角使得LLM在面对中等难度时，能尝试多种解答，这体现了其在问题解决中的优越性。

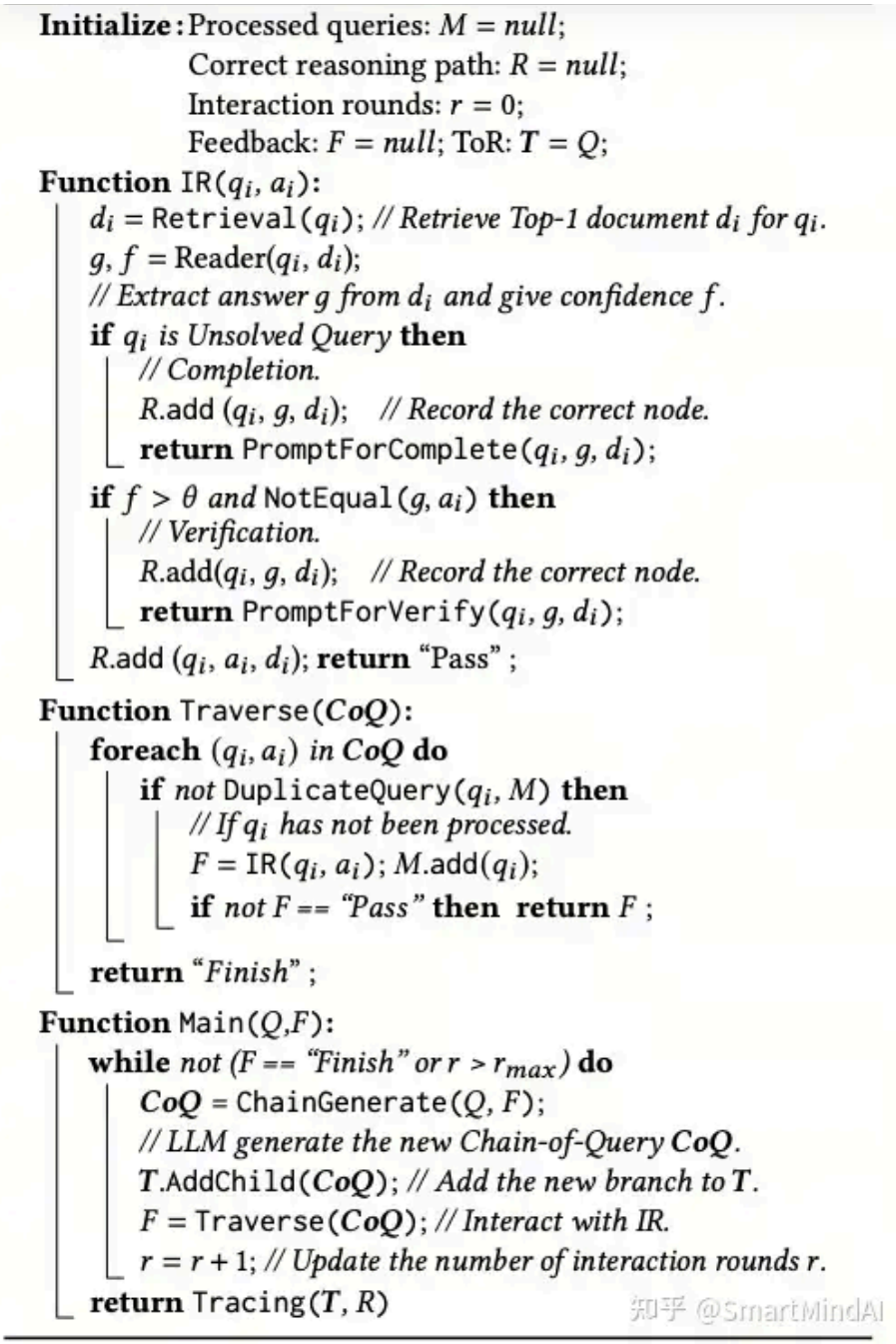
Interaction with Information Retrieval

$$s = \arg\max(\text{softmax}(\mathbf{H}\mathbf{w}_s)), e = \arg\max(\text{softmax}(\mathbf{H}\mathbf{w}_e)),$$

$$g = d_i[s:e], f = \mathbf{H}_{[CLS]}\mathbf{w}_f, (\mathbf{w}_s, \mathbf{w}_t, \mathbf{w}_f \in \mathbb{R}^E),$$

1. 对于短任务，如多跳问答和槽填充，检查答案a_i中是否存在直接相关项g，作为验证标准。
2. 若是长而开放性*任务，如ELI5，通过ROUGE相似度（如α阈值）评估a_i与d_i的一致性。若指标超过阈值，则认为一致。

知乎



Initialize: Processed queries: $M = null$;
Correct reasoning path: $R = null$;
Interaction rounds: $r = 0$;
Feedback: $F = null$; ToR: $T = Q$;

Function IR(q_i, a_i):
  d_i = Retrieval(q_i); *// Retrieve Top-1 document d_i for q_i.*
  g, f = Reader(q_i, d_i);
  // Extract answer g from d_i and give confidence f.
  if *q_i is Unsolved Query* **then**
    // Completion.
    R.add (q_i, g, d_i); *// Record the correct node.*
    return PromptForComplete(q_i, g, d_i);
  if $f > \theta$ *and* NotEqual(g, a_i) **then**
    // Verification.
    R.add(q_i, g, d_i); *// Record the correct node.*
    return PromptForVerify(q_i, g, d_i);
  R.add (q_i, a_i, d_i); **return** "Pass" ;

Function Traverse(CoQ):
  foreach (q_i, a_i) *in* CoQ **do**
    if *not* DuplicateQuery(q_i, M) **then**
      // If q_i has not been processed.
      F = IR(q_i, a_i); M.add(q_i);
      if *not* F == *"Pass"* **then return** F ;
  return *"Finish"* ;

Function Main(Q,F):
  while *not* (F == *"Finish"* or $r > r_{max}$) **do**
    CoQ = ChainGenerate(Q, F);
    // LLM generate the new Chain-of-Query CoQ.
    T.AddChild(CoQ); *// Add the new branch to T.*
    F = Traverse(CoQ); *// Interact with IR.*
    $r = r + 1$; *// Update the number of interaction rounds r.*
  return Tracing(T, R)

知乎 @SmartMindAI

若答案a_i与检索信息不符，且Reader对答案[+]的置信度[+]f高于阈值θ，为了纠正LLM可能的错误，IR会生成一个提示模板，帮助LLM进行修正。提示的具体内容没有提供，因此无法给出模板。LLM在接收到IR反馈后，针对问题q更新答案为a_i'，构建了一个新的CoQ节点。这个过程旨在填补知识空白，提升生成内容的准确性。若遇到未解决的查询，如q_i^*，LLM会标记为"未解决"，不论f是否超过θ。无论何时遇到此类情况，LLM都会接收到检索文档d_i^*中对应的正确答案g^*，作为提示，格式化为'请考虑文档d_i^*中的信息来补充$[UnsolvedQuery]:g^*$'，以便它能自我修正。这样，LLM会利用IR提供的信息来补充自己的知识，进一步完善CoQ。

这一轮结束后，LLM根据IR提供的答案a_i^*对问题q_i^*进行了修正，生成了新的CoQ节点。这个节点作为ToR的新分支，作为整个推理过程的最新进展。跟踪机制在此处起关键作用，它确保了生成内容的可追溯性。SearChain详细记录了这个未解决节点，通过追踪正确的推理路径，找到对应于问题的检索文档。这些文档成为支持文档，以注释形式出现在生成的最终内容中，如'$SupportDoc:DocIDfor<scripttype="math/tex;mode=display">q_i^*$'，这样用户能清晰了解每个部分的来源。

这样，LLM不仅能生成准确答案，还能展示解决问题的完整思路。通过LLM对问题的修正和新CoQ的生成，SearChain实现了推理过程的跟踪，生成内容具有可追溯性。每个步骤的子片段都附有支持文档引用，让用户明确了解知识来源。这种方法巧妙地利用了LLM的复杂推理，通过与IR的

知乎

向，体现了交互式学习的优势。这种方法独特地避免了依赖监督数据和额外训练，仅利用已有的LLM和IR交互即可提供支持文档。

Experimental Setup

Datasets and Evaluation Metric

在评估中，我们选取了四个代表性的知识密集型复杂任务，如HotpotQA(HoPo)、Musique(MQ)、WikiMultiHopQA(WQA)和StrategyQA(SQA)，它们需要LLM进行多步推理。我们用ROUGE-L度量ELI5的评估，因其答案长且格式自由。对于其他任务，我们通过考察生成答案是否包含目标答案（cover-EM）来评价。在MQ和HoPo的完整数据集上，我们遵循DSP和Self-Ask策略进行评估，同时在SQA的BIG-bench集合，以及WQA、zsRE、T-REx、FEVER和ELI5的子集上也进行了部分测试。

Baselines.

我们的基准方法分为两组：

一组专注于提升LLM在复杂任务推理（CoT, CoT-SC, Auto-CoT, Recite-and-answer, Least-to-Most），

另一组加入了IR技术以增强推理（直接检索, Self-Ask, ToolFormer（利用gpt-3.5-turbo的ToolFormer操作）、React, DSP, Verify-and-Edit（与CoT-SC组合）、Tree-of-Thought）。

AgentGPT和PS则采用了计划与解决（Plan-and-Solve）策略，我们也以此设立一个基线。所有方法均针对包括HotpotQA、Musique、WikiMultiHopQA、StrategyQA在内的任务进行评估，其中对ELI5使用ROUGE-L，其他任务通过目标答案覆盖率（cover-EM）来检验。在评估中，MQ和HoPo的完整数据集遵循DSP和Self-Ask规则，而WQA、zsRE、T-REx、FEVER和ELI5则选取子集进行。

Implementation

我们的研究采用了gpt-3.5-turbo作为核心的大规模语言模型，以及基于DSP的ColBERTv2作为检索模型，IR在V100 GPU上运行。针对不同任务，如HotpotQA，我们利用2017年维基百科+做训练；其他任务则使用基于维基百科的段落集合。检索基线与SearChain配置相同。复现所有基线在gpt-3.5-turbo上进行r_{max}设为5$\alpha = 0.35$ $\theta = 1.5$。通过调整θ，我们发现1.5的置信度阈值能取得最优F1分数。关于ROUGE-L，我们根据实际观察在0.3至0.5的区间内，ROUGE阈值对ELI5任务影响不大。

Main Results

Table 1: Performance of SearChain and baselines on complex knowledge-intensive tasks. Bold denotes the best result in different settings. FC: Fact Checking, LFQA: Long-Form QA. Metric for LFQA: ROUGE-L. Metric for others: cover-EM.

	Muti-Hop QA				Slot Filling		FC	LFQA
	HoPo	MQ	WQA	SQA	zsRE	T-REx	FEV.	ELI5
Without Information Retrieval								
Direct Prompting	31.95	5.91	25.82	66.25	22.75	43.85	73.45	21.90
Auto-CoT	33.53	10.55	29.15	65.40	21.30	43.98	76.61	21.55
CoT	35.04	9.46	30.41	65.83	22.36	44.51	76.98	21.79
CoT-SC	36.85	10.02	32.68	70.84	24.74	46.06	77.15	22.05
Recite-and-answer	36.49	10.97	32.53	70.47	24.98	46.14	**77.35**	22.10
Self-Ask w/o IR	33.95	11.10	35.65	65.45	20.16	44.71	75.31	21.73
Least-to-Most	34.05	11.45	32.88	65.78	21.86	44.98	75.98	21.95
Plan-and-Solve	36.33	12.95	35.68	73.21	25.15	47.58	77.08	22.23
SearChain w/o IR	**38.36**	**13.61**	**40.49**	**75.62**	**30.14**	**52.69**	77.06	**22.54**
Interaction with Information Retrieval								
Direct Retrieval	34.09	10.22	30.01	66.78	52.29	59.28	78.25	23.40
ToolFormer	36.75	12.98	35.49	67.02	51.35	59.17	80.79	23.05
Self-Ask	40.05	14.28	39.58	67.65	50.51	59.12	79.41	23.25
Plan-and-Solve w/ IR	41.65	15.07	42.05	74.58	52.15	60.03	81.04	24.56
React → CoT-SC	43.15	15.49	40.36	70.43	53.27	60.42	80.59	24.05
Verify-and-Edit	44.03	15.57	40.83	71.09	53.95	61.10	80.67	23.80
Tree-of-Thought w/ IR	50.65	15.61	42.49	72.55	54.88	62.40	81.03	24.20
DSP	51.97	15.83	43.52	72.41	54.35	61.32	80.65	23.46
SearChain	**56.91**	**17.07**	**46.27**	**76.95**	**57.29**	**65.07**	**81.15**	**25.57**
- w/o Verification	46.11	14.70	42.67	75.98	43.58	55.46	78.79	02.98
- w/o Completion	53.05	15.86	43.64	76.53	45.78	56.03	80.03	25.02

知乎 @SmartMindAI

知乎

果。它还优越于Self-Ask和Least-to-Most，强调全局视角的推理更有效。对于(2)，在与IR交互的情况下，SearChain显著超越所有基线，通过生成全局CoQ并引导LLM沿其路径，确保连贯性。这解决了自我询问、DSP和React的问题，通过IR的判断避免误导，并通过动态路径调整，使LLM能灵活改变推理方向，这是其他基线所不具备的。

Analysis

本节深入探究SearChain相对于基线的优势。首先，我们研究SearChain如何从LLM和IR的交互中获取复杂问题的解答，揭示其知识获取的独特路径。其次，我们阐明了IR如何助力LLM在难题面前表现，强调了我们方法能有效管理IR可能的误导。然后，我们着重强调了SearChain在推理连贯性和问题追踪上的优势，表明其优于那些仅依赖单一路径的基线。最后，我们通过实证分析*证明，尽管提高了性能，SearChain的运行速度并未增加太多，体现了高效与效能的平衡。

Knowledge Decoupling

Table 2: Distribution of knowledge sources.

Knowledge Src.	HoPo	MQ	WQA	SQA
LLM	74.56%	78.83%	75.83%	94.98%
Corrected by IR	20.94%	14.60%	18.96%	2.78%
Completed by IR	4.50%	6.57%	5.21%	2.24%

知乎 @SmartNLP AI

我们研究了四个多跳问答任务的数据，将知识来源分为三类：LLM直接知识、经IR修正的错误知识和LLM不知但IR提供的补充知识。我们以ToR节点作为分析单位，统计这三种类型的比例。实验数据显示，尽管LLM提供了主要知识，但大部分都通过IR校正过。IR在有足够信心确认答案时，会修正LLM的不一致，填补LLM的盲点，减少了IR对LLM潜在的干扰，提升了信息利用效率。在StrategyQA中，由于LLM已掌握大量可检索内容，IR提供的新知识相对较少。

Positive and Negative Effects of IR on LLM

Table 3: Positive and negative effects of IR on LLM.

(a) Accuracy on $\mathbb{S}_{IR}$ and $\mathbb{S}$ (positive effect ↑).

	HoPo	MQ	WQA	SQA
w/o IR ($\mathbb{S}$)	38.36	13.61	40.49	75.62
w/o IR ($\mathbb{S}_{IR}$)	31.38	10.20	32.60	68.96
w IR ($\mathbb{S}_{IR}$)	60.86	18.49	50.52	78.42

(b) Percentage that IR misleads LLM (negative effect ↓).

	HoPo	MQ	WQA	SQA
Self-Ask	15.76	14.32	25.76	10.29
React	17.68	15.22	25.99	10.03
Plan-and-Solve w/ IR	16.42	15.25	22.31	7.59
Verify-and-Edit	9.78	10.75	16.44	6.52
Tree-of-Thought w/ IR	12.07	13.25	20.52	8.46
DSP	14.72	14.03	24.31	9.22
SearChain	6.33	6.50	12.71	5.31

知乎 @SmartNLP AI

$\mathbb{S}_{IR}$上的准确率低于整体$\mathbb{S}$，这证明LLM在需要IR支持时存在问题。相反，带IR的SearChain在$\mathbb{S}_{IR}$上的表现更好，证实了IR的有效帮助。

负面风险：文中提及，IR可能误导LLM，当两者知识不一致时。为此，我们研究了在LLM能正确解答的集合$\mathbb{S}_t$中加入IR后的错误答案比例。表(b)显示，SearChain通过策略，既能识别这些问题，又能避免误导，通过考虑IR的可信度和CoQ信息，决定是修正还是补充，成功降低了IR对LLM的潜在负面影响。

CoQ vs Baselines in Reasoning

（1）推理步骤数量：在无IR条件下，我们比较了处理复杂问题时不同方法的推理步骤。在Musique数据集上，结果显示我们的方法需要更多推理步骤，且随着问题难度增加，步骤数量也随之增长，这表明我们能更好地理解问题的复杂性。

（2）解决难题子问题：相较于基线，我们的方法强调全局推理链的构建，即便无IR辅助，LLM也能全局规划解决问题。当遇到无法解决的子问题时，非但不会停止，反而会分解或重写问题以继续推进。对比图例中的案例，CoT和Self-Ask在遇到困难就停止，而SearChain通过持续的子问题重写保持了推理流程。附录中的详细案例分析⁺进一步证实了我们在解决难题时尝试次数更多，体现了全局视角的优势。

SearChain vs New Bing in Tracing

我们通过案例研究⁺比较了SearChain和New Bing在生成内容中引用文献的性能。我们定义了两个衡量指标：

Scope of Knowledge Coverage (SKC)，表示被正确标注的知识项数量，SearChain显著优于New Bing（2.882 vs 1.143），显示其更强的知识覆盖能力；

Accuracy of Marking Position (AMP)，衡量标注位置的精确性，SearChain在0.80的准确率上领先于New Bing的0.45，证明其定位准确 。 三位专业评估人员的评估结果显示，SearChain能精确标记推理过程中的所有相关知识，而New Bing的参考文献不仅覆盖不全，定位也不准确。

Efficiency Analysis

我们对SearChain和基线在几个关键指标上进行了效率评估。结果显示，SearChain显著提升了任务性能，具体表现在输入的LLM词数n和输出的LLM生成词数m减少，同时增加了LLM与IR交互的轮数r（虽然接近，但未显著增加）。然而，总体运行时间t却出现了下降，特别是相较于基线方法，这是一个明显的改进。这些改进并未伴随运行时间的显著增加，尽管没有给出具体的数值，但趋势明确。

Table 5: Efficiency analysis.

	#n ↓	#m ↓	#r ↓	t(s) ↓	Perf. (Avg) ↑
Self-Ask	401	63	2.19	6.63	46.73
Plan-and-Solve w/ IR	450	71	1	6.05	48.89
React → CoT-SC	938	110	2.35	8.25	48.47
Verify-and-Edit	565	307	2.40	13.90	48.88
Tree-of-Thought w/ IR	622	341	2.29	13.28	50.47
DSP	1759	155	2.15	10.47	50.44
SearChain	390	189	2.21	8.52	53.29

知乎 @SmartMindAI

原文《Search-in-the-Chain: Interactively Enhancing Large Language Models with Search for Knowledge-intensive Tasks》