Search and Refine During Think: Autonomous Retrieval-Augmented Reasoning of LLMs

arxiv.org/abs/2505.11277

在SEARCH + TIR推理时，让LLM对检索结果先精炼再推理

开源代码：https://github.com/syr-cn/AutoRefine

Yaorui Shi[1*], Sihang Li[1*], Chang Wu[1], Zhiyuan Liu[2], Junfeng Fang[2], Hengxing Cai[3†], An Zhang[1], Xiang Wang[1†],

[1] University of Science and Technology of China
[2] National University of Singapore
[3] DP Technology

简介

本文提出AutoRefine，简单来说，在TIR (Tool-Integrated Reasoning)任务中当tool是search tool时，作者考虑到检索返回的文档会包含不相关信息，对于推理没有帮助，为此让llm先对检索文档做refine，可以理解为一种信息摘要/过滤/压缩，然后再继续推理。为了更好的让llm做refine，还设计了refine reward。

背景

本文属于TIR (Tool-Integrated Reasoning)方向的工作，注意本文的tool是向量检索模型，也算是一种search tool吧，在之前search tool类型的TIR工作中，基本上都是把检索的文档数据直接拼接到llm的prompt中，让llm继续做reasoning，作者这为检索得到的文档中含有和推理不相关的信息，这些不相关信息对推理可能并没有什么用，因此设计了一个refine stage，即所谓的“Search-and-Refine-During-Think”。

实验设置

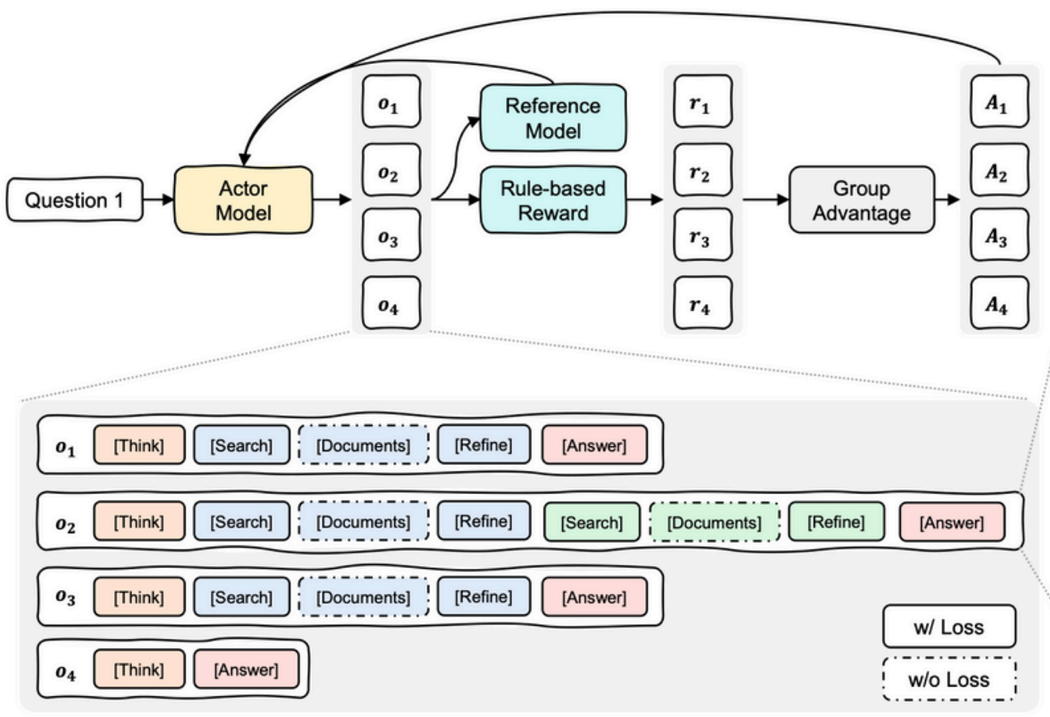
任务：单跳和多跳QA

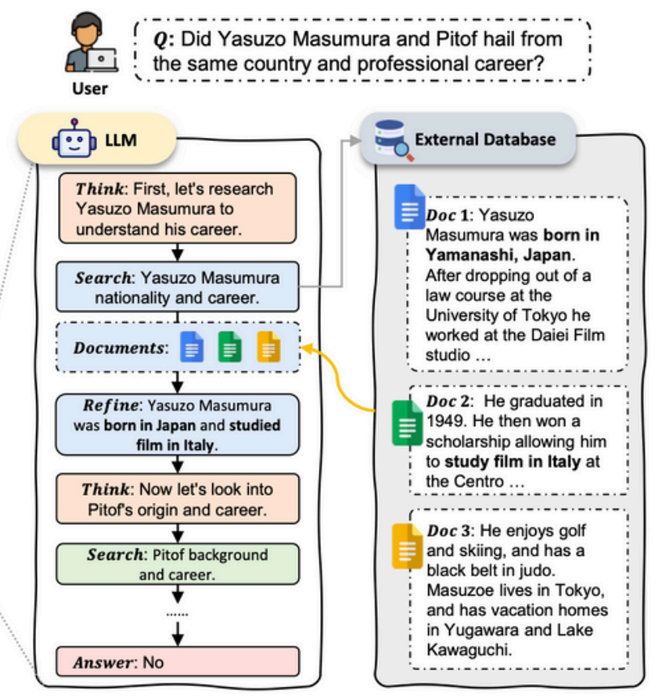
- 框架：verl，实验对象：Qwen2.5-3B Base/Instruct，强化学习算法：GRPO
- RLVR reward function: 1) answer是否正确的answer reward，计算的是预测answer和ground truth的F1值；2）二值型的refine信息reward

$$\mathcal{R}_{\text{Ans}} = \text{F1}(o_{\text{ans}}, a) = \frac{2|o_{\text{ans}} \cap a|}{|o_{\text{ans}}| + |a|} \qquad \mathcal{R}_{\text{Ret}} = \mathbb{I}(a \cap o_{\text{refine}} = a)$$

$$\mathcal{R}_{\text{Overall}} = \begin{cases} \mathcal{R}_{\text{Ans}}, & \text{if } \mathcal{R}_{\text{Ans}} > 0 \\ 0.1, & \text{if } \mathcal{R}_{\text{Ans}} = 0 \text{ and } \mathcal{R}_{\text{Ret}} > 0 \\ 0. & \text{if } \mathcal{R}_{\text{Ans}} = \mathcal{R}_{\text{Ret}} = 0 \end{cases}$$

训练流程





一般的search 类型TIR推理轨迹包含<think>, <search>, <doc>, <answer>等tag，本文多了个<refine>

部分实验结果

Table 1: **(RQ1)** Accuracy comparison of AutoRefine versus baseline methods with Qwen2.5-3B [3] across various QA benchmarks. **Bold** denotes best results, and underline denotes second best results.

	Single-Hop QA			Multi-Hop QA				
Methods	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	Musique	Bamboogle	Avg.
w/o Retreival								
Direct Generation	0.106	0.288	0.108	0.149	0.244	0.020	0.024	0.134
SFT	0.249	0.292	0.104	0.186	0.248	0.044	0.112	0.176
R1-Instruct [5]	0.210	0.449	0.171	0.208	0.275	0.060	0.192	0.224
R1-Base [5]	0.226	0.455	0.173	0.201	0.268	0.055	0.224	0.229
w/ Single-Hop Retrieval								
Naive RAG [37]	0.348	0.544	0.387	0.255	0.226	0.047	0.080	0.270
w/ Multi-Hop Retrieval								
Search-o1 [18]	0.238	0.472	0.262	0.221	0.218	0.054	0.320	0.255
IRCoT [34]	0.111	0.312	0.200	0.164	0.171	0.067	0.240	0.181
ReSearch-Instruct [21]	0.365	0.571	0.395	0.351	0.272	0.095	0.266	0.331
ReSearch-Base [21]	0.427	0.597	0.430	0.305	0.272	0.074	0.128	0.319
Search-R1-Instruct [19]	0.397	0.565	0.391	0.331	0.310	0.124	0.232	0.336
Search-R1-Base [19]	0.421	0.583	0.413	0.297	0.274	0.066	0.128	0.312
AutoRefine-Instruct	0.436	0.597	0.447	0.404	0.380	**0.169**	0.336	0.396
AutoRefine-Base	**0.467**	**0.620**	**0.450**	**0.405**	**0.393**	0.157	**0.344**	**0.405**

思考

对于search tool返回的文档问题，之前读过的ZeroSearch也考虑过，毕竟互联网上的信息质量参差不齐，所以解决文档质量问题是很有必要的，当然本文并非用搜索引擎从互联网检索信息，而是用向量模型从本身就高质量的Wikipedia数据(还是2018年dump)中做检索，我个人认为更多的是从过滤不相关数据角度提升效果而非质量，由此也延伸出一个问题，如果对搜索引擎返回的文档让llm去refine，是否合适？特别是那些对于llm来说是新知识的数据，对于不懂的知识，如何去判断相关/质量呢？