

SSRL: 自搜索强化学习

Yuchen Fan^{1,3,*} Kaiyan Zhang^{1,*,†} Heng Zhou^{3,*} Yuxin Zuo^{1,3} Yanxu Chen¹
 Yu Fu⁴ Xinwei Long¹ Xuekai Zhu² Che Jiang¹ Yuchen Zhang³ Li Kang³
 Gang Chen⁵ Cheng Huang¹ Zhizhou He¹ Bingning Wang⁶
 Lei Bai^{3,‡} Ning Ding^{1,3,‡} Bowen Zhou^{1,3,‡}

¹ Tsinghua University ² Shanghai Jiao Tong University ³ Shanghai AI Laboratory

⁴ University College London ⁵ CSCEC Third Bureau ⁶ WeChat AI

*Equal contributions [†]Project leader [‡]Corresponding author

✉ zhang-ky22@mails.tsinghua.edu.cn  TsinghuaC3I/SSRL

Abstract

我们研究了大型语言模型 (LLMs) 作为强化学习 (RL) 中代理搜索任务高效模拟器的潜力，从而减少对昂贵的外部搜索引擎交互的依赖。为此，我们首先通过结构化提示和重复采样量化 LLMs 的内在搜索能力，我们称之为自搜索。我们的结果揭示了 LLMs 在推理预算方面表现出强大的扩展行为，在包括具有挑战性的 BrowseComp 任务在内的问题回答基准上达到高 pass@k。基于这些观察，我们介绍了自搜索 RL (SSRL)，这通过基于格式和规则的奖励增强了 LLMs 的自搜索能力。SSRL 使模型能够在不需要访问外部工具的情况下内部迭代地优化其知识使用。实证评估表明，SSRL 训练的策略模型为搜索驱动的 RL 训练提供了一个具有成本效益且稳定的环境，减少了对外部搜索引擎的依赖，并促进了稳健的模拟到现实的转移。我们得出以下结论：1) LLMs 具备可以有效激发以实现高性能的世界知识；2) SSRL 展示了利用内部知识减少幻觉的潜力；3) SSRL 训练的模型无需额外努力即可与外部搜索引擎无缝集成。我们的研究结果突显了 LLMs 对支持更具扩展性的 RL 代理训练的潜力。

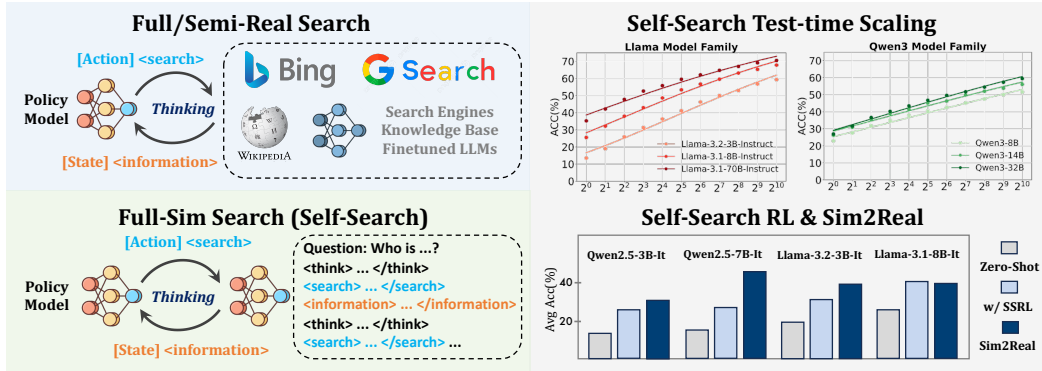


Figure 1: 左图：之前的方法，如 Search-R1 (Jin et al., 2025b) 和 ZeroSearch (Sun et al., 2025) 依赖于外部来源（例如，搜索引擎、知识库或微调的 LLMs），代表着完全或部分真实的搜索。我们提出了完全模拟搜索，其中策略模型在内部生成信息（自搜索）。右图：随着计算能力的增加，自搜索在测试时进行扩展，显示出强大的 pass@k 性能。自搜索强化学习 (SSRL) 进一步提升了不同模型和任务的结果，尤其是在模拟到真实的泛化方面。

Contents

1 自我搜索的推理时间扩展	4
1.1 任务表述	4
1.2 提示设计	5
1.3 实验设置	5
1.4 性能评估	5
1.5 进一步分析	7
1.5.1 更多的推理总是更好吗?	7
1.5.2 多数投票与通过率 @k	8
1.6 训练方法	9
1.7 实验设置	9
1.8 性能评估	10
1.8.1 从仿真到现实的泛化	11
1.8.2 测试时强化学习	13
1.9 进一步讨论	14
1.9.1 与强化学习算法的兼容性	15
2 相关工作	16
2.1 结合搜索引擎的强化学习	16
2.2 推理时刻的大型语言模型和智能体的扩展	17
3 结论	17
A 自搜索的推理时刻缩放	22
A.1 提示	22
A.1.1 重复采样说明	22
A.1.2 为大语言模型提供信息的指示	22
A.2 案例研究	24
A.2.1 Qwen3 有无思考的案例研究	24
A.2.2 多轮和反射重复采样的案例研究	24
B 自搜索强化学习	24
B.1 提示	24
B.1.1 其他算法实现	27
B.1.2 时间旅行恢复逻辑	27
B.2 消融研究	27
B.2.1 模型家族比较	27
B.2.2 通用模型与推理模型的比较	28
B.2.3 Sim2Real 搜索的附加结果	29

B.3 案例研究	29
C 格式奖励代码	30

最近,通过可验证奖励的强化学习(RL)在复杂数学问题解决和代码生成方面显著提高了大型语言模型(LLMs)的推理能力,形成了大型推理模型(LRMs)。除了数学和编码之外,众多研究还探索了在工具学习等代理背景中将RL应用于LLMs。这些方法使得LLMs能够学会调用外部工具,如网络搜索引擎,在现实环境中执行动作并观察状态。尽管像Search-R1和Kimi V2这样的最新模型在各种基准测试中表现出色,但与真实的网络搜索引擎互动仍然成本高昂,尤其是在RL训练期间需要大量的卷展和多轮工具调用时。事实上,由于在大规模网络语料库上的预训练,LLMs通常能够回答涉及世界知识的问题。一些研究还表明,LLMs可以通过提供针对给定动作的状态信息来作为世界模型。例如,最近关于ZeroSearch的研究表明,经过微调的LLM可以有效地替代网络搜索,提供稳定可靠的知识。这一发现表明,通过采用半真实的环境,搜索RL的成本可以显著降低。受到近期无监督RL技术如TTRL的启发,我们在完全模拟RL环境(标记为full-sim)中探索自搜索RL,其中在训练期间不使用真实搜索。特别是,我们关注两个关键研究问题:1)使用仅内在知识的LLMs在基于搜索的问答任务上的性能限制是什么?2)全模拟搜索RL是否能够在推理时实现与真实网络搜索的有效模拟到真实的转移?

首先,我们研究LLM是否可以基于其参数中嵌入的知识来生成查询和信息,有效地模拟查询外部搜索引擎。为此,我们通过提示模型使用多轮、工具格式化输出在单一生成轨迹内模拟搜索过程,评估需要网络搜索基准的LLMs的内在搜索能力。广泛的采样表明,LLM在其参数中编码了大量的世界知识,并产生遵循缩放规律的高预测pass@k得分。然而,可靠地提取最佳答案仍然具有挑战性,这突显了潜在知识与可操作检索之间的差距。为了应对这一挑战并探索对sim-to-real转移的完全模拟搜索RL的潜力,我们研究了自我搜索强化学习(SSRL)的潜力,该方法通过基于格式和规则的奖励增强了LLM的自我搜索能力,使其能够自主完善内部知识的利用,而无需依赖外部搜索。我们的实验表明,经过SSRL训练的模型不仅在多个基准上超越了先前基于搜索API的RL基线,如Search-R1和ZeroSearch,而且还作为成本效益高的隐含世界知识提供者,从而减少了搜索驱动的问题回答中的幻觉。此外,这种方法减少了对外部搜索引擎的依赖,并为sim-to-real泛化开辟了新的途径,使自我搜索中获得的技能能够稳健地转移到具有真实网络访问的在线环境中。

总而言之,我们的工作表明,大型语言模型(LLMs)作为网络模拟器具有重要潜力,这是一种可以用于无需外部查询的搜索驱动任务的资源。通过使用SSRL系统地量化和增强这种自我搜索能力,我们为更自主和可扩展的LLM代理铺平了道路(Leike et al., 2018; Gao et al., 2025)。

Takeaways

1. 大型语言模型可以作为世界知识的模拟器,在具有挑战性的基于搜索的基准测试中,不同的模型系列表现出不同的上限。(§ 1.4)
2. 实现自搜索的性能上限仍然具有挑战性;当前的方法,如投票,对于可靠地选择最佳输出是不足够的。(§ 1.5)
3. 通过依赖于策略模型自身的反馈,自搜索强化学习(SSRL)可以比使用外部搜索引擎进行训练取得更优异的性能。(§ ??)
4. 通过SSRL训练的模型可以无缝地适应真实的搜索场景,无需额外的努力,为模拟到现实的转移提供了初步证据。(§ 1.8.1)

1 自我搜索的推理时间扩展

1.1 任务表述

pass@k 的公式。 我们考虑仅使用一个大型语言模型的内部知识来回答信息查询的问题,而不访问诸如网络搜索引擎或数据库等外部检索工具。我们为问题 i 生成 K 个样本,并计算准确响应的数量 C_i 。我们使用下面的公式计算pass@k:

$$\text{pass@k} = \frac{1}{\# \text{ of problems}} \sum_{i=1}^{\# \text{ of problems}} \left(1 - \frac{\binom{K-C_i}{k}}{\binom{K}{k}} \right), \quad (1)$$

其中,正确性是根据底层基准的评估标准定义的(例如,精确匹配、前 k 准确性或任务特定标准)。这种设置使我们能够估计模型内化搜索能力的内在上限,而无需依赖于任何外部检索工具。

我们为测试时自搜索的缩放规律提供了详细的公式。继 Brown et al. (2024) 之后，我们定义了一个显式函数来模拟样本数量 K 和覆盖率 c 之间的相关性。我们将 c 的对数建模为 k 的函数，使用：

$$\log c \approx ak^b, \quad (2)$$

其中 a 和 b 是拟合的模型参数。我们将每一边取指数以得到覆盖率 c 的直接预测。即：

$$c \approx \exp(ak^b). \quad (3)$$

1.2 提示设计

Following Jin et al. (2025a), we use an unbiased instruction without any hints for reflection. The instruction just teaches LLMs to think step by step. The prompt template is shown in Table 1.

Prompt Template
Answer the given question. You must conduct reasoning inside <code><think></code> and <code></think></code> first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by <code><search></code> query <code></search></code> , and you should return the top searched results between <code><information></code> and <code></information></code> . You can search as many times as you want. For multi-hop QA, you can break it down into pieces and search one by one. If you find no further external knowledge needed, you can directly provide the answer inside <code><answer></code> and <code></answer></code> without detailed illustrations. For example, <code><answer></code> Beijing <code></answer></code> . Question:

Table 1: 提示模板。问题在训练和推理过程中被附加在末尾。

我们的迭代推理框架遵循一个结构化的流程，模型首先在 `<think>` ... `</think>` 标签中表达其初步想法。当模型识别出解决问题所需的缺失信息时，它会在 `<search>` ... `</search>` 标签中制定搜索查询。然后，模型自回归地生成相关信息以解决这些查询，并将其纳入 `<information>` ... `</information>` 标签内。这种思考、搜索和信息收集的循环不断迭代，直到模型得出最终答案。虽然这种方法与传统的多轮搜索系统有相似之处，但其在实现上有根本区别：我们的方法不是通过与外部系统进行真实的迭代交互，而是采用一个链式思维 (Wei et al., 2023) 过程，其中语言模型在单次前向传递中自回归地生成整个推理轨迹，包括思考、搜索查询和检索的信息。此设计使得在保持迭代搜索过程的结构化探索优势的同时，实现了高效的自包含搜索。

1.3 实验设置

我们在跨越三类问答任务的七个基准上进行评估：1) 一般问答，通过使用 Natural Questions (Kwiatkowski et al., 2019) 和 TriviaQA (Joshi et al., 2017) 测试事实知识检索；2) 多跳问答，通过 HotpotQA (Yang et al., 2018)、Musique (Trivedi et al., 2022)、Bamboogle (Press et al., 2023) 和 2WikiMultiHopQA (Ho et al., 2020) 需要跨多条信息进行推理；3) 模糊问答，通过 BrowseComp (Wei et al., 2025) 评估从各种模糊限制中提取信息。这一综合评估框架涵盖了从直接知识检索到复杂推理和信息整合的能力，提供了对模型在多种问答场景中性能的强有力评估。基准详细信息列于表 2。

模型。 为确保对重复采样效果的全面评估，我们在三个模型家族中进行实验：Qwen2.5 (Qwen et al., 2025)、Llama3 (包括 Llama-3.1 和 Llama-3.2) (Grattafiori et al., 2024)、以及 Qwen3 (Yang et al., 2025a)。我们测试的模型覆盖了从 0.6B 到 72B 的广泛参数规模。为了在所有实验中确保公平的比较，我们保持一致的采样参数，温度设定为 0.7，top-k 为 1，top-p 为 0.95，最大 token 为 8192。所使用的指令显示在附录 A.1.1 中。

1.4 性能评估

如图 2 和图 3 所示，我们观察到随着样本数量的增加，在所有基准测试中表现出一致且可预测的性能提升。值得注意的是，在 Bamboogle 上，Llama-3.1-8B-Instruct 达到 % 87.2 的准确率，相比 pass@1 性能提升了 % 150。这些显著的增益在所有三个模型系列 (Qwen2.5、Llama 和 Qwen3) 中都很明显，尤其是 Llama 系列表现出特别明显的益处。图 3 显示了

Knowledge Type	Benchmark	Time	Construction	Targeted Task	Source

Table 2: 关于搜索的基准。大多数基准是手动构建的, 除了 2WikiMultiHopQA 和 MuSiQue。大多数基准最初是为问答设计的。

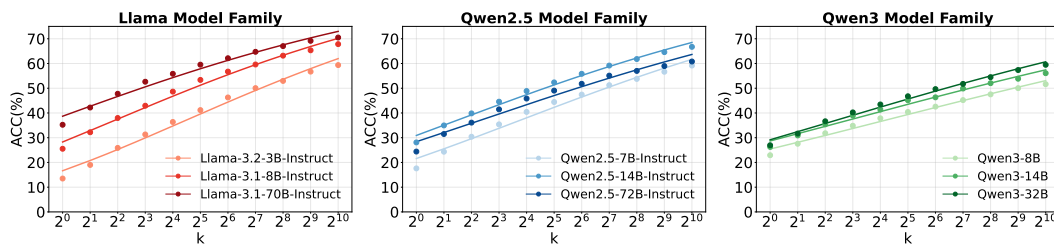


Figure 2: 三个模型家族 (Qwen2.5、Llama 和 Qwen3) 在六个基准上进行重复采样的缩放曲线。它表明预测性能的提升, 其中不同家族的平均 MAE 分别为 1.42 %、1.45 % 和 0.95 %。

在 BrowseComp 上的性能，这是一个以困难的搜索要求但验证简单为特征的基准。虽然带搜索的 GPT-4o 仅达到 % 1.9，o1 得分 % 10，但自搜索技术却带来了惊人的结果：当提供足够的样本时，Qwen2.5-14B-Instruct 和 Llama-3.1-8B-Instruct 超过了 o1 的性能。这一发现表明 LLMs 具有深厚的内在知识，即使没有外部信息来源，也能通过重复采样有效利用。对上限的分析进一步突出了 LLMs 在应对给定搜索动作时提供信息的强大潜力。更多细节见附录 ??。

Llama 表现优于 Qwen, 与先前的推理任务相反。

先前的研究 (Gandhi et al., 2025; Liu et al., 2025b; Wang et al., 2025a) 已表明, Qwen 模型包括 Qwen2.5 和 Qwen3) 在数学推理中具有更强的先验能力, 并在强化学习环境中比 Llama 模型取得更大的提升。然而, 我们的研究发现, 在自我搜索环境中, Llama 模型在世界知识的先验能力方面优于 Qwen 模型, 如图 2 和图 3 所示。这个观察表明, 自我搜索能力与推理先验能力并不强相关。我们将在第 1.5.1 节中进一步探讨知识和推理的利用。

值得注意的是，我们的结果表明，通过利用重复采样，较小的模型可以达到与参数量几乎相差 $10 \times$ 的模型相当的性能，如由 pass@k 测量。例如，在 TriviaQA 上，使用 1024 个样本时，Llama-3.1-8B-Instruct 获得 81.2 % 的得分，而 Llama-3.1-70B-Instruct 获得 81.4 %，尽管模型大小存在巨大的差距，但差异微乎其微。这一发现与以往的研究一致 (Snell et al., 2024; Li

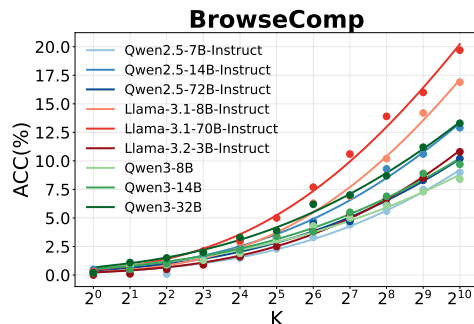


Figure 3: 在 BrowseComp 上应用 TTS 会导致所有模型的性能一致提升。这表明预测性能有所提升，其中 LLaMA、Qwen 2.5 和 Qwen 3 系列的平均 MAE 分别为 0.34 %、0.22 % 和 0.26 %。

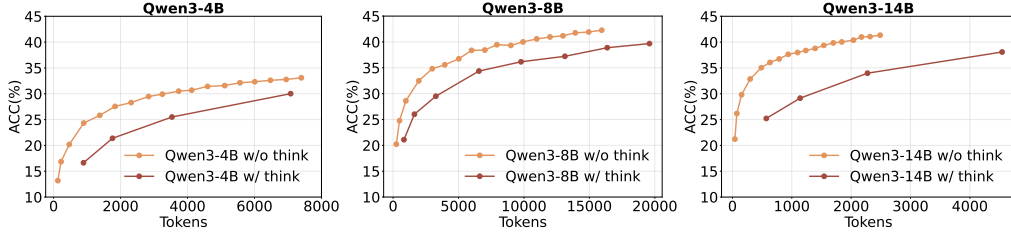


Figure 4: 在六个基准测试中，各种尺寸的 Qwen3 在启用或不启用强制思考情况下的平均表现。横轴是按照仅用于思考的标记数进行衡量的。

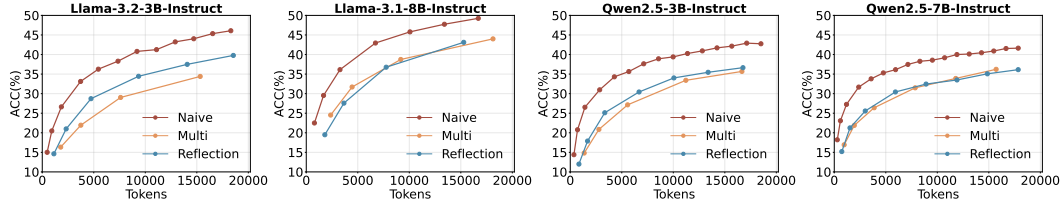


Figure 5: 在相同的令牌预算下，测量了四个模型中重复自搜索、多轮自搜索以及带反思的自搜索的性能。

1.5 进一步分析

1.5.1 更多的推理总是更好吗？

实验设置。 如在第 1.4 节中所讨论的，我们在推理和指令模型的主体性搜索基准上观察到不一致的结果，例如，Qwen3 与 Qwen2.5 和 Llama。在本节中，我们首先分析 Qwen3 模型中思考标记的利用效率，然后比较两种类型的顺序扩展：多轮搜索和多轮反思。附加案例研究见表 18。对于标记比较的实现，当解码响应达到预定义阈值时，我们不截断以限制 LLM 的响应长度，因为这可能会损害 LLM 的能力。相反，我们生成 K 响应并总结使用的标记，并在相同的标记预算下进行比较。

思维代币的低效利用 Qwen3 模型支持“思考”和“非思考”模式 (Yang et al., 2025b)，允许在模型给出最终答案之前手动调整思考 token 的数量。为了研究在自我搜索环境中增加思考 token 的影响，我们进行了一项对比研究，评估推理过程中是否启用思考过程的影响。我们仅统计用于比较思考 token 的 `<search> ... </search>`，`<信息> ... </information>`，和 `<answer> ... </answer>` 中的 token。如图 4 所示，结果表明，随着分配的 token 数量增加，长 CoT 推理并没有带来更好的表现，这与复杂数学题中的观察结果相矛盾。这可能是因为主动搜索的解决方案主要依赖于知识的使用，无论是内部的还是外部的，而不仅仅是思考。这些发现表明，在自我搜索环境中应该优先考虑短 CoT，以最大化 token 效率。

多轮自我搜索会降低性能。 按照搜索代理文献中既定的方法 (Jin et al., 2025a; Sun et al., 2025)，将搜索形式化为多轮过程，我们为每次展开执行自我搜索。在生成搜索查询后，我们提示模型提供该查询的相关信息，将这些信息整合到推理上下文中，并继续迭代推理过程，直到达成最终决定。我们将这种交互的次数记为 N 。关于让 LLM 提供相关信息的指令列在附录 A.1.2 中。由于我们的方法消除了对外部搜索引擎（如 Google、Bing 等）的需求，我们避免了通常与在线搜索相关的 API 成本和推理预算限制。因此，我们设置 $N = 10$ 以确保每个样本有足够的迭代次数以收敛到最终答案。如图 5 所示，以消耗的标记数测量，简单的重复采样表现出更好的性能和稳步的性能增长，进一步突显了 LLM 自身作为世界知识隐性模拟器的上限。

带有反思的自我搜索影响性能。 Deepseek-R1 引入的“Aha Moment” (DeepSeek-AI, 2025)，展示了大型语言模型中新兴的反思和探索能力，特别是在数学和代码生成方面。为了研究这种反思行为是否延伸到没有外部资源的信息搜索任务中，我们在采样过程中加入了触发

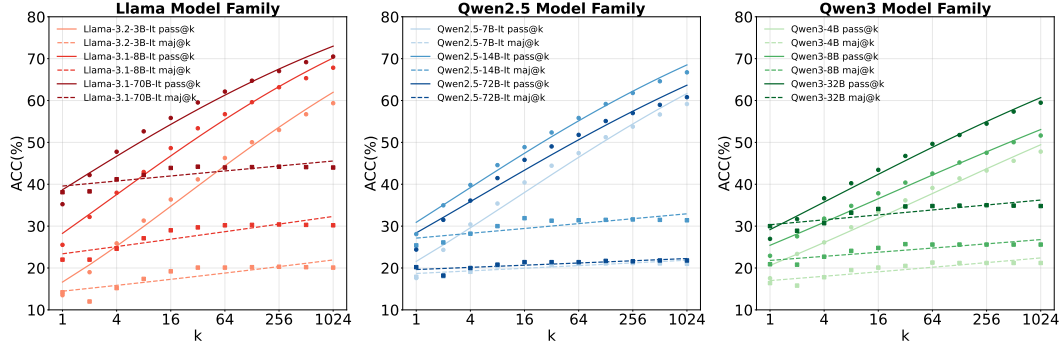


Figure 6: 不同模型在六个基准上的多数投票结果的平均值。

反思的短语。具体来说，我们在每个生成的响应后附加“等一下，等一下”以鼓励模型重新考虑并探索替代的推理路径。图 5 展示了实验结果。我们还发现，在相同的令牌预算下，与简单的重复采样相比，反思没有在 pass@k 测量下产生更好的性能。

综上所述，我们发现增加思考标记的数量和引入多轮生成在自搜索环境中并不总是有益的。这表明在这些情境下，知识利用可能比推理更具优势。特别是在将语言模型作为世界模型的背景下，需要进行进一步的研究 (Hao et al., 2023; Gu et al., 2024)。

1.5.2 多数投票与通过率 @k

在上述实验中，我们发现大型语言模型在搜索和问答任务中表现出较高的性能上限。然而，当真实情况未知时，尽管正确答案存在，但从候选答案中识别出正确答案仍然具有挑战性 (Brown et al., 2024)。这表明重复采样代表了测试时间缩放 (TTS) 的上限，进一步评估其他 TTS 策略是必要的。

多数投票被广泛认为是一种简单但有效的方法，可以与测试时缩放 (Zuo et al., 2025) 相结合。为了研究这种性能是否能够转移到知识密集型任务中，我们采用了 maj@k 指标。

本质上，当 K 样本中出现频率最高的答案与真实答案匹配时，maj@k 的值为 1，否则为 0。多数投票实验的说明详见附录 A.1.1。我们在图 6 中展示了结果。我们的实验表明，即便我们增加多数投票的回应数量 k ，性能改善也仅是微乎其微的。这种有限的扩展行为表明，简单的多数投票可能不足以应对搜索任务，因为错误答案可能在多个样本中持续出现。这些发现表明，LLMs 有潜力成为世界模型，但所呈现的世界知识是模糊的，如何提供精确的知识仍然是一个具有挑战性的任务。

在本节中，我们使用强化学习 (RL) 使大型语言模型 (LLM) 能够从它们自身的参数中导出世界知识。我们研究了在自搜索 RL 中，策略模型作为世界知识模拟器的有效性，以及它在仿真到现实 (Sim2Real) 和 TTRL 设置中的性能。

我们为利用外部搜索引擎的 LLM (大型语言模型) 搜索代理制定了强化学习目标，如下所示：

$$\max_{\pi_{\theta}} \mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot|x; R)} [r_{\phi}(x, y)] - \beta \mathbb{D}_{KL}[\pi_{\theta}(y|x; R) || \pi_{\text{ref}}(y|x; R)], \quad (4)$$

，其中 π_{θ} 表示策略模型， π_{ref} 代表参考模型， r_{ϕ} 是奖励函数， R 表示检索到的信息，而 $\mathbb{D}_{KL}$ 表示 KL 散度正则化项，系数为 β 。在我们的方法中，由于模型自动回归地从其内部参数而非外部来源检索知识，检索到的信息 R 遵循与 π_{θ} 相同的分布。这种自搜索机制使我们能够将目标函数简化为：

$$\max_{\pi_{\theta}} \mathbb{E}_{x \sim D, y \sim \pi_{\theta}(\cdot|x)} [r_{\phi}(x, y)] - \beta \mathbb{D}_{KL}[\pi_{\theta}(y|x) || \pi_{\text{ref}}(y|x)], \quad (5)$$

，其中 π_{θ} 兼作推理策略模型和内部搜索引擎。我们主要采用 GRPO (Shao et al., 2024) 作为训练算法，同时也尝试不同的强化学习算法，包括 PPO (Schulman et al., 2017)、Reinforce++ (Hu et al., 2025)、DAPO (Yu et al., 2025) 和 KL-Cov (Cui et al., 2025b)，以验证其鲁棒性。

1.6 训练方法

信息标记掩码 先前的研究 (Jin et al., 2025b; Sun et al., 2025) 表明, 从外部搜索引擎中屏蔽信息标记有助于稳定训练并提高性能。然而, 在自我搜索中, 检索到的信息来自模型自身的生成过程, 而不是外部来源, 这引发了关于在这种情况下信息屏蔽是否仍然有益的疑问。为此, 我们在两个条件下进行比较实验: 使用完整推理轨迹训练与使用信息屏蔽轨迹训练。在实现中, 我们提取由 `<information>` 和 `</information>` 包围的所有标记并在损失计算中屏蔽它们。我们的结果显示, 即使信息是模型自生成的, 信息屏蔽仍然能够提高性能。我们详细的实验结果在第 ?? 节展示。

奖励建模 根据 DeepSeek-AI (2025); Yu et al. (2025), 我们采用了一个包含两个信号的复合奖励函数: 格式奖励和结果奖励。我们直接使用模型最终预测的准确性作为结果奖励, 按照以下规则计算:

$$R(\hat{y}, y) = \begin{cases} 1, & \text{is_equivalent}(\hat{y}, y) \\ -1, & \text{otherwise} \end{cases} \quad (6)$$

其中 y 是真实答案, $\hat{y}$ 是预测答案。

由于我们的迭代搜索过程要求模型将复杂问题分解为可管理的子问题, 每次迭代专注于搜索特定信息并逐步构建最终答案, 因此维护结构化的输出格式对于有效推理至关重要。为了解决这一需求, 我们引入了一种格式奖励, 确保遵循规定的推理结构, 详见附录 C。这个格式奖励指导模型产出组织良好的、多步骤的推理轨迹。

最终奖励结合了两个组件, 如下所示:

$$r_\phi(y_i, y) = \begin{cases} 1 & \text{if is_equivalent}(\hat{y}, y) \wedge f_{\text{format}}(y) = \text{True}, \\ 1 - \lambda_f & \text{if is_equivalent}(\hat{y}, y) \wedge f_{\text{format}}(y) = \text{False}, \\ \lambda_f & \text{if !is_equivalent}(\hat{y}, y) \wedge f_{\text{format}}(y) = \text{True}, \\ 0 & \text{if !is_equivalent}(\hat{y}, y) \wedge f_{\text{format}}(y) = \text{False}, \end{cases} \quad (7)$$

, 其中我们设置 $\lambda_f = 0.1$ 以优先考虑正确性, 同时保持结构化推理, 遵循 (Wang et al., 2025b)。

1.7 实验设置

基准测试 我们在第 ?? 节中描述的六个基准上进行评估。由于 BrowseComp 的难度极大以及训练数据有限, 我们将其排除在评估之外。为了与现有基线进行公平比较, 我们采用 Sun et al. (2025) 使用的相同验证集。我们的评估采用 EM 方法, 只有在预测与真实答案完全匹配时才视为正确。这个严格的评估标准确保了模型性能的稳健评估。

基线 为评估 Self-Search 的有效性, 我们将我们的模型与以下方法进行比较: 原生提示方法, 包括直接提示和 CoT; 基于 RAG 的方法, 包括标准 RAG 和 Search-o1 (Li et al., 2025b); 基于 RL 的方法, 包括 R1、Search-R1 (Jin et al., 2025b) 和 ZeroSearch (Sun et al., 2025)。我们进行模型的离线评估, 同时在适用的情况下启用对基线方法的在线测试。为了在在线环境中确保公平比较, 我们限制了所有基于 RAG 的方法检索的段落数量为 3。对于原生提示方法, 我们使用了经过指令调优的模型, 因为它们表现出更优的提示跟随能力。基线的实现细节列在附录 ?? 中。

我们主要在 Llama 模型系列上进行强化学习实验, 具体为 Llama-3.2-3B (基础/指导) 和 Llama-3.1-8B (基础/指导), 这些模型是基于它们在重复采样条件下展示出的有效性选择的。我们使用与之前工作类似的 NQ 和 HotpotQA 训练数据集的组合, 以确保普通 QA 与多跳 QA 的结合。我们的训练框架主要利用 GRPO 作为默认算法, 同时也尝试其他方法, 包括 PPO 和 REINFORCE++, 以验证我们发现的稳健性。所有训练均在配备 8 个 NVIDIA A800 GPU 的单节点上进行。对于 GRPO, 训练配置包括一个批尺寸为 256 的大小, 学习率为 $1e-6$, 和在所有实验中进行 62 步热身。所有模型在我们的实验中最大响应长度为 4096。对于策略优化, 我们将温度设置为 1.0, 每个提示产生 5 个回合, 并应用 0.001 的 KL 散度系数。我们对每个模型进行 5 个周期的训练, 选择平均验证准确率最高的检查点进行最终评估, 确保最佳性能并防止过拟合。对于所有的评估, 我们将温度设置为 0.0。其他算法的实现细节列在附录 B.1.1。

Model	Search Engine	General QA		Multi-Hop QA				Avg
		NQ	TQ	HotpotQA	Musique	2Wiki	Bamboogle	
LLaMA-3.2-3B								
Direct Answer	∅ /-	16.2	29.6	12.6	2.0	9.2	8.0	12.9
CoT	∅ /-	26.2	44.4	16.0	5.8	10.2	21.6	20.7
RAG	∅ /G	30.0	57.6	23.4	9.6	17.6	11.2	24.9
Search-o1	∅ /G	24.2	48.4	19.4	6.0	17.4	32.0	24.6
R1-Base	-/-	28.4	44.2	22.8	7.0	28.4	11.1	23.7
R1-Instruct	-/-	35.0	52.2	21.6	11.4	17.8	20.8	26.5
Search-R1-Base	W/G	41.2	60.0	29.6	13.6	31.6	19.4	32.6
Search-R1-Instruct	W/G	37.6	53.6	21.0	8.8	20.4	27.8	28.2
ZeroSearch-Base	∞/G	43.4	63.8	32.2	13.8	35.6	15.3	34.0
ZeroSearch-Instruct	∞/G	40.2	58.0	22.8	10.4	21.4	18.1	28.5
Self-Search-Base	-/-	35.0	45.8	28.2	14.2	29.6	30.2	30.5
Self-Search-Instruct	-/-	43.8	58.4	25.0	14.2	31.6	38.4	35.2
LLaMA-3.1-8B								
Direct Answer	∅ /-	21.2	52.8	21.0	3.2	8.0	23.8	21.7
CoT	∅ /-	23.0	46.6	18.8	8.8	17.6	35.2	25.0
RAG	∅ /G	40.8	62.8	37.0	22.4	34.0	38.4	39.2
Search-o1	∅ /G	26.8	37.2	21.0	9.2	23.6	25.6	23.9
R1-Base	-/-	21.0	48.8	23.0	5.4	28.0	5.6	22.0
R1-Instruct	-/-	39.2	59.8	30.4	18.2	36.8	47.2	38.6
Search-R1-Base	W/G	41.0	62.6	40.0	25.0	37.8	36.1	40.4
Search-R1-Instruct	W/G	39.6	59.6	36.8	19.6	34.8	31.9	37.1
ZeroSearch-Base	∞/G	38.2	52.4	26.0	9.6	28.4	12.5	27.9
ZeroSearch-Instruct	∞/G	48.2	68.2	36.6	19.6	36.2	40.3	41.5
Self-Search-Base	-/-	41.0	49.6	30.0	18.4	34.4	32.8	34.4
Self-Search-Instruct	-/-	48.0	62.6	34.4	24.2	35.2	54.4	43.1

Table 3: 我们训练模型在六个基准上使用 EM 测量的主要结果。列 *Search Engine* 指的是在训练阶段和评估阶段使用的外部搜索引擎。我们用 ∅ 表示基线没有经过该阶段，并用“-”表示使用内部知识。我们用 ∞ 表示模拟 LLM，用 W 表示维基百科以简化表述。我们用 G 表示谷歌。每个模型的最大分数用 **bold** 表示。

1.8 性能评估

我们在表 3 中展示了主要的实验结果，并在附录（表 27 和表 28）中展示了案例研究。我们还对 Qwen 系列进行了实验，Qwen2.5 和 Qwen3 的结果列在附录 B.2.1 和附录 B.2.2 中。结果揭示了几个关键的见解：

SSRL 取得了卓越的性能。 我们的结果表明，训练有自回归内部检索的模型，一贯优于那些依赖外部搜索引擎的模型，无论是使用其他大型语言模型还是谷歌搜索。我们还观察到，与通过简单的链式思维提示训练的类似 R1 模型相比，其性能更佳。这些发现表明，通过精心设计的指令和奖励，语言模型可以有效地同时作为推理器和知识检索器，在没有外部依赖的情况下，从其内部参数知识中成功提取相关信息。

在相同的数据和相同的训练时长下，经过指令调优的模型相比其基础模型取得了显著更好的性能，这表明在监督微调过程中可能融入了额外的知识运算。然而，这一优势似乎是依赖于上下文的：尽管指令调优模型在利用内部知识方面表现出色，但当有外部信息源可用时，基础模型显示出更优的性能。这一发现暗示着对于内部与外部知识的利用需要不同的优化策略。

图

更大的模型表现出更好的自我搜索性能。 图 8 展示了两个模型的训练曲线。我们观察到在整个过程中训练奖励稳步增长。在早期训练阶段，响应长度和搜索次数都随着模型适应格式奖励约束而减少。在后期阶段，Llama-3.1-8B-Instruct 发展出更复杂的策略，学会分解问题并运用自我反思来提高性能，从而在我们的基准测试中取得更好的表现。

SSRL 更加高效和稳健。 图 8 展示了 ZeroSearch 和 SSRL 的训练曲线。与 ZeroSearch 相比，SSRL 展示了显著提高的训练效率，实现了训练时间的 5.53 倍缩短。此外，SSRL 在整个训练过程中表现出稳定的奖励增长而没有崩溃，表明其具有稳健的性能。尽管在训练初

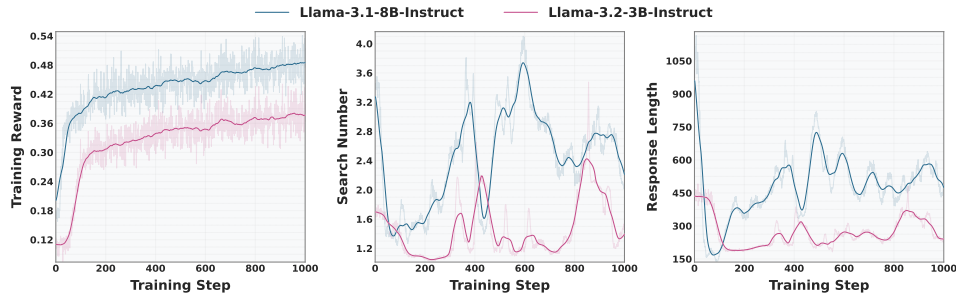


Figure 7: Llama-3.2-3B-Instruct 和 Llama-3.1-8B-Instruct 的训练曲线。第一个图是训练奖励。第二个图是响应长度，第三个图是响应中包含的搜索次数。

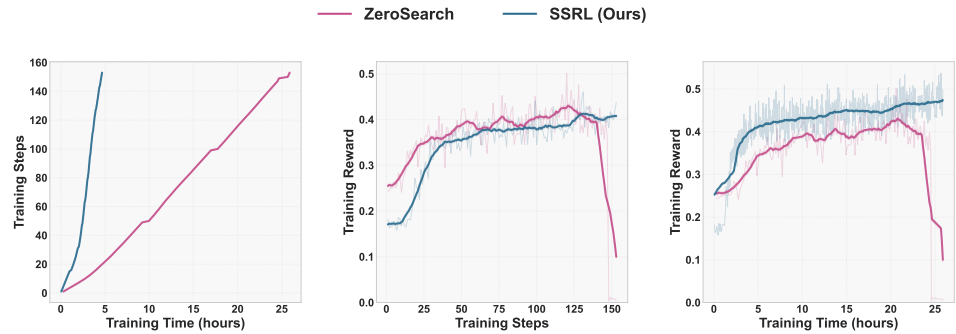


Figure 8: 使用 SSRL 和 ZeroSearch 进行 Llama-3.1-8B-Instruct 训练的曲线。第一个图是训练过程中使用的时间。第二个图是相同训练步骤上的训练奖励，第三个图是消耗相同时间的训练奖励。

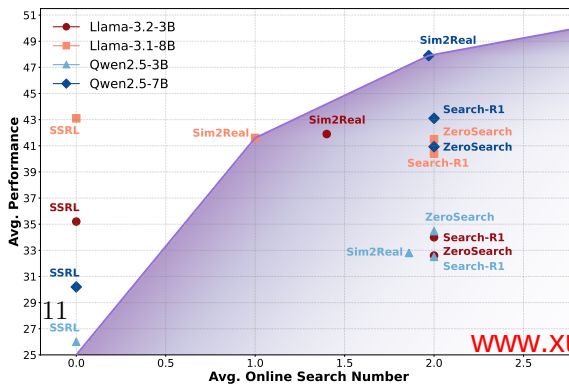
期由于外部知识有限，SSRL 显示出比 ZeroSearch 较低的训练奖励，但其卓越的效率和稳健性弥补了这种初期的不利。

1.8.1 从仿真到现实的泛化

尽管 SSRL 在静态基准测试上取得了强劲的结果，但这些模型内在的知识是固定的，这限制了它们在现实场景中的适用性。在这项工作中，我们研究了 SSRL 是否能推广到实时搜索环境。由于我们训练的模型完全符合 Search-R1 (Jin et al., 2025b) 的格式规格，我们可以无缝集成真实的搜索功能。我们将这种设置称为模拟到实际的泛化，沿用了机器人 RL (Kaspar et al., 2020; Da et al., 2025) 领域的先前术语。

我们将模型生成的信息替换为来自谷歌搜索或本地语料库的实际搜索结果，最多替换 K 个自生成的响应，其中 K 代表由 Jin et al. (2025b) 使用的最大轮次。为了确保兼容性，我们使用基于规则的修改对检索到的信息进行后处理，以去除我们训练数据中不存在的模式。表 4 和图 9 展示了我们的实验结果。除 Llama-3.1-8B-Instruct 外，对于所有模型，性能随着最大轮次的增加而持续提高。此外，与 Search-R1 和 ZeroSearch 基线相比，在 Sim2Real 设定下，基于 SSRL 的模型通常在减少各种基准测试中在线搜索的情况下，实现更优越的性能。这些发现表明，专门在内部知识上训练的搜索代理在保持格式一致时，可以有效利用外部知识来源，从而降低训练成本并提高效率。案例研究见表 29。

我们的研究表明，大语言模型 (LLMs) 具有大量的内在知识，表明它们只需在必要时进行外部搜索。基于这一见解，我们提出了一种基于熵的搜索策略。对于每个生成的序列，我们分析初始搜索查询的熵趋势：熵的增加表明模型的不确定性，从而触发外部搜索；否则，我们依赖于内部知识。我们使用 Sim2Real (All) 作为我们的基线进行



<i>Model</i>	<i>General QA</i>		<i>Multi-Hop QA</i>				<i>Avg</i>
	<i>NQ</i>	<i>TQ</i>	<i>HotpotQA</i>	<i>Musique</i>	<i>2Wiki</i>	<i>Bamboogle</i>	
<i>LLaMA-3.2-3B-Instruct</i>							
Zero-shot CoT	26.2	44.4	16.0	5.8	10.2	21.6	20.7
SSRL	43.8	58.4	25.0	14.2	31.6	38.4	35.2
Sim2Real ($K=1$)	<u>44.4</u>	63.4	34.8	17.2	37.8	42.4	40.0
Sim2Real ($K=3$)	44.8	<u>63.0</u>	35.4	<u>19.4</u>	<u>41.8</u>	47.2	41.9
Sim2Real (All)	44.0	61.6	<u>35.2</u>	20.8	42.8	<u>46.4</u>	<u>41.8</u>
<i>LLaMA-3.1-8B-Instruct</i>							
Zero-shot CoT	23.0	46.6	18.8	8.8	17.6	35.2	25.0
SSRL	48.0	62.6	<u>34.4</u>	24.2	35.2	54.4	43.1
Sim2Real ($K=1$)	39.4	<u>55.8</u>	34.0	26.8	39.8	<u>53.6</u>	<u>41.6</u>
Sim2Real ($K=3$)	33.2	50.6	29.7	23.4	<u>39.2</u>	36.6	35.5
Sim2Real (All)	<u>39.6</u>	54.6	34.6	<u>25.0</u>	36.8	50.4	40.2
<i>Qwen2.5-3B-Instruct</i>							
Zero-shot CoT	15.0	33.6	16.2	3.6	18.0	12.8	14.7
SSRL	23.6	41.0	22.4	10.4	26.0	32.8	26.0
Sim2Real ($K=1$)	<u>35.2</u>	44.0	22.0	14.8	<u>36.6</u>	<u>26.4</u>	29.8
Sim2Real ($K=3$)	37.8	51.6	<u>26.4</u>	<u>22.4</u>	36.8	21.6	<u>32.8</u>
Sim2Real (All)	37.8	<u>51.4</u>	27.4	22.4	36.4	22.4	33.0
<i>Qwen2.5-7B-Instruct</i>							
Zero-shot CoT	12.8	35.6	16.2	6.6	22.6	24.0	17.4
SSRL	31.4	44.4	26.0	11.8	31.0	36.8	30.2
Sim2Real ($K=1$)	38.4	58.0	35.6	18.4	36.0	41.6	38.0
Sim2Real ($K=3$)	43.8	<u>64.4</u>	<u>42.0</u>	29.4	53.4	54.5	47.9
Sim2Real (All)	<u>41.8</u>	65.0	43.2	<u>28.6</u>	<u>50.4</u>	<u>52.0</u>	<u>46.8</u>

Table 4: Sim2Real 搜索泛化的性能。最大分数用黑体表示。第二大分数用 underline 表示。

Model	GeneralQA		Multi-HopQA				Avg	Avg. Search
	NQ	TQ	HotpotQA	Musique	2Wiki	Bamboogle		
LLaMA-3.2-3B-Instruct								
Sim2Real (All)	44.0	61.6	35.2	20.8	42.8	46.4	41.8	1.9
Entropy-guided Search	45.2	62.4	34.6	18.6	40.0	46.4	41.2	1.5
LLaMA-3.1-8B-Instruct								
Sim2Real-guided Search (All)	39.6	54.6	34.6	25.0	36.8	50.4	40.2	2.6
Entropy	43.2	56.2	33.4	26.8	40.8	49.6	41.7	1.5
Qwen2.5-3B-Instruct								
Sim2Real (All)	37.8	51.4	27.4	22.4	36.4	22.4	33.0	3.0
Entropy-guided Search	36.4	54.4	30.4	19.6	36.8	25.6	33.9	1.8
Qwen2.5-7B-Instruct								
Sim2Real (All)	41.8	65.0	43.2	28.6	50.4	52.0	46.8	2.6
Entropy-guided Search	40.6	63.4	39.0	23.8	45.4	48.0	43.4	1.9

Table 5: LLaMA 和 Qwen2.5 模型在使用全选或基于熵的选择进行真实搜索引擎搜索时的性能。平均搜索次数是指在生成过程中使用的 `<search>` 的平均次数，即在线搜索加上自搜索（如果存在的话）。

公平比较，并且总是在性能增益的基础上为第一次查询使用外部搜索，如表 4 所示（有关第一次搜索重要性的消融研究，请参见附录 B.2.3）。我们在表 5 中展示了我们的结果。这种基于熵的策略将搜索频率减少了 20-42%，从而显著节省了成本，同时保持了与完全外部搜索相当的性能。如我们上文所述，虽然 Llama-3.1-8B-Instruct 在 Sim2Real ($K=3$) 下失败，但它比 Sim2Real(All)

表现更好，这表明 Llama-3.1-8B-Instruct 很难轻松利用外部信息，这可能是由于自我搜索收集的信息与外部信息之间的差距。这些结果增强了我们的关键发现：当大语言模型拥有相关信息并知道如何访问它时，它们可以有效利用内部知识，使得在许多情况下外部搜索变得不必要。

Algorithm	GeneralQA		Multi-HopQA				Avg
	NQ	TQ	HotpotQA	Musique	2Wiki	Bamboogle	
LLaMA-3.2-3B-Instruct							
GRPO	43.8	58.4	25.0	14.2	31.6	38.4	35.2
TTRL (w/o info)	58.6	76.4	47.2	37.2	59.4	57.6	56.1
Δ	+14.8	+18.0	+22.2	+23.0	+27.8	+19.2	+20.9
	↑ 33.8%	↑ 30.8%	↑ 88.8%	↑ 162.0%	↑ 87.9%	↑ 50.0%	↑ 59.4%
TTRL (w/ info)	57.4	74.0	45.2	36.4	60.2	56.0	54.9
Δ	+13.6	+15.6	+20.2	+22.2	+28.6	+17.6	+19.7
	↑ 31.1%	↑ 26.7%	↑ 80.8%	↑ 156.3%	↑ 90.5%	↑ 45.8%	↑ 56.0%
LLaMA-3.1-8B-Instruct							
GRPO	48.0	62.6	34.4	24.2	35.2	54.4	43.1
TTRL (w/o info)	43.0	64.0	35.6	27.2	47.0	52.0	44.8
Δ	-5.0	+1.4	+1.2	+3.0	+11.8	-2.4	+1.7
	↓ 10.4%	↑ 2.2%	↑ 3.5%	↑ 12.4%	↑ 33.5%	↓ 4.4%	↑ 3.9%
TTRL (w/ info)	49.2	67.4	35.4	40.2	48.2	52.0	48.7
Δ	+1.2	+4.8	+1.0	+16.0	+13.0	-2.4	+5.6
	↑ 2.5%	↑ 7.7%	↑ 2.9%	↑ 66.1%	↑ 36.9%	↓ 4.4%	↑ 13.0%
Qwen-2.5-3B-Instruct							
GRPO	23.6	41.0	22.4	10.4	26.0	32.8	26.0
TTRL (w/o info)	39.2	59.8	37.8	23.8	51.2	49.4	43.5
Δ	+13.2	+18.8	+15.4	+13.4	+25.2	+16.6	+17.5
	↑ 55.9%	↑ 45.9%	↑ 68.8%	↑ 128.8%	↑ 96.9%	↑ 50.6%	↑ 67.3%
TTRL (w/ info)	31.8	58.0	33.6	22.0	49.0	48.8	40.5
Δ	+8.2	+17.0	+11.2	+11.6	+23.0	+16.0	+14.5
	↑ 34.7%	↑ 41.5%	↑ 50.0%	↑ 111.5%	↑ 88.5%	↑ 48.8%	↑ 55.8%
Qwen-2.5-7B-Instruct							
GRPO	31.4	44.4	26.0	11.8	31.0	36.8	30.2
TTRL (w/o info)	40.6	63.2	40.4	28.8	53.2	64.0	48.4
Δ	+9.2	+18.8	+14.4	+17.0	+22.2	+27.2	+18.2
	↑ 29.3%	↑ 42.3%	↑ 55.4%	↑ 144.1%	↑ 71.6%	↑ 73.9%	↑ 60.3%
TTRL (w/ info)	34.6	54.8	32.6	20.2	43.0	50.4	39.3
Δ	+3.2	+10.4	+6.6	+8.4	+12.0	+13.6	+9.1
	↑ 10.2%	↑ 23.4%	↑ 25.4%	↑ 71.2%	↑ 38.7%	↑ 36.7%	↑ 30.1%

Table 6: 使用 TTRL 和 GRPO 训练的 Llama 和 Qwen 的性能。w/o info 和 w/ info 分别表示无信息掩码和有信息掩码。最大值用粗体表示。

1.8.2 测试时强化学习

考虑到无监督 RL 算法，例如 TTRL，(Zuo et al., 2025)，在数学和代码生成方面展示了极大的潜力，我们对其在 Self-Search 上的泛化能力感到好奇。我们在 Llama 系列上进行实验，使用由 NQ、TQ、HotpotQA、Musique、Bamboogle、2WikiMultiHopQA 和 BrowseComp¹ 组成的数据集。实现细节列在附录 B.1.2 中。为了确保全面的分析，我们进行了消融研究，包括有和没有信息屏蔽的情况，同时保持格式奖励组件，这对于标签投票机制仍然是必要的。

我们使用 EM 衡量结果，并在表 6 中显示实验结果。我们观察到，与 GRPO 相比，使用 TTRL 进行训练时性能更好。对于 Llama-3.2-3B-Instruct，平均性能提高了 59%。这一现象表明了 TTRL 在跨域和模型族中的泛化能力。使用 TTRL 时，我们发现无需信息掩码训练可以得到略好的结果，这与 RLVR 相矛盾。令人惊讶的是，我们发现在合并基准测试上简单地应用 TTRL 即使没有外部搜索引擎也能在 BrowseComp 上取得显著改进。在 BrowseComp 上的准确度曲线见图 13，最终性能指标在表 7 中总结。

有一个有趣的观察，小模型通过 TTRL 在 Browsecomp 上获得了更高的分数，当我们深入研究这些案例时，发现这些模型倾向于首先指出一个实体，并检查它是否符合

Models	BrowseComp
WebSailor-3B	2.0
Qwen2.5-3B-Instruct (TTRL)	3.9
Qwen2.5-3B-Instruct (TTRL-Sim2Real)	1.4
Llama-3.2-3B-Instruct (TTRL)	6.2
Llama-3.2-3B-Instruct (TTRL-Sim2Real)	4.1

¹对于 WebSailor，我们从 BrowseComp 中抽取 250 条记录，并使用了字符串匹配进行评估。只有当生成的预测在真实值中，或者真实值在预测中时，WebSailor 的响应才被认为是正确的。

Table 7: Performance on BrowseComp.

Model	GeneralQA		Multi-HopQA				Avg
	NQ	TQ	HotpotQA	Musique	2Wiki	Bamboogle	
LLaMA-3.2-3B-Instruct							
TTRL	58.6	76.4	47.2	37.2	59.4	57.6	56.1
Sim2Real	56.6	74.8	46.0	36.0	59.0	54.4	54.5
Qwen2.5-3B-Instruct							
TTRL	39.2	59.8	37.8	23.8	51.2	49.4	43.5
Sim2Real	39.8	61.2	40.2	22.8	51.8	41.6	42.9

Table 8: Sim2Real 搜索在 TTRL 上的性能泛化

所有要求，这与搜索-回答范式相反。这进一步加强了我们的观点，即大型语言模型包含的信息一旦被引出，可以应用于解决极其复杂的问题。

我们还在 TTRL 训练的模型上进行了 Sim2Real 实验，并在表 8 中展示了结果。尽管 TTRL 相比于 RLVR 实现了更好的性能，但它引入了模型过度依赖其内部知识并且难以轻松适应现实环境的偏差。我们发现，几乎所有查询都使用一个搜索查询完成，即使是 BroweseComp。因此，在一次性生成时，网页搜索引擎无法像 LLMs 一样提供灵活的信息。此外，我们观察到 TTRL 训练的模型更倾向于选择一个候选答案并验证它，而不是根据问题逐步搜索。我们还发现，它比 RLVR 更频繁地出现崩溃，这归因于策略模型不可预期的确定性行为。表 30 中提供了一个案例研究。

1.9 进一步讨论

由于所有检索到的信息都源于推理模型本身，在推理过程和信息生成上联合训练模型是一个自然的优化策略。为了测试训练完整轨迹的有效性，我们进行了在学习过程中有无信息屏蔽的训练实验。图 10 展示了对比结果。实验结果表明，信息屏蔽在各种基准测试中一致地提升了模型性能。列于附录 ?? 的训练动态分析显示，训练过程中屏蔽信息标记促使模型生成更全面和详细的推理轨迹。这种增强的能力为在不同问答任务中观察到的一致性性能提升提供了有力解释。通过在训练期间避免模型简单复制检索信息，屏蔽策略迫使模型更深入地参与推理过程本身，从而在推断时最终形成更强的解决问题能力。

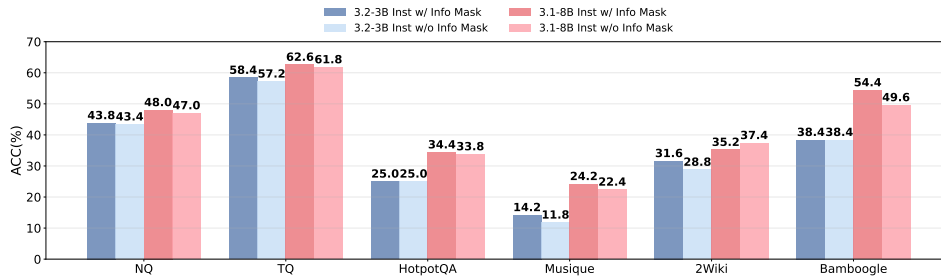


Figure 10: 在训练时有无信息掩码的情况下，Llama-3.2-3B-Instruct 和 Llama-3.1-8B-Instruct 的性能表现。

为了有效激发语言模型作为推理者和内部搜索引擎的双重能力，我们设计了一种格式奖励，强制遵循我们的结构化推理框架。这个奖励组件确保模型在其推理过程中一致地遵循思考、搜索和信息收集的预定格式。我们通过对比有无此组件训练的模型的消融研究来评估格式奖励的有效性。图 11 展示了对比结果，表明格式奖励在所有基准测试中始终提升了基础模型和经过指令调整的模型的性能。这些发现强调了结构化输出格式对于在单一模型中成功结合推理和搜索能力至关重要。格式奖励作为一个关键的支撑机制，引导模型保持有组织的推理轨迹，以促进有效的内部知识检索。没有这种结构引导，模型往往会产生不太连贯的推理路径，未能充分利用其内部搜索能力，导致整体性能下降。此外，我们观察到没有格式奖励时响应长度显著增长，这表明格式奖励稳定了训练过程。

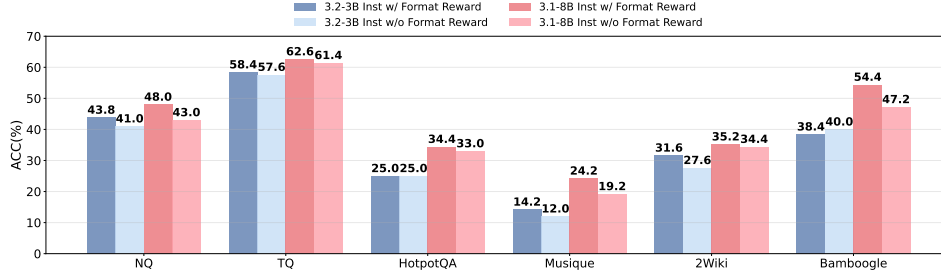


Figure 11: 在有格式奖励和没有格式奖励训练的情况下，Llama-3.2-3B-Instruct 和 Llama-3.1-8B-Instruct 的性能表现。所有的模型都通过信息屏蔽进行训练。

Algorithm	GeneralQA		Multi-HopQA				Avg
	NQ	TQ	HotpotQA	Musique	2Wiki	Bamboogle	
LLaMA-3.2-3B-Instruct							
GRPO	43.8	58.4	25.0	14.2	31.6	38.4	35.2
GRPO (Freezing)	28.6	46.6	15.8	5.6	13.8	20.8	21.9
Δ	-15.2	-11.8	-9.2	-8.6	-17.8	-17.6	-13.3
	↓ 34.7%	↓ 20.2%	↓ 36.8%	↓ 60.6%	↓ 56.3%	↓ 45.8%	↓ 37.8%
LLaMA-3.1-8B-Instruct							
GRPO	48.0	62.6	34.4	24.2	35.2	54.4	43.1
GRPO (Freezing)	24.4	46.6	15.4	7.6	17.2	23.2	22.4
Δ	-23.6	-16.0	-19.0	-16.6	-18.0	-31.2	-20.7
	↓ 49.2%	↓ 25.6%	↓ 55.2%	↓ 68.6%	↓ 51.1%	↓ 57.4%	↓ 48.0%
Qwen-2.5-3B-Instruct							
GRPO	23.6	41.0	22.4	10.4	26.0	32.8	26.0
GRPO (Freezing)	9.8	18.4	7.4	5.0	7.5	12.8	10.2
Δ	-13.8	-22.6	-15.0	-5.4	-18.5	-20.0	-15.8
	↓ 58.5%	↓ 55.1%	↓ 67.0%	↓ 51.9%	↓ 71.2%	↓ 61.0%	↓ 60.8%
Qwen-2.5-7B-Instruct							
GRPO	31.4	44.4	26.0	11.8	31.0	36.8	30.2
GRPO (Freezing)	15.6	37.4	15.0	7.2	15.2	23.2	22.6
Δ	-15.8	-7.0	-11.0	-4.6	-15.8	-13.6	-7.6
	↓ 50.3%	↓ 15.8%	↓ 42.3%	↓ 39.0%	↓ 51.0%	↓ 37.0%	↓ 25.2%

Table 9: Llama 和 Qwen2.5 在使用 on-policy GRPO 下的性能与冻结策略相比。

在之前的工作中，例如 ZeroSearch (Sun et al., 2025)，微调的 LLM 作为信息提供者。相比之下，我们将策略模型视为世界知识的隐式模拟器，以在上述部分中提供信息，这不仅简化了训练，还显著降低了训练成本，特别是与多轮运行相关的成本。

为了获得全面的理解，我们研究了两种设置：一种是信息提供者是策略本身，另一种是提供者是零步策略（即冻结的策略）。在所有训练过程中，都应用了信息掩码和格式化的奖励。我们在两个不同模型家族的四种模型上进行了实验，以评估它们的泛化能力。结果如表 9 所示，显示在大约 100 次训练步骤后出现显著的崩溃，训练奖励要么停滞不前，要么急剧下降。我们还观察到，当使用一个冻结的大型语言模型作为信息提供者时，性能显著下降。

1.9.1 与强化学习算法的兼容性

我们展示了使用不同算法训练模型时的性能，包括 PPO、GRPO、Reinforce++、DAPO 和 KL-Conv。我们使用 Llama-3.2-3B-Instruct 和 Llama-3.1-8B-Instruct 作为我们的主干。实施细节列在附录 B.1.1 中。我们在表 10 中展示了我们的结果。我们观察到不同训练算法之间存在显著的性能差距，其中基于 GRPO 的算法（例如 GRPO、DAPO 等）比 PPO 和 REINFORCE++ 表现更好。在 Sun et al. (2025) 中也观察到了 PPO 的卓越表现，证明了重复展开对搜索代理训练的有效性。值得注意的是，当使用像谷歌这样的在线引擎训练时，重复展开会导致更高的成本。然而，由于我们完全离线训练模型，因此更多的展开可能会带来更好的性能而不会增加额外成本。

Algorithm	GeneralQA		Multi-HopQA				Avg
	NQ	TQ	HotpotQA	Musique	2Wiki	Bamboogle	
LLaMA-3.2-3B-Instruct							
GRPO	43.8	58.4	25.0	14.2	31.6	38.4	35.2
DAPO	44.6	58.0	26.8	12.8	26.6	38.4	34.5
KL-Cov	41.8	58.6	24.6	12.4	28.6	38.4	34.1
REINFORCE++	42.2	55.8	25.6	12.6	32.0	30.8	33.2
PPO	35.0	55.8	21.8	11.4	29.6	30.4	30.7
LLaMA-3.1-8B-Instruct							
GRPO	48.0	62.6	34.4	24.2	35.2	54.4	43.1
DAPO	48.6	63.8	34.4	21.6	39.2	52.0	43.3
KL-Cov	44.8	63.6	32.6	22.8	37.4	52.8	42.3
REINFORCE++	46.2	64.4	33.4	18.4	43.2	36.4	40.3
PPO	37.4	58.4	27.0	17.0	38.4	37.4	36.2

Table 10: 在使用不同 RL 算法训练时, Llama-3.2-3B-Instruct 和 Llama-3.1-8B-Instruct 的性能表现。所有模型均使用信息掩码和格式奖励进行训练。

<i>Terminology</i>	<i>Explanation</i>	<i>Example</i>
Full-Real Search	Search external real engines like RAG or Google.	Search-R1 (Jin et al., 2025b)
Semi-Real Search	Search external simulated engines like LLMs.	ZeroSearch (Sun et al., 2025)
Full-Sim Search	Search internal engines, e.g., implicitly retrieving information from embedded knowledge.	Self-Search
Sim2Real Search	Train with Full-Sim Search but inference with external real engines, such as Google Search or Bing.	Self-Search

Table 11: 我们展示了本文讨论的关键概念。我们上面提到的术语是用于训练和推理过程中的搜索方法。

2 相关工作

2.1 结合搜索引擎的强化学习

强化学习 (RL) 已成为增强 LLMs (DeepSeek-AI, 2025; Cui et al., 2025a; OpenAI, 2024) 推理能力的强大方法。经过 RL 训练的推理模型, 利用过程奖励或结果奖励, 在通过自我反思和探索的复杂任务 (如数学和代码生成) 上表现出色。最近的一些研究已探索应用 RL 来提高基于 LLM 的搜索代理的性能。Search-R1 (Jin et al., 2025a) 利用 RL 训练模型, 以检索器 (如 e5) 在本地文本语料库中进行迭代搜索。同样, ReSearch (Chen et al., 2025) 仅利用结果奖励来增强 LLMs 在推理过程中寻求更多信息的能力。然而, 这些方法受限于对文本语料库的依赖, 如维基百科, 这些语料库是静态的, 并且不能充分代表现实世界在线搜索环境中的复杂性和噪声。为了解决这些限制, Zheng et al. (2025) 提出了结合浏览代理的在线搜索 RL, 将训练的模型与 Google 和 Bing 等网络搜索引擎对齐。尽管这种方法带来了优越的性能, 但其训练过程需要大量的 API 调用以进行诸如群体相对策略优化 (GRPO) (Shao et al., 2024) 的 RL 算法, 导致显著的 API 成本。作为一种具有成本效益的替代方案, ZeroSearch (Sun et al., 2025) 提出使用 LLM 作为搜索引擎模拟器创建合成的在线搜索环境, 大大减少了计算开销, 同时保持可比或更优的性能。然而, 考虑到巨量的训练数据, LLMs 作为搜索世界模型的潜力并未得到广泛探索。特别是, LLMs 作为用于代理搜索的强化学习世界模型的上限仍然未知。

随着大型语言模型 (LLMs) 的进步, 一种名为生成式搜索的新范式已经出现, 它通过生成而非传统的基于匹配的检索, 向用户提供灵活、多层次的信息。当前关于使用 LLMs 作为搜索引擎的研究主要探索了两种方法。第一种, 生成式检索, 直接生成文档标识符而没有明确匹配, 每个标识符对应于语料库中的一个特定文档。这些方法的操作假设是 LLMs 已经记住了语料库, 实际上充当了隐式知识库。第二种, 可靠响应生成, 利用 LLMs 对检索到的项目进行总结, 如论文和网页, 并生成以用户为中心的响应作为搜索结果。这些方法解决了传统信息检索系统的关键限制, 比如文档粒度刚性和相关性匹配, 同时为现实世界的应用

提供更好的灵活性、效率和创造力。根据这些应用，LLMs 有可能作为世界模型提供基于关键词的世界知识搜索。然而，关于使用 LLMs 作为代理强化学习中的文本世界模型的探索还很有限。

2.2 推理时刻的大型语言模型和智能体的扩展

重复采样是指使用概率采样从同一提示中生成多个候选输出的做法。Brown et al. (2024) 发现正确答案的覆盖率与重复样本的数量显著相关。这个发现被 Yue et al. (2025) 进一步证实，他们证明增加样本数量显著提高了正确答案的捕获百分比，即使是在诸如 AIME 等具有挑战性的基准上。在代码生成任务中也观察到了类似的缩放效应 (Li et al., 2025a)。除了简单的重复采样，这些方法还可以通过与验证机制结合来增强。最佳 N 采样 (Liu et al., 2025a; Qiu et al., 2024) 和多数投票 (Zuo et al., 2025) 都利用不同选择标准的多个样本来实现比单一贪婪解码更优的性能。尽管在推理和生成任务中取得了这些进展，在信息检索和搜索上下文中，重复采样策略的有效性仍未得到充分探索。

另一方面，最近在 TTS 方面的发展也揭示了语言基础和具身代理系统中的一个巨大且尚未充分探索的设计空间。Zhu et al. (2025) 系统地探索了语言代理的各种 TTS 策略，展示了并行采样、反思修正和多样化展开的效果。此外，像 Web 代理这样的代理展现出优越的适应性行为，如探索和回溯，当应用缩放测试时刻交互 (TTI) (Shen et al., 2025) 时，远远超过传统的每步缩放方法。除了语言代理之外，Yang et al. (2025c) 引入了一种 GUI 测试时刻缩放代理 (GTA1)，利用并发采样和评估，显著增强了图形用户界面 (GUI) 交互任务的鲁棒性，而不依赖于大量的前瞻性。补充这些策略，Lifshitz et al. (2025) 提出了多代理验证 (MAV)，多个方面验证器协同评估输出，显著提升了整体代理性能。这些最新研究共同强调了在基于代理的系统中 TTTS 的多样化方法，突显了计算资源分配和交互策略在增强不同环境中代理适应性和鲁棒行为方面的潜力。

3 结论

总之，我们的研究表明，大语言模型 (LLM) 作为隐式世界模型在搜索驱动任务中具有未开发的潜力，通常内部包含回答复杂查询所需的知识。虽然可靠地提取这些知识仍然困难，但我们提出的自搜索强化学习 (SSRL) 方法显著提升了自搜索能力，优于基于搜索 API 的基线，并实现了稳健的模拟到真实的转移。这些发现表明了一条通向更自主和可扩展的 LLM 代理的有前途的道路，使其能够在无需依赖外部搜索引擎的情况下有效运作。

References

- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL <https://arxiv.org/abs/2407.21787>.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to reason with search for llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2503.19470>.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025a.
- Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025b.
- Longchao Da, Justin Turnau, Thirulogasankar Pranav Kutralingam, Alvaro Velasquez, Paulo Shakarian, and Hua Wei. A survey of sim-to-real methods in rl: Progress, prospects and challenges with foundation models, 2025. URL <https://arxiv.org/abs/2502.13187>.

- DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Yifan Ding, Matthew Facciani, Ellen Joyce, Amrit Poudel, Sanmitra Bhattacharya, Balaji Veeramani, Sal Aguinaga, and Tim Weninger. Citations and trust in llm generated responses. In Proceedings of the AAAI Conference on Artificial Intelligence , volume 39, pp. 23787–23795, 2025.
- Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, et al. Competitive programming with large reasoning models. arXiv preprint arXiv:2502.06807 , 2025.
- Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025. URL <https://arxiv.org/abs/2504.11536>.
- Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025. URL <https://arxiv.org/abs/2503.01307>.
- Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. arXiv preprint arXiv:2507.21046 , 2025.
- Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. arXiv preprint arXiv:2305.14627 , 2023.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, and et al. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, et al. Is your llm secretly a world model of the internet? model-based planning for web agents. arXiv preprint arXiv:2411.06559 , 2024.
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. arXiv preprint arXiv:2305.14992 , 2023.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, 2020. URL <https://arxiv.org/abs/2011.01060>.
- Jian Hu, Jason Klein Liu, and Wei Shen. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models, 2025. URL <https://arxiv.org/abs/2501.03262>.
- Bowen Jin, Jinsung Yoon, Priyanka Kargupta, Sercan O. Arik, and Jiawei Han. An empirical study on reinforcement learning for reasoning-search interleaved llm agents, 2025a. URL <https://arxiv.org/abs/2505.15117>.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. arXiv preprint arXiv:2503.09516 , 2025b.
- Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL <https://arxiv.org/abs/1705.03551>.
- Manuel Kaspar, Juan D Muñoz Osorio, and Jürgen Bock. Sim2real transfer for reinforcement learning without dynamics randomization. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) , pp. 4383–4388. IEEE, 2020.

- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics* , 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL <https://aclanthology.org/Q19-1026/>.
- Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871* , 2018.
- Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E. Gonzalez, and Ion Stoica. S*: Test time scaling for code generation, 2025a. URL <https://arxiv.org/abs/2502.14382>.
- Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *ICLR* , 2023.
- Xiaoxi Li, Zhicheng Dou, Yujia Zhou, and Fangchao Liu. Corpuslm: Towards a unified language model on corpus for knowledge-intensive tasks, 2024a. URL <https://arxiv.org/abs/2402.01176>.
- Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models, 2025b. URL <https://arxiv.org/abs/2501.05366>.
- Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. From matching to generation: A survey on generative information retrieval. *ACM Transactions on Information Systems* , 43(3):1–62, 2025c.
- Yongqi Li, Xinyu Lin, Wenjie Wang, Fuli Feng, Liang Pang, Wenjie Li, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. A survey of generative search and recommendation in the era of large language models. *arXiv preprint arXiv:2404.16924* , 2024b.
- Yongqi Li, Nan Yang, Liang Wang, Furu Wei, and Wenjie Li. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence* , volume 38, pp. 8716–8723, 2024c.
- Shalev Lifshitz, Sheila A. McIlraith, and Yilun Du. Multi-agent verification: Scaling test-time compute with multiple verifiers, 2025. URL <https://arxiv.org/abs/2502.20379>.
- Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* , 2024.
- Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling, 2025a. URL <https://arxiv.org/abs/2502.06703>.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025b. URL <https://arxiv.org/abs/2503.20783>.
- Xinwei Long, Jiali Zeng, Fandong Meng, Zhiyuan Ma, Kaiyan Zhang, Bowen Zhou, and Jie Zhou. Generative multi-modal knowledge retrieval with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence* , volume 38, pp. 18733–18741, 2024.
- OpenAI. Openai o1 system card, 2024. URL <https://arxiv.org/abs/2412.16720>.
- Ivo Petrov, Jasper Dekoninck, Lyuben Baltadzhiev, Maria Drencheva, Kristian Minchev, Mislav Balunović, Nikola Jovanović, and Martin Vechev. Proof or bluff? evaluating llms on 2025 usa math olympiad. *arXiv preprint arXiv:2503.21934* , 2025.

- Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023. URL <https://arxiv.org/abs/2210.03350>.
- Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiushi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. arXiv preprint arXiv:2504.13958, 2025.
- Yujia Qin, Zihan Cai, Dian Jin, Lan Yan, Shihao Liang, Kunlun Zhu, Yankai Lin, Xu Han, Ning Ding, Huadong Wang, Ruobing Xie, Fanchao Qi, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Webcpm: Interactive web search for chinese long-form question answering, 2023. URL <https://arxiv.org/abs/2305.06849>.
- Jiahao Qiu, Yifu Lu, Yifan Zeng, Jiacheng Guo, Jiayi Geng, Huazheng Wang, Kaixuan Huang, Yue Wu, and Mengdi Wang. Treebon: Enhancing inference-time alignment with speculative tree-search and best-of-n sampling, 2024. URL <https://arxiv.org/abs/2410.16033>.
- Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and et al. Qwen2.5 technical report, 2025. URL <https://arxiv.org/abs/2412.15115>.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.
- Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, and Aviral Kumar. Thinking vs. doing: Agents that reason by scaling test-time interaction, 2025. URL <https://arxiv.org/abs/2506.07976>.
- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face, 2023. URL <https://arxiv.org/abs/2303.17580>.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL <https://arxiv.org/abs/2408.03314>.
- Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. Zerossearch: Incentivize the search capability of llms without searching, 2025. URL <https://arxiv.org/abs/2505.04588>.
- Hao Tang, Darren Key, and Kevin Ellis. Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment. Advances in Neural Information Processing Systems, 37:70148–70212, 2024.
- Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. Transformer memory as a differentiable search index. Advances in Neural Information Processing Systems, 35:21831–21843, 2022.
- Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. arXiv preprint arXiv:2507.20534, 2025.
- Harsh Trivedi, Niranjana Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition, 2022. URL <https://arxiv.org/abs/2108.00573>.

- Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, et al. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems*, 35:25600–25614, 2022.
- Zengzhi Wang, Fan Zhou, Xuefeng Li, and Pengfei Liu. Octothinker: Mid-training incentivizes reinforcement learning scaling, 2025a. URL <https://arxiv.org/abs/2506.20512>.
- Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025b. URL <https://arxiv.org/abs/2504.20073>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL <https://arxiv.org/abs/2201.11903>.
- Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models, 2024. URL <https://arxiv.org/abs/2411.04368>.
- Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents, 2025. URL <https://arxiv.org/abs/2504.12516>.
- Fengli Xu, Qianye Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. Towards large reasoning models: A survey of reinforced reasoning with large language models, 2025. URL <https://arxiv.org/abs/2501.09686>.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, and et al. Qwen3 technical report, 2025a. URL <https://arxiv.org/abs/2505.09388>.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025b.
- Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, Ran Xu, Liyuan Pan, Caiming Xiong, and Junnan Li. Gta1: Gui test-time scaling agent, 2025c. URL <https://arxiv.org/abs/2507.05791>.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018. URL <https://arxiv.org/abs/1809.09600>.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, and et al. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL <https://arxiv.org/abs/2503.14476>.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL <https://arxiv.org/abs/2504.13837>.
- Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments, 2025. URL <https://arxiv.org/abs/2504.03160>.
- King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling test-time compute for llm agents, 2025. URL <https://arxiv.org/abs/2506.12928>.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, et al. Ttrl: Test-time reinforcement learning. arXiv preprint arXiv:2504.16084, 2025.

A 自搜索的推理时刻缩放

A.1 提示

A.1.1 重复采样说明

我们使用表 12 中的指令进行重复采样。

Answer the given question. You must conduct reasoning inside `<think>` and `</think>` first every time you get new information. After reasoning, if you find you lack some knowledge, you can call a search engine by `<search>` query `</search>`, and it will return the top searched results between `<information>` and `</information>`. You can search as many times as you want. If you find no further external knowledge needed, you can directly provide the answer inside `<answer>` and `</answer>` without detailed illustrations. For example, `<answer>` Beijing `</answer>`. Question:

Table 12: 重复采样的说明。

A.1.2 为大语言模型提供信息的指示

当查询 LLM 以提供信息时，我们使用表 13 中的指令。

Given a query, you need to imitate the style of the following demos and generate five useful documents for the query.
[EXAMPLE]
You should generate documents that can help the user find the answer. Each document should contain about 30 words. You must directly output the English documents and not output any other texts.
Query: query
Useful Output:

Table 13: 为 LLM 提供信息的指南。

我们在图 12 中介绍了对七个基准进行 16 个模型的重复抽样结果。我们还在表 ?? 中列出了每个模型的模拟参数，并在表 14、15、16 中对比了实际值与拟合值、残差及不同 k 值下每个模型的相对误差。

K	Llama-3.2-3B-Instruct					Llama-3.1-8B-Instruct					Llama-3.1-70B-Instruct				
	Actual	Fitted	Residual	Rel.	Error (%)	Actual	Fitted	Residual	Rel.	Error (%)	Actual	Fitted	Residual	Rel.	Error (%)
2^0	13.50	16.64	-3.14		23.26	25.53	28.27	-2.74		10.73	35.23	38.69	-3.45		9.80
2^1	19.00	20.78	-1.78		9.35	32.20	32.87	-0.67		2.09	42.20	42.72	-0.52		1.24
2^2	25.88	25.24	0.64		2.49	37.97	37.54	0.43		1.12	47.77	46.69	1.07		2.24
2^3	31.32	29.92	1.40		4.45	42.93	42.20	0.74		1.71	52.63	50.56	2.07		3.93
2^4	36.36	34.74	1.62		4.44	48.63	46.78	1.86		3.82	55.83	54.30	1.54		2.75
2^5	41.16	39.60	1.56		3.79	53.37	51.22	2.15		4.03	59.53	57.88	1.66		2.78
2^6	46.28	44.41	1.87		4.04	56.73	55.47	1.26		2.22	62.13	61.28	0.85		1.37
2^7	50.04	49.10	0.94		1.87	59.60	59.52	0.08		0.14	64.73	64.50	0.24		0.36
2^8	52.96	53.62	-0.66		1.25	63.17	63.32	-0.15		0.24	67.03	67.52	-0.49		0.73
2^9	56.72	57.92	-1.20		2.12	65.33	66.87	-1.53		2.35	69.17	70.35	-1.18		1.71
2^{10}	59.36	61.97	-2.61		4.40	67.83	70.16	-2.33		3.43	70.50	72.98	-2.48		3.52

Table 14: 比较三种骆驼模型在不同 K 值时的实际值、拟合值、残差和相对误差。

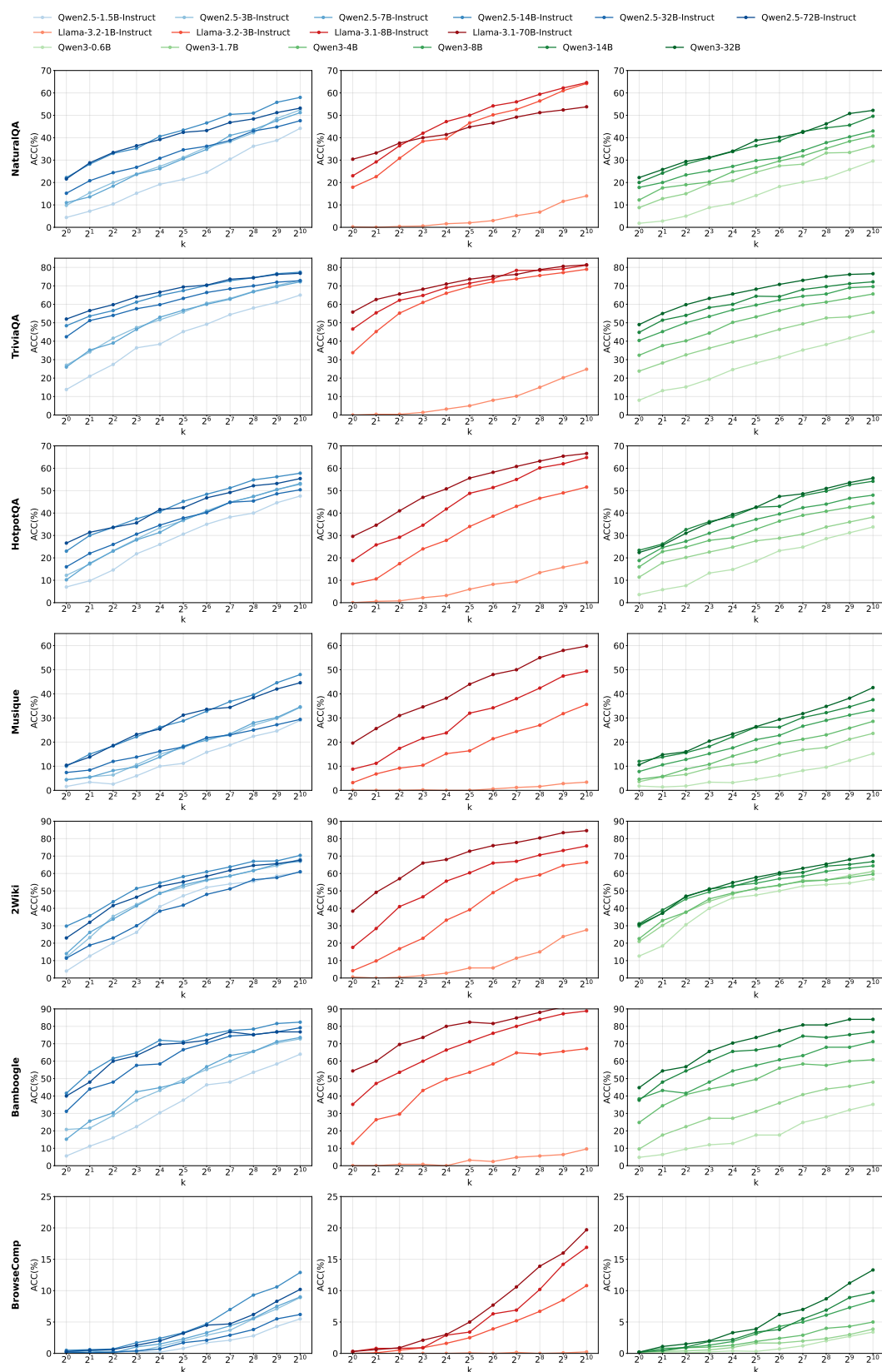


Figure 12: 对七个基准反复采样的结果

K	Qwen3-8B					Qwen3-14B					Qwen2.5-7B-Instruct				
	Actual	Fitted	Residual	Rel.	Error (%)	Actual	Fitted	Residual	Rel.	Error (%)	Actual	Fitted	Residual	Rel.	Error (%)
2 ⁰	22.92	25.42	-2.50		10.90	26.28	28.69	-2.41		9.15	17.60	21.58	-3.98		22.63
2 ¹	27.56	28.14	-0.58		2.09	30.92	31.62	-0.70		2.26	24.33	25.51	-1.18		4.83
2 ²	31.80	30.91	0.89		2.79	35.40	34.59	0.81		2.29	30.40	29.61	0.79		2.61
2 ³	34.84	33.73	1.11		3.20	38.96	37.57	1.39		3.56	35.40	33.81	1.59		4.50
2 ⁴	37.84	36.56	1.28		3.39	41.40	40.55	0.85		2.04	40.43	38.05	2.38		5.89
2 ⁵	40.40	39.39	1.01		2.50	45.16	43.51	1.65		3.65	44.43	42.28	2.16		4.85
2 ⁶	42.56	42.21	0.35		0.82	46.32	46.43	-0.11		0.23	47.43	46.44	1.00		2.10
2 ⁷	45.20	45.00	0.20		0.44	49.88	49.29	0.59		1.18	51.23	50.49	0.74		1.45
2 ⁸	47.52	47.75	-0.23		0.48	52.04	52.09	-0.05		0.09	53.77	54.39	-0.63		1.17
2 ⁹	50.04	50.44	-0.40		0.80	53.84	54.80	-0.96		1.79	56.67	58.13	-1.46		2.58
2 ¹⁰	51.64	53.07	-1.43		2.76	56.08	57.44	-1.36		2.42	59.17	61.67	-2.50		4.23

Table 15: Qwen3-8B、Qwen3-14B 和 Qwen2.5-7B-Instruct 的实际值与 拟合值、残差及相对误差对比。

K	Qwen2.5-14B-Instruct					Qwen2.5-72B-Instruct					Qwen3-32B				
	Actual	Fitted	Residual	Rel.	Error (%)	Actual	Fitted	Residual	Rel.	Error (%)	Actual	Fitted	Residual	Rel.	Error (%)
2 ⁰	28.10	30.91	-2.81		10.01	24.40	28.39	-3.99		16.34	26.96	29.16	-2.20		8.17
2 ¹	34.97	35.05	-0.09		0.25	31.50	32.09	-0.59		1.88	31.68	32.44	-0.76		2.39
2 ²	39.83	39.22	0.62		1.55	36.10	35.85	0.25		0.69	36.64	35.75	0.89		2.43
2 ³	44.57	43.35	1.22		2.73	41.47	39.62	1.85		4.45	40.24	39.07	1.17		2.91
2 ⁴	48.87	47.41	1.46		2.98	45.83	43.36	2.47		5.39	43.44	42.37	1.07		2.45
2 ⁵	52.37	51.35	1.01		1.94	49.03	47.04	1.99		4.06	46.76	45.64	1.12		2.41
2 ⁶	55.83	55.15	0.68		1.22	51.80	50.63	1.17		2.26	49.64	48.83	0.81		1.62
2 ⁷	59.17	58.78	0.39		0.65	55.10	54.10	1.00		1.81	51.76	51.95	-0.19		0.37
2 ⁸	61.80	62.22	-0.42		0.68	57.00	57.44	-0.44		0.77	54.48	54.97	-0.49		0.91
2 ⁹	64.60	65.46	-0.86		1.34	58.97	60.63	-1.66		2.82	57.36	57.89	-0.53		0.92
2 ¹⁰	66.73	68.50	-1.77		2.65	60.80	63.66	-2.86		4.70	59.48	60.69	-1.21		2.03

Table 16: Qwen2.5-14B-Instruct、Qwen2.5-72B-Instruct 和 Qwen3-32B 的实际值与 拟合值、残差和相对误差的比较。

A.2 案例研究

A.2.1 Qwen3 有无思考的案例研究

我们在表格 17 中展示了 Qwen3 在有/无思考模式下的一个案例研究。启用思考模式时，模型在开始时花费许多标记进行推理。

A.2.2 多轮和反射重复采样的案例研究

我们在表 18 中展示了多轮和基于反射的重复采样的案例研究。

B 自搜索强化学习

B.1 提示

我们在表 19 中列出了用于直接推理的指令，在表 20 中列出了用于 RAG 的指令，以及在表 21 中列出了用于类似 R1 模型训练的指令。

我们在附录 B.1 和 A.1.1 中分别展示了使用的指令。为了确保评估的一致性，我们将温度设置为 0.0。

检索增强生成 (RAG) 我们使用了附录 B.1 中列出的说明。我们使用了 Jina 和谷歌搜索。为了确保评估的一致性，我们将温度设置为 0.0。

R1 我们使用附录 B.1 中列出的指令。我们训练最多 5 个周期，如果观察到任何崩溃，包括响应长度和训练精度，我们就停止训练。我们选择表现最好的检查点。训练批量大小为 256，学习率为 1e-6。我们使用 KL 损失系数为 0.001。我们基于 GRPO 训练我们的模型，对于每个提示，我们生成 5 个响应。在评估时，我们使用温度 = 0.0。

Question: Who is the mother of the director of film Polish-Russian War (Film)?

Golden Answer: Małgorzata Braunek

Extracted Answer: Elżbieta Munk

Thinking Process (With Thinking):

<think> Okay, let's see. The question is asking for the mother of the director of the film "Polish-Russian War." First, I need to figure out who directed that film. But wait, I don't remember any film specifically called "Polish-Russian War." Maybe it's known by a different title in English? Sometimes movies have different names in different countries.

Alternatively, maybe it's a historical film about the Polish-Russian conflicts. Let me think. There are several conflicts between Poland and Russia, like the Polish-Soviet War in the early 20th century. Could the film be about that? Maybe the user is referring to a film like "The Battle of Somosierra," but that's a different conflict. Hmm.

Wait, perhaps the user is referring to a Polish film. Maybe the director is someone well-known. Let me try to search for the film "Polish-Russian War" to find the director. If I can find the director, then I can look up their mother.

So, first step: search for "Polish-Russian War film director." Let's see what comes up. If there's a film by that name, I can get the director's name. If not, maybe there's confusion with the title. Alternatively, maybe it's a documentary or a less-known film.

...

So first, I need to search for "Polish-Russian War film director" to get the correct director's name. </think>

<search> Polish-Russian War film director </search>

<information> 1. "Bitwa Warszawska" (1975) directed by Andrzej Munk - focuses on the Polish-Soviet War. 2. "The First Echelon" (1968) directed by Georgy Yungvald-Khilkevich - Soviet film about Polish-Soviet War. 3. "The War" (2015) directed by Andrzej Munk - Polish film about the Polish-Soviet War. </information>

<search> Andrzej Munk mother </search>

<information> 1. Andrzej Munk's mother was Elżbieta Munk, a Jewish woman from Warsaw who perished in the Holocaust. </information>

<search> Georgy Yungvald-Khilkevich mother </search>

<information> 1. Limited information found; focus on his career in Soviet cinema. </information>

<answer> Elżbieta Munk </answer>

Thinking Process (Without Thinking):

To answer the question, I first need to identify the director of the film Polish-Russian War and then determine who their mother is. Let me start by finding out who directed the film.

<search> Who directed the film Polish-Russian War? </search>

<information> 1. The film "Polish-Russian War" (also known as "Polsko-Rosyjska Wojna") was directed by Andrzej Wajda. 2. Andrzej Wajda was a renowned Polish film director, screenwriter, and actor, known for his contributions to Polish cinema. </information>

Now that I know the director is Andrzej Wajda, I can proceed to find out who his mother was.

<search> Who was the mother of Andrzej Wajda? </search>

<information> 1. Andrzej Wajda's mother was Jadwiga Wajda (née Bujak). 2. Jadwiga Wajda was a Polish teacher and a significant influence in Andrzej Wajda's early life. </information>

<answer> Jadwiga Wajda </answer>

Table 17: Qwen3-32B 的案例研究：有思考和无思考。

Question: Were Scott Derrickson and Ed Wood of the same nationality?

Golden Answer: yes

Extracted Answer: No, Scott Derrickson and Ed Wood are not of the same nationality, with Derrickson being Canadian and Wood being American.

Thinking Process (Multi):

`<think>` I need to consider the nationality of Scott Derrickson and Ed Wood to determine if they are of the same nationality. Scott Derrickson is known for directing movies like “Doctor Strange” and “The Exorcism of Emily Rose”. Ed Wood is known for directing cult classic films like “Plan 9 from Outer Space”. To find their nationalities, I’ll need to look into their birthplaces. `</think>`

`<search>` Scott Derrickson nationality `</search>`

`<information>` Doc 1: Scott Derrickson is a Canadian film director, screenwriter, and producer known for directing Marvel’s Doctor Strange, released in 2016. Doc 2: Born in 1966, Derrickson grew up in Ontario, Canada, before pursuing a career in filmmaking. His background in animation and visual effects influenced his work. Doc 3: Derrickson’s work often explores the intersection of faith and science. He co-created the TV series Eureka and wrote the screenplay for The Exorcism of Emily Rose. Doc 4: As a director, Derrickson’s films often combine supernatural and psychological elements. His work has been praised for its unique visual style and atmospheric tension. Doc 5: Derrickson’s production company, Film Science, has produced several films and TV series. He continues to explore the boundaries of storytelling through his work. `</information>`

`<think>` Now that I know Scott Derrickson is Canadian, I’ll look into Ed Wood’s nationality. `</think>`

`<search>` Ed Wood nationality `</search>`

`<information>` Doc 1: Ed Wood is an American filmmaker known for his low-budget horror and sci-fi films, including the cult classic Plan 9 from Outer Space. Doc 2: As a pioneer in the low-budget film industry, Wood’s work often featured unique and bizarre storylines, captivating a niche audience. Doc 3: Born in Poughkeepsie, New York, Ed Wood began making films as a teenager and developed a passion for storytelling and filmmaking. Doc 4: Wood’s most famous film, Plan 9 from Outer Space, is often cited as one of the worst films ever made, but also as a cult classic. Doc 5: Ed Wood’s life and career have been the subject of numerous documentaries and books, exploring his legacy as a filmmaker and outsider artist. `</information>`

`<answer>` No, Scott Derrickson and Ed Wood are not of the same nationality, with Derrickson being Canadian and Wood being American. `</answer>`

Thinking Process (Reflection):

`<think>` I need to determine the nationality of Scott Derrickson and Ed Wood to compare them. `</think>`

`<search>` Scott Derrickson nationality `</search>`

`<information>` Scott Derrickson is an American film director, screenwriter, and cinematographer. He was born in the United States. `</information>`

`<search>` Ed Wood nationality `</search>`

`<information>` Ed Wood was an American film director, screenwriter, and actor. He was born in the United States. `</information>`

`<answer>` Yes, Scott Derrickson and Ed Wood are of the same nationality. They are both American. `</answer>`

wait, wait, wait

`<search>` nationality American `</search>`

`<information>` Americans are people born or citizens of the United States. `</information>`

`<search>` nationality same `</search>`

`<information>` The terms “same” and “nationality” typically refer to identical or similar ethnic or cultural identity. `</information>`

`<answer>` Yes, Scott Derrickson and Ed Wood are of the same nationality. They are both American. `</answer>`

Table 18: 多轮和反射重复采样的案例研究。

Answer the given question. Provide the answer inside `<answer>` and `</answer>` without any additional information. For example, `<answer>` Beijing `</answer>` .

Table 19: 直接推理指导。

You are a knowledgeable assistant that utilizes the provided documents to answer the user's question accurately.

Question: question

Documents: documents

Guidelines:

- Analyze the provided documents to extract relevant information. Synthesize the information to formulate a coherent and accurate answer.
 - Ensure that your response directly addresses the user's question using the information from the documents.
-

Table 20: RAG 的指令。

零搜索 我们使用附录 A.1.1 中列出的指令。我们使用与 R1 相同的设置。最大轮数是 2。模拟 LLM 是 Simulation_LLM_google_14B。开始阈值是 0.0，结束阈值是 0.5。

搜索-R1 我们使用与 R1 相同的设置。在训练时，我们使用 e5 作为检索器，使用 Wikipedia 作为语料库。在测试时，我们使用 Google 搜索进行一致性比较。

B.1.1 其他算法实现

PPO 我们训练最多 5 个周期，如果观察到任何崩溃，包括响应长度和训练精度，我们就停止训练。我们选择表现最好的检查点。训练批量大小为 256，学习率为 1e-6。我们使用 KL 损失系数为 0.001。评论模型的学习率为 1e-5。我们使用标准 GAE 作为优势估计器，带有 $\gamma = 1.0$ 和 $\lambda = 1.0$ 。

REINFORCE++ 我们训练最多 5 个周期，如果观察到任何崩溃，包括响应长度和训练精度，我们就停止训练。我们选择表现最好的检查点。训练批量大小为 256，学习率为 1e-6。我们使用 KL 损失系数为 0.001。

DAPO 我们训练最多 5 个周期，如果观察到任何崩溃，包括响应长度和训练精度，我们就停止训练。我们选择表现最好的检查点。训练批量大小为 256，学习率为 1e-6。我们不使用 KL 损失。低剪辑比率为 0.2，高剪辑比率为 0.28。我们根据准确性过滤组。

KL-协方差 我们最多训练 5 个周期，如果观察到任何崩溃，包括响应长度和训练精度，我们就停止训练。我们选择表现最好的检查点。训练批次大小为 256，学习率为 1e-6。我们不用 KL 损失。我们使用 0.2 的 k 百分比。

B.1.2 时间旅行恢复逻辑

我们将最大提示长度设置为 1024，最大响应长度设置为 3076。批次大小为 8，对于每个提示，我们展开 32 次。展开温度设置为 0.6。我们使用 5e-7 的学习率和 62 的热身步骤。我们去除了原始论文中使用的 KL 损失。我们训练 80 个周期，并在性能收敛时停止。

B.2 消融研究

B.2.1 模型家族比较

Answer the given question. Provide the answer inside `<answer>` and `</answer>` . For example, `<answer>` Beijing `</answer>` . Let's search step by step. You can break the question into pieces and answer one by one.

Table 21: R1 培训说明

Model	GeneralQA		Multi-HopQA				Avg
	NQ	TQ	HotpotQA	Musique	2Wiki	Bamboogle	
Qwen2.5-3B-Instruct							
Search-R1-base	40.6	60.0	29.2	11.2	32.0	12.5	30.9
Search-R1-inst	35.8	55.8	33.2	7.6	26.0	12.5	28.5
ZeroSearch-base	43.0	61.6	33.8	13.0	34.6	13.9	33.3
ZeroSearch-inst	41.4	57.4	27.4	30.0	9.8	11.1	29.5
Self-Search-Base	26.2	38.0	21.8	8.4	30.2	24.0	24.8
Self-Search-Instruct	23.6	41.0	22.4	10.4	26.0	32.8	26.0
Qwen2.5-7B-Instruct							
Search-R1-Base	43.4	61.4	31.2	18.2	35.2	27.8	36.2
Search-R1-Instruct	42.4	63.4	32.8	17.4	33.2	26.4	35.9
ZeroSearch-Base	42.4	66.4	32.0	34.0	18.0	33.3	37.7
ZeroSearch-Instruct	43.6	65.2	34.6	18.4	35.2	27.8	37.5
Self-Search-Base	28.8	44.2	25.0	11.4	30.4	35.2	29.0
Self-Search-Instruct	31.4	44.4	26.0	11.8	31.0	36.8	30.2

Table 22: Qwen2.5 模型在一般问答和多跳问答任务上的表现。

Model	GeneralQA		Multi-HopQA				Avg
	NQ	TQ	HotpotQA	Musique	2Wiki	Bamboogle	
Qwen2.5							
Qwen2.5-3B-Instruct	23.6	41.0	22.4	10.4	26.0	32.8	26.0
Qwen2.5-7B-Instruct	31.4	44.4	26.0	11.8	31.0	36.8	30.2
Qwen3							
Qwen3-4B	22.0	37.4	21.8	7.6	24.2	34.4	24.7
Qwen3-8B	27.0	45.2	27.0	10.8	31.8	36.0	29.6

Table 23: Qwen2.5 和 Qwen3 模型在常规问答和多跳问答任务上的表现。

由于在数学或代码任务中，Qwen 普遍被认为是比 Llama 更强的基础模型，我们希望了解当它依赖内部知识来回答知识密集型问题时，这一结论是否成立。我们也使用默认设置来训练 Qwen。所有的训练都包含格式、奖励和信息标记屏蔽。实验结果列在表 22 中。虽然我们仍然观察到与 Llama 相同的训练模式，例如缩放效应和指令模型的优越能力，但绝对性能相对低于 Llama 系列，表明 Qwen 作为世界知识模拟器的能力不如 Llama。这个发现与推理任务的趋势相矛盾，比如数学和代码生成，在这些任务中，Qwen 总是被认为是首选的基础模型。

B.2.2 通用模型与推理模型的比较

LRMs 在推理任务如数学和代码生成方面表现出色。然而，很少有工作继续训练 LRMs 以适应其他领域。为了全面了解，我们比较了一般模型和推理模型之间的 RL 性能。我们使用 Qwen2.5 和 Qwen3 进行比较。实验结果如表 23 所示。我们发现 Qwen3 的性能通常低于 Qwen2.5。回忆图 2，Qwen3 的上界也低于 Qwen2.5。这些发现表明，用过多的数学或代码生成数据训练的推理模型可能难以轻松迁移到其他领域。我们还注意到，在训练过程中存在较差的指令跟随能力，导致搜索数量下降，在训练的后期阶段降至 0。然而，这也可能归因于某个提示的格式奖励，这与 Qwen3 系列的初始工具调用格式相矛盾。

我们在图 14 中展示了有无信息掩码情况下的训练动态。实验结果表明，信息掩码显著增强了模型的搜索行为活动。这表明信息掩码机制鼓励模型执行更多的搜索操作，可能提高模型在复杂任务中的推理能力。

对于 GRPO，我们将组大小设置为 5，如 Jin et al. (2025b) 所示。在这一部分，我们分析组大小对训练动态和最终性能的影响。我们训练 5 个周期，并在最终性能收敛时停止，并选择验证分数最高的检查点。我们使用上面提到的默认设置。我们在 Qwen2.5-3B-Instruct 和

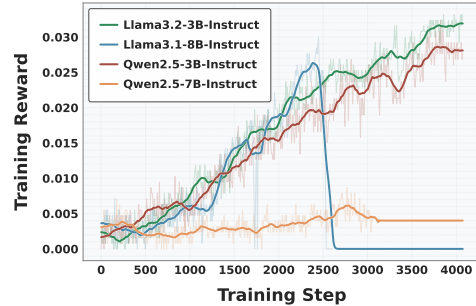


Figure 13: 各种模型在 BrowseComp 上的 TTRL 性能。

<i>Model</i>	<i>GeneralQA</i>		<i>Multi-HopQA</i>				<i>Avg</i>
	<i>NQ</i>	<i>TQ</i>	<i>HotpotQA</i>	<i>Musique</i>	<i>2Wiki</i>	<i>Bamboogle</i>	
<i>LLaMA-3.2-3B-Instruct</i>							
Group Size = 5	43.8	58.4	25.0	14.2	31.6	38.4	35.2
Group Size = 10	44.0	57.8	27.0	12.0	31.4	40.8	35.5
<i>Qwen2.5-3B-Instruct</i>							
Group Size = 5	23.6	41.0	22.4	10.4	26.0	32.8	26.0
Group Size = 10	26.2	37.8	22.6	8.4	27.0	24.8	24.5

Table 24: LLaMA 和 Qwen2.5 模型在不同组大小训练下的性能。

LLaMA-3.2-3B-Instruct 上进行实验。我们在图 14 中展示了训练曲线，并在表 24 中展示了结果。我们观察到在更大的组大小训练时具有可比的性能，但收敛速度更快。

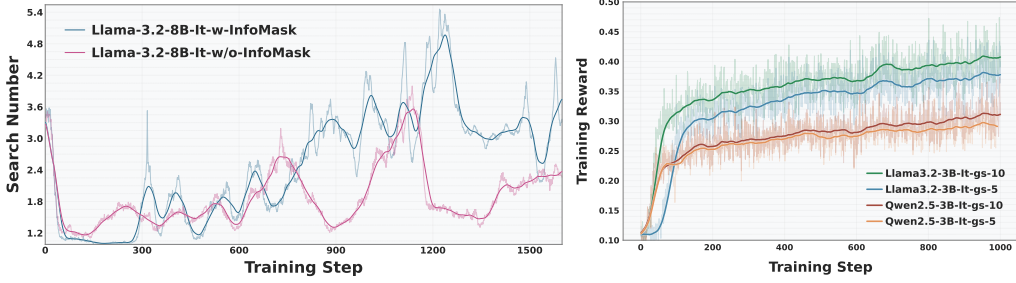


Figure 14: 左：在 Llama-3.1-8B-Instruct 上对比有无信息掩码的搜索数量。右：组大小比较。

为了全面评估自包含搜索，我们进一步在 SimpleQA 上进行离线搜索、在线搜索（我们简化舍弃 $K = 3$ ）和熵引导搜索的评估。我们从 SimpleQA 中抽取了 200 个记录。结果列在表 26 中。我们发现仅利用内部知识无助于完成像 SimpleQA 这样任务，可能是因为 SimpleQA 对于模型来说过于挑战，无法从它们的参数中检索事实知识。然而，当访问外部知识库时，我们的模型在这种复杂任务中仍然表现出巨大潜力，表明 Self-Search 在实际场景中优于组织搜索查询和基于收集的信息进行推理，即便是在完全模拟环境中训练的情况下。

B.2.3 Sim2Real 搜索的附加结果

为了测试首次检索的重要性，我们进行了一个两阶段生成实验，其中我们修改生成的响应，使用检索信息替换第一或最后的信息部分，然后重新生成以获得最终答案。实验结果在表格 25 中展示。结果清楚地表明确保首次检索及相关信息质量的重要性。也就是说，首次检索和相关信息作为成功的检索和回答轨迹的锚定点。我们进一步展示了在表格 5 中的熵引导搜索和 Sim2Real 搜索的实验结果。

B.3 案例研究

我们在表 27 和表 28 中提供了 SSRL 训练的 Llama-3.2-3B-Instruct 和 SSRL 训练的 Qwen2.5-7B-Instruct 的案例研究，用于通用问答和多跳问答。之后，我们展示 Sim2Real 推理的案例研究，提供带有真实搜索引擎的模型，在表 29 中。最后，我们在表 30 中展示了 TTRL 训练的模型在 BrowseComp 上的案例研究。

<i>Model</i>	<i>GeneralQA</i>		<i>Multi-HopQA</i>				<i>Avg</i>
	<i>NQ</i>	<i>TQ</i>	<i>HotpotQA</i>	<i>Musique</i>	<i>2Wiki</i>	<i>Bamboogle</i>	
<i>LLaMA-3.2-3B-Instruct</i>							
Replace First	44.4	63.4	34.8	17.2	37.8	42.4	40.0
Replace Last	41.0	59.6	24.8	12.8	32.2	39.2	34.9
<i>LLaMA-3.1-8B-Instruct</i>							
Replace First	39.4	55.8	34.0	26.8	39.8	53.6	41.6
Replace Last	47.4	62.2	34.4	22.2	39.0	49.6	42.5
<i>Qwen2.5-3B-Instruct</i>							
Replace First	33.8	49.6	28.2	12.0	33.6	28.0	30.9
Replace Last	23.2	37.0	22.8	7.4	29.2	35.5	25.9
<i>Qwen2.5-7B-Instruct</i>							
Replace First	35.8	56.6	34.0	17.0	34.8	40.8	36.5
Replace Last	28.2	45.2	25.4	11.4	30.2	28.8	28.2

Table 25: 使用真实搜索引擎在第一个或最后一个搜索步骤替换检索信息时，LLaMA 和 Qwen2.5 模型的性能。

<i>Model</i>	<i>SimpleQA</i>
<i>LLaMA-3.2-3B-Instruct</i>	
SSRL	4.5
Sim2Real	45.5
Entropy-guided Search	43.0
<i>LLaMA-3.1-8B-Instruct</i>	
SSRL	7.0
Sim2Real	35.0
Entropy-guided Search	30.0
<i>Qwen2.5-3B-Instruct</i>	
SSRL	2.7
Sim2Real	51.0
Entropy-guided Search	45.0
<i>Qwen2.5-7B-Instruct</i>	
SSRL	4.0
Sim2Real	51.0
Entropy-guided Search	48.5

Table 26: LLaMA 和 Qwen2.5 模型在 SimpleQA 任务上的表现。

C 格式奖励代码

本节中，我们提供了用于判断展开序列是否符合搜索代理所需格式的代码。

Listing 1: Format Reward Code.

```
def format_reward(response: str) -> float:
    response = response.strip()

    # Check if any tag content contains disallowed tags
    allowed_tags = {'think', 'search', 'information', 'answer', '/think',
                    '/search', '/information', '/answer'}
    all_tags = re.findall(r'<([>]+)>', response)
    for tag in all_tags:
        if tag not in allowed_tags:
            return 0.0

    # Must start with <think> and end with </answer>
```

Question: Who got the first nobel prize in physics?

Answer: Wilhelm Conrad Röntgen

Thinking Process:

<think> The question is asking who received the first Nobel Prize in Physics. I need to find out who that is. </think>

<search> first Nobel Prize in Physics </search>

<information> 1. The first Nobel Prize in Physics was awarded in 1901 to Wilhelm Conrad Röntgen for his discovery of X-rays. 2. The first Nobel Prize in Physics was awarded in 1901 to Wilhelm Conrad Röntgen. 3. The first Nobel Prize in Physics was awarded in 1901 to Wilhelm Conrad Röntgen for his discovery of X-rays. </information>

<think> The first Nobel Prize in Physics was awarded in 1901 to Wilhelm Conrad Röntgen for his discovery of X-rays. </think>

<answer> Wilhelm Conrad Röntgen </answer>

Question: If both Dictyosperma and Huernia are described as a genus

Answer: Yes

Thinking Process:

<think> The question is asking if both Dictyosperma and Huernia are described as a genus. </think>

<search> Dictyosperma genus </search>

<information> 1. Dictyosperma is a genus of plants in the asparagus family, Asparagaceae. 2. Dictyosperma is indeed a genus of plants. </information>

<think> Now we need to check if Huernia is also a genus. </think>

<search> Huernia genus </search>

<information> 1. Huernia is a genus of succulent plants in the family Apocynaceae. 2. Huernia is indeed a genus of plants. </information>

<think> Both Dictyosperma and Huernia are described as a genus. </think>

<answer> Yes </answer>

Question: Who plays the mouse in Alice in Wonderland

Answer: Barbara Windsor

Thinking Process:

<think> The question is asking who plays the mouse in Alice in Wonderland. I need to find out who plays the mouse in the 2010 film adaptation of Alice in Wonderland. </think>

<search> who plays the mouse in alice in wonderland </search>

<information> 1. According to Wikipedia, the mouse in the 2010 film Alice in Wonderland is played by Barbara Windsor. 2. According to IMDb, the mouse in the 2010 film Alice in Wonderland is played by Barbara Windsor. 3. According to Wikipedia, the mouse in the 2010 film Alice in Wonderland is played by Barbara Windsor. </information>

<think> The mouse in the 2010 film Alice in Wonderland is played by Barbara Windsor. However, I need to verify if this information is correct. </think>

<search> alice in wonderland mouse </search>

<information> 1. According to IMDb, the mouse in the 2010 film Alice in Wonderland is played by Barbara Windsor. 2. According to Wikipedia, the mouse in the 2010 film Alice in Wonderland is played by Barbara Windsor. 3. According to IMDb, the mouse in the 2010 film Alice in Wonderland is played by Barbara Windsor. </information>

<think> The mouse in the 2010 film Alice in Wonderland is played by Barbara Windsor. </think>

<answer> Barbara Windsor </answer>

Table 27: Self-Search -Llama-3.2-3B-Instruct 在一般问答和多跳问答上的案例研究。

Question: When was the last time Philadelphia won the Super Bowl?
Golden Answers: Super Bowl LII, 2017
Extracted Answer: 2017
Solution Process:

<think> First, we need to identify the last Super Bowl win for the Philadelphia team.
</think>
<search> last time philadelphia won the superbowl </search>
<information>
- The Philadelphia Eagles last won the Super Bowl in 2017, defeating the New England Patriots in Super Bowl LII.
- The Philadelphia Eagles won Super Bowl LII 41-33.
- The last time the Philadelphia Eagles won the Super Bowl was in 2017.
</information>
<think> Based on the information, the last time the Philadelphia Eagles won the Super Bowl was in 2017. </think>
<search> year philadelphia eagles last won superbowl </search>
<information>
- The Philadelphia Eagles last won the Super Bowl in 2017.
- The 2017 Super Bowl was Super Bowl LII, which the Eagles won.
- The Philadelphia Eagles have not won a Super Bowl since 1980, but their last win was in 2017.
</information>
<think> Confirming the last Super Bowl win for the Philadelphia Eagles is indeed in 2017.
</think>
<search> last superbowl win philadelphia team </search>
<information>
- The Philadelphia Eagles last won the Super Bowl in 2017.
- The last time the Philadelphia 76ers won an NBA championship was in 1983.
- The last time the Philadelphia Flyers won the Stanley Cup was in 1975.
</information>
<think> Based on all the information, the last time the Philadelphia Eagles won the Super Bowl was in 2017. </think>
<answer> 2017 </answer>

Question: Who is older, Aryeh Ben-Eliezer or Jason Pociask
Answer: Aryeh Ben-Eliezer
Thinking Process:

<think> First, we need to determine the birth years of both individuals to compare their ages. </think>
<search> Who is older, Aryeh Ben-Eliezer or Jason Pociask </search>
<information>
- Aryeh Ben-Eliezer was born on 15 October 1937.
- Jason Pociask was born on 19 July 1971.
</information>
<think> Now we know the birth years of both individuals. Aryeh Ben-Eliezer was born in 1937, while Jason Pociask was born in 1971. Clearly, Aryeh Ben-Eliezer is older. </think>
<search> Birth year of Aryeh Ben-Eliezer </search>
<information>
- Aryeh Ben-Eliezer was born on 15 October 1937. </information>
<think> Confirming the birth year of Aryeh Ben-Eliezer, we can now definitively state that he is older than Jason Pociask. </think>
<answer> Aryeh Ben-Eliezer </answer>

Table 28: Self-Search -Qwen2.5-7B-Instruct 在一般问答和多跳问答中的案例研究。

Question: Who was president of the United States in the year that Citibank was founded?

Answer: James Madison

Ground Truth: james madison

Thinking Process:

<think> First, we need to determine the year Citibank was founded to identify the president at that time. </think>

<search> Year Citibank was founded </search>

<information> - Citibank was founded in 1812 as City Bank of New York, and later became First National City Bank of New York. </information>

<think> Now that we know Citibank was founded in 1812, we need to find out who was the president of the United States in 1812. </think>

<search> Who was president in 1812 United States </search>

<information> - James Madison (March 16, 1751 [O.S. March 5, 1750] –June 28, 1836) was an American statesman, diplomat, and Founding Father who served as the fourth. </information>

<think> James Madison was the president in 1812, the year Citibank was founded. </think>

<search> James Madison presidency year 1812 </search>

<information> - Although he was accused of weakness in dealing with France and England, Madison won the presidency in 1808 by publishing his vigorous diplomatic dispatches. - American statesman, diplomat, and Founding Father who served as the fourth president of the United States from 1809 to 1817. - Fourth President 1809-1817. Fast Fact: James Madison, “Father of the Constitution,” led the inconclusive War of 1812. </information>

<think> James Madison was the president from 1809 to 1817, and he was the president during the year Citibank was founded in 1812. </think>

<search> Who was president in 1812 United States </search>

<information> - James Madison was the fourth president of the United States, serving from 1809 to 1817. </information>

<think> Based on the information, James Madison was the president of the United States in 1812 when Citibank was founded. </think>

<answer> James Madison </answer>

Question: What physical force is used by the machine that extracts honey from honeycombs?

Answer: centrifugal force

Thinking Process:

<think> First, we need to identify the machine used for extracting honey and the physical force it uses. </think>

<search> machine used to extract honey from honeycombs physical force </search>

<information> - A honey extractor extracts the honey from the honey comb without destroying the comb. Extractors work by centrifugal force. A drum or container holds a frame. - A honey extractor is a device used to extract honey from the comb by spinning it at high speed. The centrifugal force causes the honey to be flung out of. - Honey extractors are simple mechanical devices that extract honey from honeycombs. They use centrifugal force in a drum or container with frame. </information>

<think> Based on the information, the machine used to extract honey from honeycombs uses centrifugal force. </think>

<search> physical force used in honey extractor </search>

<information> - A honey extractor is a piece of equipment used by beekeepers in the extraction of honey from honeycombs. A honey extractor works by using centrifugal force. - Extractors work by centrifugal force. A drum or container holds a frame basket which spins, flinging the honey out. With this method the wax comb stays intact. - Drum. The drum is the central component that holds the frames. It spins to create centrifugal force, which helps remove honey from the comb. </information>

<think> It is clear that the physical force used in a honey extractor is centrifugal force. </think>

<search> centrifugal force in honey extractor </search>

<information> - Extractors work by centrifugal force. A drum or container holds a frame basket which spins, flinging the honey out. With this method the wax comb stays intact. - A honey extractor uses centrifugal force to extract honey from the honey comb without destroying the comb. </information>

<answer> centrifugal force </answer>

Question: A professional football player completed a transfer to a new club in the 2010s, in exchange for another player. This player later described his time at the new club negatively, and seven articles were written about him by Football Insider in the years 2016 and 2017. He was released from his contract by a club founded in the 1920s more than 3 years after this transfer. He entered a new coaching role in 2020 at a football club in a different country known for their attacking style. The player for which he was exchanged ended his career at a football club representing a town that has a castle. How many league goals did the first player score for the club that he left in this transfer?

Golden Answer: 76

Extracted Answer: 76

Thinking Process:

`<think>` I need to find the professional football player who completed the transfer in the 2010s. I will search for the player's name. `</think>`

`<search>` Professional football player transferred in 2010s `</search>`

`<information>` The player is Daniel Sturridge, who transferred from Manchester City to Liverpool in 2013. He was released by Liverpool in 2019. He entered a new coaching role in 2020 at Middlesbrough. He ended his career at Middlesbrough, a football club representing a town with a castle. According to his career statistics, Daniel Sturridge scored 76 league goals for Manchester City. `</information>`

`<think>` Since I found the player's name, I can now find the number of league goals he scored for the club that he left. `</think>`

`<answer>` 76 `</answer>`

Question: A professional football player completed a transfer to a new club in the 2010s, in exchange for another player. This player later described his time at the new club negatively, and seven articles were written about him by Football Insider in the years 2016 and 2017. He was released from his contract by a club founded in the 1920s more than 3 years after this transfer. He entered a new coaching role in 2020 at a football club in a different country known for their attacking style. The player for which he was exchanged ended his career at a football club representing a town that has a castle. How many league goals did the first player score for the club that he left in this transfer?

Golden Answer: 76

Extracted Answer: 105

Thinking Process:

`<think>` I need to find the professional football player who completed the transfer in the 2010s. I will search for the player's name. `</think>`

`<search>` Professional football player transferred in 2010s `</search>`

`<information>` Heading into the 2010s, Cristiano Ronaldo's €94 million move from Manchester United to Real Madrid was the world record transfer. This list of the top 10 best college football transfers dating back to 2010 to come into creation from best to worst, beginning from one to ten. `</information>`

`<think>` Since I found the player, I can now find the club he was released from and the coaching role he entered. I can also find the club he ended his career at. I will search for the number of league goals he scored for the club that he left. `</think>`

`<answer>` 105 `</answer>`

Table 30: 在 BrowseComp 上的 TTRL 训练的 Llama-3.2-3B-Instruct 案例研究。

```

if not (response.startswith('<think>') and response.endswith('</
answer>')):
    return 0.0

# Extract all tags in order
tags = re.findall(r'<(/?(?:think|search|information|answer))>',
    response)

# Check if any tag content is empty
tag_contents = {
    'think': re.findall(r'<think>(.*?)</think>', response, re.DOTALL),
    'search': re.findall(r'<search>(.*?)</search>', response, re.
        DOTALL),
    'information': re.findall(r'<information>(.*?)</information>',
        response, re.DOTALL),
    'answer': re.findall(r'<answer>(.*?)</answer>', response, re.
        DOTALL)
}

if len(tags) < 4:
    return 0.0
# Return 0 if any tag has empty content
for tag_type, contents in tag_contents.items():
    for content in contents:
        if not content.strip():
            return 0.0
        if tag_type == 'search' and len(content.split('\n')) != 1:
            return 0.0
        if tag_type == 'search' and 'your query' in content.lower():
            return 0.0
        if tag_type == 'think' and 'your thoughts' in content.lower():
            return 0.0
        if tag_type == 'answer' and 'your answer' in content.lower():
            return 0.0
        if tag_type == 'information' and 'your information' in
            content.lower():
            return 0.0

# Check structure
if tags[0] != 'think' or tags[1] != '/think':
    return 0.0

if tags[-2] != 'answer' or tags[-1] != '/answer':
    return 0.0

# Check search-information pairing in the middle
middle_tags = tags[2:-2] # Exclude initial think and final answer

i = 0
while i < len(middle_tags):
    if middle_tags[i] == 'search':
        # Must be followed by /search, information, /information
        if (i + 3 >= len(middle_tags) or
            middle_tags[i + 1] != '/search' or
            middle_tags[i + 2] != 'information' or
            middle_tags[i + 3] != '/information'):
            return 0.0
        i += 4
    else:
        i += 1

think_num = response.count('<think>')

```

```
search_num = response.count('<search>')
information_num = response.count('<information>')
if search_num != information_num:
    return 0.0

max_turn = 2
score = 1.0 / max_turn * think_num
ratio = 1.0

upper_bound = 8
if think_num != search_num + 1:
    ratio = min(think_num, search_num + 1) / max(think_num,
        search_num + 1)

return min(score, 1.0) * ratio if think_num <= upper_bound else 0.0
```