SD-Search: Hindsight Self-Distillation for Search-Augmented Reasoning

Zihan Liang[*], **Yufei Ma**[*], **Zhipeng Qian**[*], **Ben Chen**[†], **Huangyu Dai**
Lingtao Mao, **Xuxin Zhang**, **Chenyi Lei**[†], **Wenwu Ou**

Kuaishou Technology
benchen4395@gmail.com

Abstract

Search-augmented reasoning agents interleave internal reasoning with calls to an external retriever, and their performance relies on the quality of each issued query. However, under outcome-reward reinforcement learning, every search decision in a rollout shares the same trajectory-level reward, leaving individual queries without step-specific credit. Recent process-supervision approaches address this gap by drawing step-level signals from outside the policy, relying either on a much larger teacher model, or on sub-question annotations produced by a stronger external system. In contrast, we propose **SD-Search**, which derives step-level supervision *from the policy itself* through hindsight self-distillation, requiring neither an external teacher nor additional annotations. In SD-Search, a single model plays two roles that differ only in conditioning: a student that sees only the context available at inference time, and a teacher that additionally conditions on a compact hindsight block summarizing the search queries and final outcomes of a group of rollouts sampled from the same question. Since the teacher knows how each rollout unfolded and which ones succeeded, its query distribution implicitly marks which decisions were worth making, and the student is trained to recover this behavior by minimizing the token-level Jensen–Shannon divergence to the teacher at search-query positions. This layers a dense, step-level signal on top of GRPO's coarse trajectory reward. Crucially, this signal is produced by the policy itself within the standard RL training loop, without external model inference, auxiliary annotation pipeline, or additional training stage. On seven single-hop and multi-hop QA benchmarks, SD-Search reaches 0.428 average Exact Match with Qwen2.5-3B, matching the leading process-supervision baseline without its external teacher, and 0.476 with Qwen2.5-7B, surpassing all outcome-reward and process-supervision baselines on the average.

1 Introduction

Large language models (LLMs) have achieved strong reasoning performance, yet on knowledge-intensive tasks they remain limited by what was available at pretraining and cannot incorporate information that emerged afterwards [14, 18, 26, 3, 31]. A recent line of work addresses this by training LLMs as search-augmented reasoning agents that interleave internal reasoning with calls to an external retriever, autonomously deciding when to search and what to query for [11, 4, 25, 20, 35]. These agents are typically optimized with reinforcement learning such as GRPO [22, 21], using a trajectory-level reward derived from the correctness of the final answer. This formulation has driven

[*]Equal Contribution.
[†]Corresponding Author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

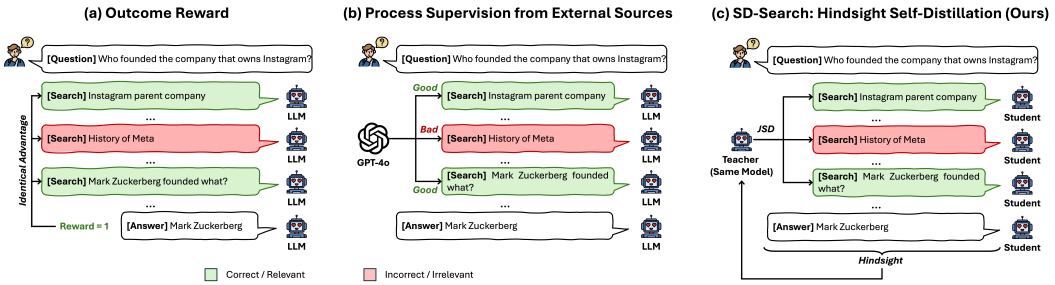


Figure 1: Three paradigms for supervising search decisions. **(a) Outcome Reward (e.g., AutoRefine):** a single trajectory-level reward, failing to distinguish between good and bad individual queries. **(b) Process Supervision (e.g., Thinker, StepSearch):** step-level signals imported from external sources (a 72B teacher or GPT-4o-generated sub-question annotations). **(c) SD-Search (Ours):** hindsight self-distillation produces step-level supervision directly from the policy itself, without external resources.

steady progress on standard QA benchmarks, but individual search decisions, which largely determine what evidence the model gets to reason over, receive no step-specific credit.

Several recent methods attempt to close this gap by injecting step-level supervision into the training signal. Thinker [33] decomposes problems into sub-questions with a 72B teacher, then supervises a student through SFT on the resulting decomposed trajectories, while StepSearch [30] derives step-wise rewards from decomposition annotations produced by GPT-4o [18]. These approaches deliver clear improvements, but they share a common dependency: the step-level signal is imported from a stronger external system, through either distillation data or auxiliary annotations, which adds cost and restricts applicability to settings where such resources exist.

We make a different observation. The policy itself, once given access to how its rollouts actually unfolded and which ones succeeded, is in a much better position to judge which earlier decisions were worth making than the same policy at decision time. This suggests that the step-level signal from external sources can instead be recovered from the policy's own predictions under an appropriately constructed hindsight context. Building on this, we propose **SD-Search** (Figure 1), a reinforcement learning framework that supplies step-level supervision through *hindsight self-distillation*. The signal is derived entirely from the policy itself, not requiring external teacher or annotations beyond the question-answer pairs used for outcome-reward training. The policy is queried under two conditioning contexts on the same trajectory: a *student* context, which contains only the prefix available at inference time, and a *teacher* context, which additionally exposes a compact hindsight summary of the search queries and outcome labels of a group of rollouts sampled from the same question. With hindsight, the teacher's predictive distribution over the search-query tokens implicitly encodes which queries were worth issuing. The student is then aligned to this distribution via a token-level Jensen-Shannon divergence at the query positions, providing a dense step-level signal that complements the coarse trajectory-level advantages of GRPO. This group-level conditioning is a deliberate design choice. Instead of only showing teacher the single rollout being supervised, we expose it to a contrast between successful and unsuccessful rollouts. This comparison sharpens the per-token target and ensures the hindsight signal remains informative, even if the supervised rollout ends in failure.

We evaluate SD-Search on seven single-hop and multi-hop QA benchmarks with both Qwen2.5-3B and Qwen2.5-7B [26]. At the 3B scale, SD-Search improves over the strongest outcome-reward baselines, AutoRefine [25] and MR-Search [32], by 2.3 and 1.4 points in average Exact Match respectively, and matches the leading process-supervision baseline Thinker [33] without its 72B teacher or supervised distillation stage. At the 7B scale, SD-Search surpasses every baseline we compare against, exceeding AutoRefine by 2.1 and Thinker by 2.4 points on the seven-benchmark average. Our contributions are summarized as follows:

- We propose **SD-Search**, a framework that supplies step-level supervision for search decisions through hindsight self-distillation, using no external teacher and no annotations beyond those required for outcome-reward training.
- We introduce a token-level Jensen-Shannon objective that aligns the policy's query distribution at each search step with the distribution produced under hindsight conditioning. The

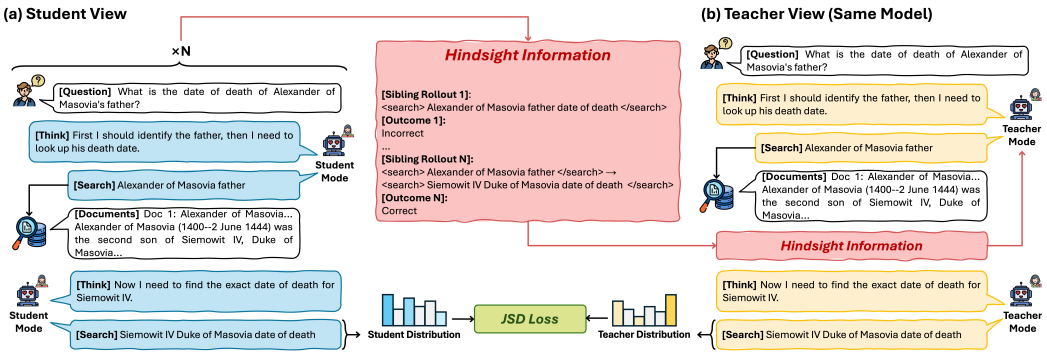


Figure 2: Method overview of SD-Search. The policy acts as its own teacher by conditioning on a **Hindsight Information** block containing the search trajectories and outcomes of sibling rollouts. The student (a), restricted to standard inference-time context, is aligned to the teacher (b) distribution at search-query positions using a token-level JSD loss, providing dense step-level supervision.

objective integrates into GRPO by adding a single auxiliary forward pass and one loss term, leaving the advantage estimator untouched.

- We conduct extensive experiments on seven QA benchmarks at both 3B and 7B scales, showing consistent gains over outcome-reward baselines and matching or surpassing process-supervision baselines, with ablations that isolate the contribution of each design.

2 Method

This section describes the components of **SD-Search**. We first review the trajectory format and the GRPO objective inherited from prior work (§2.1). Then we describe how an asymmetric student-teacher pair is constructed from the same policy by varying only the conditioning context (§2.2), formulate the token-level Jensen-Shannon objective that aligns the two (§2.3), and detail how this objective is integrated with GRPO (§2.4). An overview is provided in Figure 2.

2.1 Preliminaries

Trajectory Format. Following recent work on search-augmented reasoning [11, 25], the policy π_θ generates structured trajectories by interleaving reasoning with retrieval. A trajectory τ consists of typed spans: THINK for internal reasoning, SEARCH for issuing a query to a fixed retriever $\mathcal{E}$, DOCUMENTS for the retrieved passages, and ANSWER for the final prediction. We write $\mathcal{A}_\tau$ for the set of token positions generated by the policy, that is, all positions inside think, search, and answer spans; document tokens are excluded as they are produced by the retriever. We further write $\mathcal{Q}_\tau \subseteq \mathcal{A}_\tau$ for the subset of positions that fall strictly between the `<search>` and `</search>` tags (i.e., search query tokens), which our self-distillation objective will target in §2.3.

GRPO with Trajectory-Level Reward. We adopt GRPO [22] for policy optimization. For each question q, a group of trajectories $\mathcal{G} = \{\tau_1, \ldots, \tau_G\}$ is sampled from the policy $\pi_{\theta_{\text{old}}}$, and this group will later serve as the hindsight source in §2.2. Each trajectory receives a scalar reward R_i based on the F1 score between the predicted and gold answers. The group-normalized advantage shared by all tokens in τ_i is

$$\hat{A}_i = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G)}. \tag{1}$$

The policy is optimized with the standard GRPO loss

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}\Big[\min\Big(r_p(\theta)\,\hat{A}_i,\ \text{clip}(r_p(\theta),\,1-\epsilon,\,1+\epsilon)\,\hat{A}_i\Big)\Big], \quad r_p(\theta) = \frac{\pi_\theta(a_p \mid \tau_{<p})}{\pi_{\theta_{\text{old}}}(a_p \mid \tau_{<p})}, \tag{2}$$

where $r_p(\theta)$ is the importance ratio. Since this reward is trajectory-level, *every token in τ_i is weighted by the same advantage $\hat{A}_i$ when updating the policy*, regardless of whether it belongs to a well-formed search query or a redundant one.

2.2 Self-Distillation under Hindsight Conditioning

As outlined in §1, SD-Search uses the same policy under two conditioning contexts: a *student view* that contains only the inference-time context, and a *teacher view* that additionally conditions on a hindsight information block as shown in Figure 2. We now make these two views precise.

Student View. The student is the policy under its standard inference-time conditioning. For any action token at position $p \in \mathcal{A}_\tau$, the student distribution is

$$P_p^{\text{stu}} \;=\; \pi_\theta(\cdot \mid \tau_{<p}), \tag{3}$$

where $\tau_{<p}$ denotes the trajectory tokens preceding position p.

Teacher View. The teacher operates on the same parameters θ but on a context augmented with a *hindsight block* $h(\mathcal{G})$. We write the resulting context abstractly as

$$c^{\text{tch}}(\tau_i; \mathcal{G}) \;=\; \big[\, h(\mathcal{G});\; \tau_i \,\big], \qquad h(\mathcal{G}) \;=\; \{\mathcal{M}(\tau_j),\; y(\tau_j)\}_{j=1}^{G}, \tag{4}$$

where $[\,\cdot\,;\,\cdot\,]$ denotes that $h(\mathcal{G})$ is causally placed before the action positions of τ_i that we supervise; the concrete token-level layout, including where the block sits relative to the question prefix, is given in Appendix I. Concretely, the hindsight block aggregates all rollouts that GRPO already samples for the question. For each τ_j, it contains a masked view $\mathcal{M}(\tau_j)$ that retains only the search spans of τ_j, together with a binary outcome label $y(\tau_j) \in \{\text{CORRECT}, \text{INCORRECT}\}$. The label is derived by thresholding the F1 score of τ_j's final answer at ρ, which we set to 0 by default so that any non-empty overlap with the gold answer counts as CORRECT. For any action token at position p in τ_i, the teacher distribution at the corresponding position is

$$P_p^{\text{tch}} \;=\; \text{sg}[\,\pi_\theta(\cdot \mid h(\mathcal{G}),\; \tau_{i,<p})\,], \tag{5}$$

where $\tau_{i,<p}$ denotes the trajectory tokens of τ_i before position p, and $\text{sg}[\cdot]$ indicates that P_p^{tch} is treated as a constant target when computing gradients for the student. The teacher forward over $c^{\text{tch}}(\tau_i; \mathcal{G})$ yields the teacher distributions $\{P_p^{\text{tch}}\}_{p \in \mathcal{Q}_\tau}$ that the self-distillation objective in §2.3 aligns the student to, with $h(\mathcal{G})$ causally visible at every supervised position. At inference time, where the hindsight is unavailable, the two roles collapse back into a single policy without teacher forward pass.

Future Masking. We define the masking operator $\mathcal{M}$ as one that retains only the search spans of a trajectory and discards all other span types (THINK, DOCUMENTS, ANSWER). Since the dropped spans are downstream of each search within the trajectory, we refer to this procedure as *future masking*. Retaining the full trajectory in the hindsight block leaks the answer. Once the teacher conditions on the retrieved DOCUMENTS and downstream THINK spans, the gold answer may become directly extractable from the hindsight prefix, and the teacher's search-position distributions collapse onto retrieval-skipping continuations, pushing the student to issue fewer searches rather than better ones. The masking operator $\mathcal{M}$ prevents this collapse; we quantify its severity in §3.3.1.

Outcome Conditioning. Attaching the outcome label $y(\tau_j)$ to each rollout in the hindsight information turns the conditioning into a contrast. Conditioned on a CORRECT label, the teacher tends to reaffirm the query tokens that led to success; conditioned on an INCORRECT label, it redistributes mass away from the failed queries and toward query patterns shared with the successful siblings. The student, which sees the same τ_i prefix without any hindsight, inherits both effects by matching the teacher through the divergence objective introduced in §2.3. In particular, when $j = i$, the hindsight block exposes τ_i's own search spans to the teacher, raising a natural concern that the teacher may reduce to copying τ_i's queries through the prepended block. We address this concern in §3.3.1 with a leave-one-out variant that excludes τ_i from $h(\mathcal{G})$ while keeping its $G - 1$ sibling rollouts.

2.3 Token-Level Jensen-Shannon Objective

For each trajectory τ, the self-distillation loss averages a token-level divergence between the teacher and student distributions:

$$\mathcal{L}_{\text{SD}}(\tau) \;=\; \frac{1}{|\mathcal{Q}_\tau|} \sum_{p \in \mathcal{Q}_\tau} \text{JSD}\Big(P_p^{\text{tch}} \,\|\, P_p^{\text{stu}}\Big), \tag{6}$$

where the Jensen-Shannon divergence is the symmetric form

$$\text{JSD}(P \| Q) \;=\; \tfrac{1}{2}\text{KL}\Big(P \,\Big\|\, \tfrac{P+Q}{2}\Big) + \tfrac{1}{2}\text{KL}\Big(Q \,\Big\|\, \tfrac{P+Q}{2}\Big). \tag{7}$$

We choose JSD [1] over forward or reverse KL for two reasons. It is symmetric, thus avoiding biasing the student toward either mode-seeking or mode-covering behavior, and it is bounded by $\log 2$, which empirically improves training stability when teacher and student distributions occasionally diverge sharply on early-training rollouts. For computational efficiency, we additionally restrict both distributions to the union of their top-k tokens (renormalizing to sum to one over this support), with $k = 50$ by default, and we ablate this choice in §3.3.3. We restrict the sum to $\mathcal{Q}_\tau$ because these positions directly express each search decision, whereas the trajectory-level reward already propagates, albeit sparsely, to all action positions; we ablate broadening this scope to $\mathcal{A}_\tau$ in §3.3.2. For trajectories with $|\mathcal{Q}_\tau| = 0$, we set $\mathcal{L}_{\text{SD}}(\tau) = 0$ and train only with the GRPO term.

2.4 Training Procedure

Total Objective. The self-distillation loss is added to the standard GRPO objective without modifying its advantage estimator:

$$\mathcal{L}_{\text{total}} \;=\; \mathcal{L}_{\text{GRPO}} \;+\; \alpha_{\text{SD}}\, \mathcal{L}_{\text{SD}}. \tag{8}$$

These two terms are complementary. $\mathcal{L}_{\text{GRPO}}$ provides coarse trajectory-level reinforcement through the advantage $\hat{A}_i$, modulating token logits indirectly via reward. $\mathcal{L}_{\text{SD}}$ provides dense token-level supervision through direct distribution matching. Because $\mathcal{L}_{\text{SD}}$ never enters the advantage computation, existing infrastructure can adopt SD-Search by adding a single auxiliary forward pass and an extra term to the loss, with no changes to rollout sampling, reward shaping, or advantage normalization.

Warmup. At the start of training, the policy has not yet learned to follow the structured trajectory format, and its rollouts contain few well-formed search queries. Distilling from such trajectories injects noise rather than signal. We therefore apply a warmup of T_{warm} training steps (50 by default) during which α_{SD} is held at zero; afterwards it is set to its target value (10^{-3} by default).

Computational Cost. SD-Search adds a training-only teacher evaluation over the hindsight-augmented context and a JSD computation at search-query positions. Because the teacher target is stop-gradiented, the additional backward pass comes only from the student-side distillation loss, not from updating a separate teacher model. Inference cost is unchanged relative to the outcome-reward baseline. Empirically, a full 200-step run at the 3B scale takes about 11.9 hours on 8×H800, compared with 10.3 hours for the AutoRefine outcome-reward baseline under the same hardware, batch size, rollout configuration, and retriever, corresponding to a 15.5% end-to-end wall-clock overhead. This overhead remains within the standard RL loop: SD-Search requires no external teacher inference, unlike methods that synthesize training trajectories with a larger model, and no auxiliary API annotation phase, unlike methods that rely on GPT-4o-generated sub-question annotations. A per-stage cost breakdown is provided in Appendix E.

3 Experiments

3.1 Experimental Setup

Datasets. Following [11, 25], we train on a combined NQ and HotpotQA training set and evaluate on seven benchmarks: three single-hop (NQ [13], TriviaQA [12], PopQA [17]) and four multi-hop (HotpotQA [34], 2Wiki [10], MuSiQue [27], Bamboogle [19]). We report Exact Match (EM) accuracy after standard normalization. Dataset sizes and split statistics are reported in Appendix C.

Baselines. We compare SD-Search against three groups: retrieval-free baselines (Direct Generation, SFT, R1 [7]), a single-round retrieval baseline (Naive RAG [14]), and multi-round search-augmented reasoning methods. The last group further splits into inference-time prompting methods (Search-o1 [15], IRCoT [28]), outcome-reward RL methods (Search-R1 [11], ReSearch [4], AutoRefine [25], MR-Search [32]), and process-supervision methods that rely on external resources (StepSearch [30] uses GPT-4o sub-question annotations, and Thinker [33] builds its training data with a 72B teacher). All multi-round baselines share the same fixed retrieval corpora (December 2018 Wikipedia), fixed retriever (E5-base-v2 [29]), and retrieval depth of three passages per query. Baseline numbers are taken from their original papers when evaluation protocols match.

Implementation. SD-Search is implemented based on the veRL framework [24] with GRPO as the policy optimizer. Main results use Qwen2.5-3B and Qwen2.5-7B (both base and instruct) as backbones; all ablations are conducted with Qwen2.5-3B-Base. SD-Search introduces four additional hyperparameters, set to $\alpha_{\text{SD}}=10^{-3}$, $T_{\text{warm}}=50$, top-k=50, and $\rho=0$ throughout all experiments without per-dataset tuning. GRPO hyperparameters, training-time resources, and the full configuration are listed in Appendix D.

3.2 Main Results

Table 1 reports EM on all seven benchmarks at the 3B scale. SD-Search-Base reaches 0.428 average EM, improving over every outcome-reward baseline and statistically matching the leading process-supervision baseline Thinker-Instruct (seed-level variance is reported in Appendix B), despite using no external teacher model.

Table 1: EM on seven QA benchmarks with **Qwen2.5-3B**. Dashes denote settings not reported by the original paper. Bold denotes the best and underline the second best per column.

Method	Training Pipeline	External Teacher	Single-Hop			Multi-Hop				Avg.
			NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
w/o Retrieval										
Direct Generation	Zero-Shot	–	0.106	0.288	0.108	0.149	0.244	0.020	0.024	0.134
SFT	SFT	–	0.249	0.292	0.104	0.186	0.248	0.044	0.112	0.176
R1-Instruct	Pure RL	–	0.210	0.449	0.171	0.208	0.275	0.060	0.192	0.224
R1-Base	Pure RL	–	0.226	0.455	0.173	0.201	0.268	0.055	0.224	0.229
w/ Single-Hop Retrieval										
Naive RAG	Zero-Shot	–	0.348	0.544	0.387	0.255	0.226	0.047	0.080	0.270
w/ Multi-Hop Retrieval										
Search-o1	Prompting	–	0.238	0.472	0.262	0.221	0.218	0.054	0.320	0.255
IRCoT	Prompting	–	0.111	0.312	0.200	0.164	0.171	0.067	0.240	0.181
ReSearch-Base	Pure RL	–	0.427	0.597	0.430	0.305	0.272	0.074	0.128	0.319
ReSearch-Instruct	Pure RL	–	0.365	0.571	0.395	0.351	0.272	0.095	0.266	0.331
Search-R1-Base	Pure RL	–	0.421	0.583	0.413	0.297	0.274	0.066	0.128	0.312
Search-R1-Instruct	Pure RL	–	0.397	0.565	0.391	0.331	0.310	0.124	0.232	0.336
AutoRefine-Base	Pure RL	–	0.467	0.620	0.450	0.405	0.393	0.157	0.344	0.405
AutoRefine-Instruct	Pure RL	–	0.436	0.597	0.447	0.404	0.380	0.169	0.336	0.396
MR-Search-Base	Pure RL	–	**0.477**	**0.635**	0.460	0.419	0.401	0.165	0.344	0.414
StepSearch-Base	Pure RL	GPT-4o	–	–	–	0.329	0.339	0.181	0.328	–
StepSearch-Instruct	Pure RL	GPT-4o	–	–	–	0.345	0.320	0.174	0.344	–
Thinker-Instruct	Synthesize → SFT	Qwen2.5-72B	0.439	0.598	**0.469**	0.400	**0.469**	**0.214**	**0.424**	**0.430**
SD-Search-Base	Pure RL	–	<u>0.470</u>	<u>0.624</u>	<u>0.467</u>	**0.425**	<u>0.420</u>	<u>0.188</u>	0.402	<u>0.428</u>
SD-Search-Instruct	Pure RL	–	0.469	<u>0.624</u>	0.465	<u>0.424</u>	0.419	0.184	<u>0.404</u>	0.427

Against **pure outcome-reward methods**, SD-Search-Base surpasses AutoRefine-Base by 2.3 points on average and MR-Search-Base by 1.4 points. The gains are larger on benchmarks requiring multiple retrieval hops (Bamboogle $+5.8$, MuSiQue $+3.1$, 2Wiki $+2.7$ over AutoRefine-Base). The self-distillation loss is applied at every search-query token, so trajectories with more search steps receive proportionally denser supervision. Multi-hop questions also induce greater within-trajectory variance in query quality (some hops route correctly, others retrieve irrelevant passages), and the token-level JSD signal explicitly distinguishes these positions, whereas a trajectory-level advantage averages over them. The two effects compound, accounting for the larger gains on multi-hop benchmarks. SD-Search-Instruct tracks the Base variant closely at 0.427, indicating the benefit is not tied to any particular instruction-tuning prior. Against **process-supervision methods**, SD-Search-Base is statistically indistinguishable from Thinker on average (0.428 versus 0.430), while using only the question-answer pairs required by standard GRPO, without the 72B teacher or supervised distillation stage that Thinker depends on. It also surpasses StepSearch on every multi-hop benchmark, with gains of $+9.6$ points on HotpotQA and $+8.1$ on 2Wiki over StepSearch-Base, while not requiring GPT-4o-generated sub-question annotations that StepSearch relies on. These results indicate that: *the step-level signal that process-supervision methods obtain from external sources can be recovered from the policy itself through hindsight conditioning, without compromising performance.* Training dynamics further support this interpretation, with SD-Search's gain driven by query quality rather than search volume (Appendix F). An illustrative rollout comparison on a 2Wiki question is provided in Appendix G.

Table 2 reports the same comparison at the 7B scale. SD-Search-Instruct reaches 0.476 average EM, improving over MR-Search-Base by 1.6 points and AutoRefine-Base by 2.1 points. It also surpasses Thinker-Instruct (0.452) by 2.4 points while using no external teacher model, confirming at this scale the same pattern observed at 3B. The per-benchmark gains are concentrated on multi-hop settings (HotpotQA $+2.0$, MuSiQue $+3.5$ over AutoRefine-Base), and SD-Search-Base reaches 0.471, within 0.5 point of the Instruct variant, indicating that the hindsight signal transfers to both pretraining-only and instruction-tuned starting points. Notably, while AutoRefine-Base improves by $+5.0$ points from 3B to 7B ($0.405 \rightarrow 0.455$), Thinker-Instruct only adds $+2.2$ points ($0.430 \rightarrow 0.452$), lagging behind its own non-distilled outcome-reward baseline. By contrast, SD-Search scales with the student under both conditioning regimes: comparing Base-to-Base, SD-Search-Base gains $+4.3$ points ($0.428 \rightarrow 0.471$), tracking AutoRefine-Base's $+5.0$ within 0.7 point; comparing Instruct-to-Instruct, SD-Search-Instruct gains $+4.9$ points ($0.427 \rightarrow 0.476$), more than double Thinker-Instruct's $+2.2$. This is consistent with the view that hindsight self-distillation scales with the student's own in-context reading capacity rather than against a fixed external reference. An extended scaling analysis at 1.5B, 3B, 7B, and 14B (Appendix H) confirms this pattern: Thinker's gain over AutoRefine shrinks monotonically with scale and turns negative from 7B onward, while SD-Search's gain remains positive at every scale, narrowing at 14B but never crossing zero.

Table 2: EM on seven QA benchmarks with **Qwen2.5-7B**. Dashes denote settings not reported by the original paper. Bold denotes the best and underline the second best per column.

Method	**Training Pipeline**	**External Teacher**	**Single-Hop**			**Multi-Hop**				**Avg.**
			NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
Search-R1-Base	Pure RL	–	0.469	0.627	0.449	0.410	0.272	0.173	0.456	0.408
Search-R1-Instruct	Pure RL	–	0.393	0.610	0.397	0.370	0.414	0.146	0.368	0.385
AutoRefine-Base	Pure RL	–	0.484	0.659	0.487	0.451	0.405	0.187	0.512	0.455
MR-Search-Base	Pure RL	–	**0.502**	0.666	0.472	0.468	0.436	0.221	0.452	0.460
StepSearch-Base	Pure RL	GPT-4o	–	–	–	0.380	0.385	0.216	0.467	–
StepSearch-Instruct	Pure RL	GPT-4o	–	–	–	0.386	0.366	**0.226**	0.400	–
Thinker-Instruct	Synthesize $\rightarrow$ SFT	Qwen2.5-72B	0.450	0.642	0.484	0.421	**0.469**	0.221	0.480	0.452
SD-Search-Base	Pure RL	–	0.500	0.665	0.489	0.468	0.438	0.204	0.532	0.471
SD-Search-Instruct	Pure RL	–	0.495	**0.668**	**0.494**	**0.471**	0.441	0.222	**0.544**	**0.476**

3.3 Ablation Studies

We run three families of ablations on Qwen2.5-3B-Base. The first (§3.3.1) isolates the construction of the hindsight block, asking what information the teacher should and should not see. The second (§3.3.2) isolates the student-teacher alignment objective itself. The third (§3.3.3) reports sensitivity to the four SD-Search hyperparameters around their defaults.

3.3.1 Hindsight Block Construction

The hindsight block $h(\mathcal{G})$ makes three design choices: attaching an outcome label to each rollout, masking non-search tokens, and aggregating over the entire rollout group. We modify each choice in turn while keeping the rest of the pipeline fixed, and additionally probe whether the outcome label is used as a CORRECT/INCORRECT contrast or as label conditioning (by shuffling labels or filtering to correct rollouts only), and whether the block can be replaced by the current-step documents d_t alone. Figure 3 reports EM under each configuration.

The outcome-label family (three orange shades in Figure 3) probes the labels themselves. Removing the outcome label costs 1.4 points, since the teacher cannot tell queries from successful rollouts apart from those from failed ones. Shuffling each label uniformly at random costs 2.3 points, *worse* than removing them: random labels are actively misleading, isolating the CORRECT/INCORRECT contrast (rather than label conditioning) as the source of the gain. Keeping only successful skeletons (correct rollouts only) costs 0.7 points: one-sided imitation recovers part but not all of the contrast benefit.

Removing the multi-rollout group costs 1.0 points; the leave-one-out variant, which removes τ_i from $h(\mathcal{G})$ while keeping its $G-1$ siblings, is its complement and costs only 0.5 points. The cross-rollout contrast contributes more than τ_i's own search spans, and the teacher cannot be trivially reduced to a copy of τ_i through the prepended block.

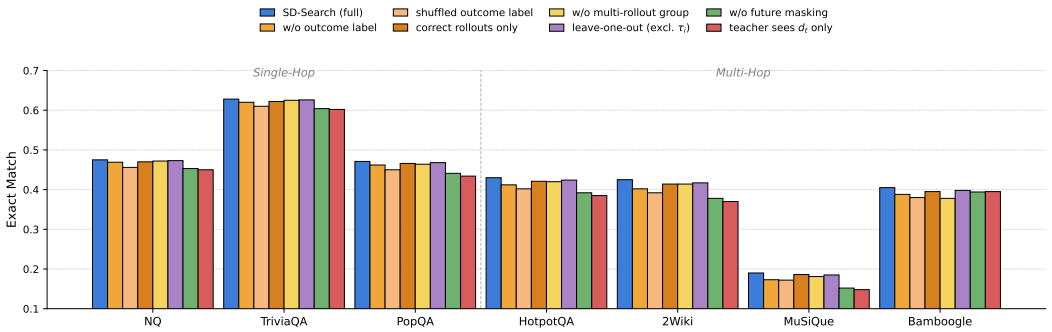


Figure 3: Hindsight block construction ablation on Qwen2.5-3B-Base. Eight configurations cluster into three colour-coded families. The *outcome-label* family (three orange shades) dissects the role of outcome labels. The *group-structure* family (yellow, purple) probes which part of the rollout group carries the signal. The *future-content* family (green, red) probes whether downstream tokens leak the answer. Drops are concentrated on multi-hop benchmarks for every family.

Removing future masking costs 3.0 points on average, with the sharpest losses on multi-hop benchmarks (HotpotQA -3.8, 2Wiki -4.7). Once the teacher conditions on downstream think tokens and retrieved documents, the gold answer becomes predictable from the prefix and the teacher's distribution concentrates on retrieval-skipping continuations. Replacing the entire block with the current-step documents d_t alone produces the largest drop of 3.4 points: the teacher either skips the search or echoes entities from d_t rather than assessing query quality. Local retrieval traces therefore do not recover the role of the full hindsight block as a query-quality supervisor.

3.3.2 Student-Teacher Alignment Objective

The self-distillation loss in Eq. 6 makes two choices: Jensen-Shannon divergence as the distributional distance, and a summation scope restricted to the search-query positions $\mathcal{Q}_\tau$. We probe each choice by replacing it with alternatives while keeping the rest fixed. Results are reported in Table 3.

Table 3: Alignment objective ablation on Qwen2.5-3B-Base. The top block varies the divergence function while keeping the scope fixed to $\mathcal{Q}_\tau$; the bottom row broadens the scope from $\mathcal{Q}_\tau$ to $\mathcal{A}_\tau$ while keeping the divergence fixed to JSD.

Configuration	**Single-Hop**			**Multi-Hop**				**Avg.**
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
SD-Search-Base (full)	0.470	0.624	0.467	0.425	0.420	0.188	0.402	**0.428**
forward KL instead of JSD	0.465	0.619	0.459	0.413	0.405	0.178	0.387	0.418
reverse KL instead of JSD	0.447	0.619	0.464	0.408	0.430	0.153	0.376	0.414
MSE instead of JSD	0.460	0.614	0.454	0.400	0.393	0.168	0.362	0.407
scope $\mathcal{A}_\tau$ instead of $\mathcal{Q}_\tau$	0.466	0.620	0.461	0.417	0.410	0.181	0.392	0.421

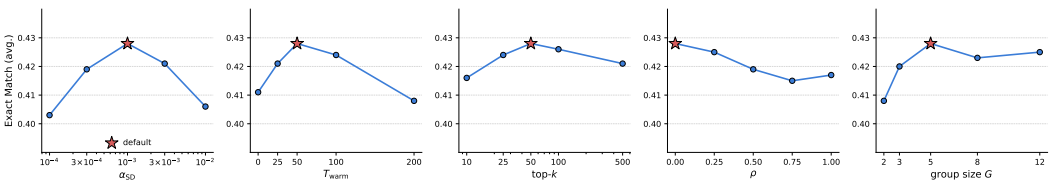


Figure 4: Sensitivity of hyperparameters on Qwen2.5-3B-Base. Each panel varies one hyperparameter while the others are fixed at their defaults (marked with a red star). The first four panels sweep the self-distillation hyperparameters; the fifth sweeps the GRPO group size G.

All three alternatives underperform JSD, by 1.0, 1.4, and 2.1 points on average for forward KL [9], reverse KL [6], and MSE respectively. The two asymmetric KLs underperform in opposite ways.

Forward KL is mode-covering: it requires the student to place non-zero mass on *every* token supported by the teacher, preventing the student from committing sharply to any single keyword even when the teacher itself is highly concentrated. The resulting entropy in the student's predictions slightly blunts the signal. **Reverse KL** is mode-seeking, collapsing the student onto the teacher's single highest-probability token and foreclosing exploration, a cost that shows up most on multi-hop benchmarks where multiple near-equivalent query reformulations are often viable. **MSE** (applied to the raw logits) performs worst because it treats every logit coordinate as equally important, ignoring the probability-simplex structure that the KL family respects; small logit shifts on irrelevant tokens are penalized as heavily as shifts on the high-probability tokens that actually carry the query information. These results support the symmetry and boundedness properties we attributed to JSD in §2.3. **Broadening the scope** from $\mathcal{Q}_\tau$ to all policy-generated positions $\mathcal{A}_\tau$ costs 0.7 points, a smaller effect than any of the divergence swaps. We believe this reflects a subtler failure mode: at think and answer positions, the teacher's distribution is partially shaped by the outcome labels in $h(\mathcal{G})$, which pushes it toward sharply concentrated continuations that short-circuit the reasoning chain. Aligning the student to these overly confident targets suppresses the exploratory think patterns that the trajectory-level advantage was relying on.

3.3.3 Hyperparameter Sensitivity

We sweep each of the four SD-Search hyperparameters together with the GRPO group size G in turn while fixing the others at their defaults. Figure 4 reports EM across the seven benchmarks. All five defaults, including the group size $G{=}5$, sit at the peak of their respective curves, and the overall amplitude across the sweep window is modest, never exceeding 2.5 points.

α_{SD} shows a mild inverted-U shape around its default: too small a coefficient underweights the step-level signal, while too large a coefficient overwhelms the GRPO exploration that the advantage term was supposed to induce. T_{warm} shows a similar shape but for a different reason: starting distillation too early feeds noise from poorly formed trajectories into the student, while starting it too late forfeits the training window in which query formulation is most plastic. The top-k sweep is nearly flat in the range $k \in [25, 100]$, with modest degradation at both ends. This insensitivity is consistent with our observation in §2.3 that most of the JSD mass at each position is concentrated on a small number of high-probability tokens. ρ peaks at the default $\rho{=}0.0$ and drops as the threshold tightens, reaching its minimum near $\rho{=}0.75$, where too many rollouts get relabeled to INCORRECT and the hindsight block loses its outcome contrast. The small recovery at $\rho = 1.0$ reflects that a strict perfect-match criterion successfully filters out ambiguous outcomes, leaving a sharper but sparser contrast between CORRECT and INCORRECT trajectories.

4 Limitations

SD-Search has two main limitations. First, the hindsight contrast relies on outcome labels $y(\tau_j)$ derived from gold answers, as it is inherited from GRPO itself, which restricts the method to tasks with reliably scorable references. Extending SD-Search to open-ended generation requires a substitute for the binary outcome label, for example a preference score from a learned model or a majority-vote proxy across the rollout group, which we leave to future work. Second, although the multi-rollout group is designed to keep the hindsight signal informative when the current rollout τ_i fails, the outcome contrast in the hindsight block degenerates whenever the group becomes label-homogeneous in either direction. The all-failure scenario, in which every rollout in the group ends INCORRECT, is the same scenario where GRPO's trajectory-level advantages also collapse; the symmetric all-success regime, in which every rollout ends CORRECT, removes the contrast as well, and we observe a corresponding narrowing of SD-Search's gain over AutoRefine at 14B (Appendix H), where higher base success rates make group-uniform CORRECT labels more frequent. Designing a hindsight construction that remains informative under both group-uniform scenarios is an open problem.

5 Conclusion

We introduce SD-Search, a reinforcement learning framework that supplies step-level supervision for search-augmented reasoning through hindsight self-distillation. The same policy serves as both student and teacher, with the teacher conditioned on a hindsight block that aggregates the rollout group and its outcomes. Aligning the student to the teacher on search-query positions yields a

dense gradient signal that complements GRPO's trajectory-level reward, without any external teacher model or auxiliary annotations. On seven QA benchmarks, SD-Search outperforms outcome-reward baselines and matches process-supervision methods that rely on much larger external systems, at a 15.5% end-to-end training-time overhead which is entirely contained within the standard RL loop, without external teacher inference, annotation phase, or additional inference cost. More broadly, the construction of a teacher by conditioning the same policy on hindsight observations points toward a more general recipe for extracting step-level signal from trajectory-level reinforcement learning, beyond the search-augmented reasoning setting studied here.

Acknowledgments and Disclosure of Funding

We thank Kuaishou Technology for supporting this research work.

Funding. This work was supported by Kuaishou Technology. The authors received no other third-party funding or support for this work during the 36 months preceding submission.

Competing interests. The authors' affiliation with Kuaishou Technology is disclosed in the author information. The authors declare no additional financial relationships or competing interests relevant to this work during the 36 months preceding submission.

References

[1] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, pages 21246–21263, 2024.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2018.

[3] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*, 2022.

[4] 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. *arXiv preprint arXiv:2503.19470*, 2025.

[5] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[6] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations*, pages 32694–32717, 2024.

[7] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, and Wu. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

[8] Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, and Remi Munos. Hindsight credit assignment. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[10] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, 2020.

[11] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025.

[12] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, 2017.

[13] 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.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2021.

[15] 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. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438, Suzhou, China, 2025.

[16] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2016.

[17] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada, 2023.

[18] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2024.

[19] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore, 2023.

[20] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, pages 68539–68551, 2023.

[21] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[22] 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. *arXiv preprint arXiv:2402.03300*, 2024.

[23] Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

[24] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pages 1279–1297, 2025.

[25] Yaorui Shi, Sihang Li, Chang Wu, Zhiyuan Liu, Junfeng Fang, Hengxing Cai, An Zhang, and Xiang Wang. Search and refine during think: Facilitating knowledge refinement for improved retrieval-augmented reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[26] Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

[27] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

[28] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada, 2023.

[29] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2024.

[30] Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *arXiv preprint arXiv:2505.15107*, 2025.

[31] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, pages 24824–24837, 2022.

[32] Teng Xiao, Yige Yuan, Hamish Ivison, Huaisheng Zhu, Faeze Brahman, Nathan Lambert, Pradeep Dasigi, Noah A. Smith, and Hannaneh Hajishirzi. Meta-reinforcement learning with self-reflection for agentic search. *arXiv preprint arXiv:2603.11327*, 2026.

[33] Jun Xu, Xinkai Du, Yu Ao, Peilong Zhao, Yang Li, Ling Zhong, Lin Yuan, Zhongpu Bo, Xiaorui Wang, Mengshu Sun, Zhengke Gui, Dalong Zhang, Zhaoyang Wang, Qiwei Wang, Yangyang Hou, Zhiying Yin, Haofen Wang, Huajun Chen, Lei Liang, and Jun Zhou. Thinker: Training llms in hierarchical thinking for deep search via multi-turn interaction. *arXiv preprint arXiv:2511.07943*, 2025.

[34] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, 2018.

[35] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[36] Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

A Related Work

Search-augmented reasoning with reinforcement learning. A line of recent work trains LLMs to invoke retrieval tools during reasoning, optimizing the policy with outcome-level reinforcement learning. Search-R1 [11] introduces the search-during-think paradigm and uses GRPO to train the agent on final-answer correctness. ReSearch [4] and AutoRefine [25] extend this formulation with stronger reward shaping and structured trajectories, and MR-Search [32] further introduces cross-episode reflection. All of these methods share a common property: supervision operates only at the trajectory level, so every token within a rollout, regardless of whether a query is well or poorly formed, receives the same gradient signal. SD-Search inherits the trajectory format and GRPO outer loop from this line of work but adds a dense step-level signal on top, without modifying the advantage estimator.

Process supervision for LLM reasoning agents. Several recent methods target the same limitation SD-Search addresses, namely the absence of step-level credit assignment under trajectory-level rewards. Thinker [33] constructs sub-question decompositions using a 72B teacher and supervises the student through distillation on these trajectories. StepSearch [30] derives step-wise rewards from GPT-4o-generated sub-question annotations. GiGPO [5] takes a different route, exploiting repeated environment states to build step-level advantage groups within a trajectory; while originally developed for embodied and web-interaction agents, it exemplifies a structural approach to step-level signal. These methods deliver clear improvements, but each obtains its step-level signal from outside the policy, either by importing external supervision (Thinker, StepSearch) or by assuming structural regularities in the trajectory (GiGPO). SD-Search differs in deriving the signal entirely from the policy itself, through conditioning on hindsight rather than through any external or structural proxy.

Self-distillation and hindsight learning. Recent self-distillation work shows that a single language model can serve as its own teacher when conditioned on privileged information the student does not see. The underlying principle of learning from privileged information dates back to Lopez-Paz et al. [16]; recent applications include mathematical reasoning [36] and continual learning [23]. In all these settings the privileged information is drawn from fixed external sources such as ground-truth traces or expert demonstrations. SD-Search applies the same mechanism in a reinforcement learning setting, where the privileged information is instead produced by the policy itself during rollout, namely the hindsight block over the rollout group and its outcomes. The term "hindsight" here is borrowed from Hindsight Experience Replay [2] and the more recent hindsight credit assignment line [8], both of which operate at the reward level whereas SD-Search operates at the token distribution level.

B Robustness Across Random Seeds

To verify that our improvements are robust to random initialization, we re-train SD-Search-Base, AutoRefine-Base, and Thinker-Instruct with *five* random seeds on Qwen2.5-3B under otherwise identical settings. Table 4 reports the mean and standard deviation across seeds on each benchmark. The per-benchmark variability is uneven and is dominated by two factors: the size of the evaluation split and the intrinsic stochasticity of RL training. Standard deviations are smallest on the large single-hop test sets (NQ, TriviaQA, PopQA, all containing $>$3k examples; std $\approx$ 0.006–0.010), larger on the multi-hop dev sets (HotpotQA, 2Wiki, MuSiQue; std $\approx$ 0.009–0.015), and largest on Bamboogle (std $\approx$ 0.022–0.028), whose 125-example test split inflates the seed variance relative to the others. The two RL methods (AutoRefine-Base and SD-Search-Base) exhibit comparable seed-level variance, while Thinker-Instruct, which is trained via SFT on synthetic trajectories, is slightly more stable, consistent with SFT generally producing lower seed-to-seed variance than RL.

Across seeds, SD-Search-Base's average EM (0.428 ± 0.008) lies above AutoRefine-Base's (0.404 ± 0.008) on every seed we ran, while overlapping with Thinker-Instruct's (0.429 ± 0.007). We describe these gaps as consistent rather than as the result of formal significance testing: five seeds is a modest sample under RL training variance, and the per-benchmark standard deviations on small test sets such as Bamboogle remain large enough that single-benchmark conclusions should be read with caution. However, the pattern across the average column matches the main-text claim that SD-Search reliably beats outcome-reward baselines without an external teacher, and performs comparably to the external-teacher baseline.

Table 4: Exact Match (mean$_{\text{std}}$) across five random seeds on Qwen2.5-3B.

Method	**Single-Hop**			**Multi-Hop**				**Avg**$^{\cdot}$
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	
AutoRefine-Base	$0.464_{\pm.009}$	$0.621_{\pm.006}$	$0.447_{\pm.008}$	$0.408_{\pm.013}$	$0.390_{\pm.014}$	$0.158_{\pm.012}$	$0.339_{\pm.028}$	$0.404_{\pm.008}$
Thinker-Instruct	$0.441_{\pm.008}$	$0.594_{\pm.006}$	$0.466_{\pm.007}$	$0.403_{\pm.011}$	$0.466_{\pm.010}$	$0.211_{\pm.009}$	$0.421_{\pm.022}$	$0.429_{\pm.007}$
SD-Search-Base	$0.470_{\pm.010}$	$0.624_{\pm.007}$	$0.467_{\pm.009}$	$0.425_{\pm.012}$	$0.420_{\pm.015}$	$0.188_{\pm.013}$	$0.402_{\pm.026}$	$0.428_{\pm.008}$

C Dataset Statistics

Table 5 summarizes the seven question answering benchmarks used in our experiments. All datasets are sourced from the FlashRAG datasets collection. We construct the training set by combining the train splits of NQ and HotpotQA, following Jin et al. [11]. For evaluation we use the test split where available (NQ, TriviaQA, PopQA, Bamboogle) and the dev split otherwise (HotpotQA, 2Wiki, MuSiQue).

Table 5: Statistics of the seven benchmarks used in this paper.

Dataset	**Split sizes**			**Eval split**
	Train	Dev	Test	
NQ	79,168	8,757	3,610	test
TriviaQA	78,785	8,837	11,313	test
PopQA	–	–	14,267	test
HotpotQA	90,447	7,405	–	dev
2Wiki	15,000	12,576	–	dev
MuSiQue	19,938	2,417	–	dev
Bamboogle	–	–	125	test

D Hyperparameters

Table 6 lists the complete set of hyperparameters used in our experiments. The SD-Search-specific hyperparameters ($\alpha_{\text{SD}}, T_{\text{warm}}$, top-$k$, ρ) were selected on a validation subset of NQ and HotpotQA; their sensitivity is analyzed in §3.3.3. All other values follow the default configuration of the veRL framework [24].

Table 6: Complete hyperparameter configuration for SD-Search training.

Hyperparameter	**Value**
Training configuration	
Group size G	5
Training batch size	256
Actor learning rate	1×10^{-6}
Total training steps	200
KL coefficient β	0.001
Clip ratio ϵ	0.2
Max search calls per rollout	3
Rollout temperature	1.0
Retrieval configuration	
Retriever	E5-base-v2
Corpus	Wikipedia (Dec. 2018)
Top-k documents per query	3
SD-Search specific configuration	
Distillation coefficient α_{SD}	1×10^{-3}
Warmup steps T_{warm}	50
Distribution truncation top-k	50
Outcome threshold ρ	0
Hindsight max length	1024 tokens
Future masking	enabled
Alignment scope	$\mathcal{Q}_\tau$

E Training Cost Breakdown

We measure SD-Search's training cost both end-to-end (full 200-step run wall-clock) and at per-step granularity (a separate 100-step instrumented run). Both numbers are reported in Table 7; together they let us isolate which stages of the training loop the SD-Search additions actually touch.

End-to-end. On 8×H800, the full 200-step training run takes about 11.9h with SD-Search enabled (Qwen2.5-3B-Base) versus about 10.3h for the AutoRefine baseline under identical hardware, batch size, rollout configuration, and retriever, a $+15.5\%$ end-to-end wall-clock overhead.

Per-step breakdown. To attribute this overhead to specific training stages, we instrument both runs at the level of `update_policy`'s internal sections (PPO forward, PPO backward, SD-Search compute, SD-Search backward, optimizer + communication) using CUDA-synced wall-clock timers, and average over a 100-step run skipping the first 50 steps as warmup. Table 7 reports the result. Roughly 94% of the per-step overhead is contributed by the SD-Search compute stage (teacher forward + JSD computation, $+22.0\%$ of baseline step time) and the SD-Search backward stage ($+17.5\%$). All other stages, including rollout generation, reward computation, PPO forward/backward, optimizer step, and communication, are unchanged within measurement noise ($|\Delta| \leq 2.1\%$ of baseline; the rollout stage is in fact slightly faster for SD-Search at -2.1%, attributable to run-to-run variance since the rollout pipeline is identical between the two methods), confirming that SD-Search does not affect the rollout, retrieval, advantage, or core PPO pipeline.

Table 7: Per-step wall-clock breakdown on Qwen2.5-3B-Base, 8×H800, batch=256, G=5. Means over 50 steps after warmup skip. The two SD-Search rows account for nearly all of the overhead; all other stages are unchanged within measurement noise.

Stage	**AutoRefine** (s)	**SD-Search** (s)	Δ (s)	Δ (% of baseline total)
Rollout generation	66.16	64.23	-1.93	-2.1%
Reward + advantage	2.69	3.31	$+0.61$	$+0.7\%$
Reference logprob	3.67	4.13	$+0.46$	$+0.5\%$
PPO forward	5.70	6.01	$+0.31$	$+0.3\%$
PPO backward	9.25	9.97	$+0.72$	$+0.8\%$
SD-Search compute	—	20.31	$+20.31$	$+22.0\%$
SD-Search backward	—	16.13	$+16.13$	$+17.5\%$
Optimizer + comm	0.13	0.13	~ 0	$\sim 0\%$
Other (data loading, sync, logging)	4.80	6.61	$+1.81$	$+2.0\%$
Total per step	**92.40**	**130.83**	$\mathbf{+38.43}$	$\mathbf{+41.6\%}$

Note on the per-step vs. end-to-end gap. The per-step overhead ($+41.6\%$) measures only `update_policy`'s post-warmup contribution; the end-to-end overhead ($+15.5\%$) is smaller because two amortization effects compound. (i) The $T_{\text{warm}} = 50$ warmup steps carry no SD-Search overhead, so averaged over the full 200-step run the training-only overhead is $(24{,}244.5 - 18{,}480)/18{,}480 = 31.2\%$, already below the per-step figure. (ii) Periodic validation, checkpoint saving, and setup, which together account for $\approx$ 5.2h of AR's 10.3h end-to-end wall-clock and are identical between the two methods, inflate both totals equally and dilute the relative overhead from 31.2% to $(11.9 - 10.3)/10.3 = 15.5\%$. Both effects shrink the gap as expected.

Comparison with process-supervision baselines. The 15.5% end-to-end overhead is contained entirely within the standard RL training loop: no external model inference, no annotation pipeline, no additional supervised stage. By contrast, the process-supervision baselines we compare against in Table 1 carry substantial costs that do not appear in their reported training time:

- **Thinker** [33] runs Qwen2.5-72B inference over the training set to generate sub-question decompositions, then performs a supervised fine-tuning stage on the resulting trajectories before RL. The 72B forward pass alone is $\sim 24\times$ the per-token cost of our 3B teacher pass on a parameter-count basis, applied to every training trajectory.
- **StepSearch** [30] relies on GPT-4o-generated sub-question annotations on the full $\sim$ 100k-example NQ+HotpotQA training set, incurring both wall-clock and monetary costs that scale directly with dataset size.

SD-Search trades these multi-stage external pipelines for a single in-loop overhead, which we view as the more favorable point on the cost-quality frontier for practitioners without access to a 72B teacher or commercial API budget.

What end-to-end wall-clock includes. The end-to-end figure (10.3h vs. 11.9h) covers the entire training run, including periodic validation evaluation on the seven benchmarks, checkpoint saving, and initial setup, in addition to the per-step training time measured in Table 7. The validation cost is identical for both methods (no SD-Search forward at evaluation time), so it inflates the absolute wall-clock equally and yields an end-to-end overhead ($+15.5\%$) that is correctly smaller than the per-step training overhead ($+41.6\%$).

F Training Dynamics

Figure 5 reports four training-time metrics on Qwen2.5-3B-Base, averaged across the seven benchmarks. The first metric, the teacher-student entropy gap on $\mathcal{Q}_\tau$ (the per-position difference $H(P_p^{\text{stu}}) - H(P_p^{\text{tch}})$ averaged over search-query positions), is meaningful only for SD-Search, since AutoRefine has no teacher view. The remaining three, including validation Exact Match, search quality (the fraction of retrieved documents containing the gold answer), and search frequency (average number of search calls per rollout), are computed for both SD-Search and AutoRefine. We restrict the cross-method comparison to SD-Search and AutoRefine because both share the same pure-RL training schedule from the same base model; Thinker is trained with SFT on synthetic trajectories rather than RL from scratch, so its per-step training trajectory is not alignable to ours.

Before the warmup cutoff $T_{\text{warm}} = 50$ SD-Search and AutoRefine are indistinguishable on EM, search quality, and search frequency, since SD-Search's distillation loss is not yet active. After step 50 the curves diverge on EM and search quality but remain nearly overlapping on search frequency: SD-Search converges to roughly the same number of search calls per rollout as AutoRefine, while achieving a higher fraction of successful retrievals and a higher final EM. The gain therefore comes from higher-quality queries at a comparable search rate, not from issuing more searches per question, ruling out the possibility that SD-Search's EM advantage is a byproduct of raw search volume or incidental training stabilisation.

The entropy-gap panel is an interpretable proxy for the alignment that $\mathcal{L}_{\text{SD}}$ in Eq. 6 drives: as the JSD objective pulls the two distributions together their entropies must also converge. The gap is positive by construction, since the teacher's hindsight conditioning yields systematically sharper predictions than the student's prefix-only context. It hovers around 0.25 pre-warmup, drops sharply to ~ 0.10 within about 30 post-warmup steps, and declines noisily toward ~ 0.065 by step 200, with a mid-training oscillation reflecting normal RL stochasticity. The gap does not reach zero: the teacher's hindsight block carries information the student cannot recover from its prefix, so a positive plateau is the expected steady state.

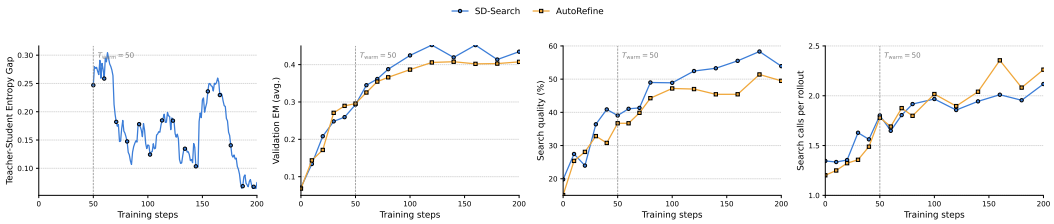


Figure 5: Training dynamics of SD-Search versus AutoRefine on Qwen2.5-3B-Base, averaged over the seven benchmarks. From left to right: the teacher-student entropy gap averaged over the search-query positions $\mathcal{Q}_\tau$ (only meaningful for SD-Search, since AutoRefine has no teacher view), validation Exact Match, search quality (the fraction of retrieved documents containing the gold answer), and search frequency (average number of search calls per rollout). The dashed line marks the warmup cutoff $T_{\text{warm}} = 50$, after which SD-Search's self-distillation loss is activated.

G Case Study

Table 8 shows how SD-Search, AutoRefine, and Thinker roll out on the same multi-hop question from 2Wiki. Incorrect answers are marked in red and correct answers in green; the underlined tokens in each search query highlight the entity or predicate the query is targeting. The example illustrates how the three methods differ in query focus during the first hop, which in this case is the hop where the chain either succeeds or fails.

Table 8: Rollout comparison on a 2Wiki question requiring a bridge entity and a date lookup.

Question: What is the date of death of Alexander of Masovia's father?	
Gold Answer: 21 January 1426 .	
AutoRefine	
`<think>`	I need to find the date of death of Alexander of Masovia's father.
`<search>`	Alexander of Masovia father date of death
`<documents>`	[Doc 1: Alexander of Masovia] Alexander of Masovia (1400–2 June 1444) . . . was the second son of Siemowit IV, Duke of Masovia . . .
`<answer>`	1400
Thinker	
`<think>`	To answer this, I need to know who Alexander of Masovia's father is, then his death date.
`<search>`	Step1: Who is the father of Alexander of Masovia? Action1: Retrieval(s=s1:person['Alexander of Masovia'], p=p1:father, o=o1:person)
`<documents>`	[Doc 1: Alexander of Masovia] . . . was the second son of Siemowit IV, Duke of Masovia . . .
`<search>`	Step2: Biography of #1 Action2: Retrieval(s=o1, p=p2:biography, o=o2:biography)
`<documents>`	[Doc 1: Siemowit IV] Siemowit IV (c. 1352–1426) was Duke of Masovia from 1381 until his death . . .
`<answer>`	1426
SD-Search	
`<think>`	First identify the father, then look up his death date.
`<search>`	Alexander of Masovia father
`<documents>`	[Doc 1: Alexander of Masovia] . . . was the second son of Siemowit IV, Duke of Masovia . . .
`<search>`	Siemowit IV Duke of Masovia date of death
`<documents>`	[Doc 1: Siemowit IV] . . . (c. 1352 – 21 January 1426) . . .
`<answer>`	21 January 1426

The three methods diverge in how they decompose the question into search queries. AutoRefine collapses the two-hop question into a single query containing both the bridge entity placeholder and the target predicate; the retriever returns a document about Alexander himself, and the model extracts Alexander's own dates rather than his father's. Thinker correctly identifies the need for two hops, but its second query is a generic biography request that returns only a coarse year, missing the precise date. SD-Search issues two focused queries, each targeting a single fact (first the father, then the father's date of death), and recovers the full answer. The pattern is consistent with the training-dynamics finding that SD-Search's gain comes from query quality rather than query volume: both Thinker and SD-Search issue two searches, but SD-Search's queries are more surgically aimed at the fact being retrieved.

H Scaling Across Model Sizes

We train SD-Search, Thinker, and AutoRefine on Qwen2.5 at four scales (1.5B, 3B, 7B, 14B) under identical settings and report average EM across the seven benchmarks at step 200 in Figure 6. The 3B and 7B points reproduce Tables 1 and 2; the 1.5B and 14B points are trained with the same recipe.

The relative position of the three methods changes systematically with scale. At 1.5B, both supervision-augmented methods clearly improve over AutoRefine ($+3.7$ for Thinker and $+2.5$ for SD-Search), with Thinker slightly ahead of SD-Search (-1.2); a 1.5B student appears to benefit more from a stronger external teacher than from its own hindsight view, since its in-context reading capacity is still limited. At 3B the two methods become essentially indistinguishable (SD-Search 0.428 vs. Thinker 0.430), with both improving over AutoRefine by $\approx +2.4$. From 7B onward the ordering inverts: Thinker's gain over AutoRefine turns mildly negative (-0.3 at 7B and -0.5 at 14B), consistent with the saturation argument in §3.2: as the student grows toward the 72B teacher, distillation from a fixed external reference delivers diminishing returns and at some point becomes a soft ceiling on the student. In contrast, SD-Search stays above AutoRefine at every scale. Comparing the best-performing variant of each method against AutoRefine-Base, the gain is $+2.5/+2.3/+2.1/+0.9$ at 1.5B/3B/7B/14B; the apples-to-apples Base-to-Base comparison is $+2.5/+2.3/+1.6/+0.6$ across the same scales (the 7B-Base gap of $+1.6$ widens to $+2.1$ once the SD-Search-Instruct variant is included, see Table 2). Under both reportings the margin narrows at 14B but never crosses zero, because the teacher view is produced by the student itself and grows with the student's own in-context reading capacity rather than against a fixed reference.

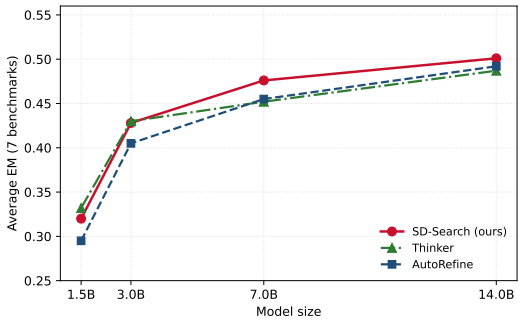


Figure 6: Scaling on Qwen2.5 at 1.5B, 3B, 7B, 14B (average EM across the seven benchmarks at training step 200). Thinker leads at 1.5B and ties SD-Search at 3B, but its gain over AutoRefine drops from $+3.7$ to -0.5 as scale grows from 1.5B to 14B. SD-Search's gain over AutoRefine stays positive at all four scales, narrowing only at 14B where higher base success rates reduce the within-group outcome contrast (see §4).

I Teacher Input Schematic

§ 2.2 formalizes the teacher view as $c^{\text{tch}}(\tau_i; \mathcal{G}) = [h(\mathcal{G}); \tau_i]$, which makes the conditioning structure clear but abstracts away what the teacher's input literally looks like as a token sequence. We provide a concrete instance in Figure 7: a single teacher input rendered as the model receives it, on the same 2Wiki question used in Appendix G under our default $G{=}5$ configuration. The figure shows the four blocks the teacher sees in causal order: the original *prompt prefix* (identical to the student input up to the supervised step), a *hindsight block* that prepends the $G{-}1$ sibling rollouts' search-only skeletons together with their outcome labels (after a string-level deduplication on the (queries, outcome) tuple), the focal trajectory's *own future search spans* under future masking $\mathcal{M}$, and the focal trajectory's own *outcome label*, after which the model predicts the next token of the supervised search query.

(1) Prompt prefix *identical for student and teacher*

```
You are a helpful assistant... Question: What is the date of death of Alexander
of Masovia's father?
<think> First identify the father, then look up his death date. </think>
<search> Alexander of Masovia father </search>
<documents> [Doc 1: Alexander of Masovia] ...second son of Siemowit IV, Duke of
Masovia ...</documents>
```

(2) Hindsight block: sibling rollouts after dedup

```
\n[Trajectory Hindsight]:\n
[Sibling Rollout]: <search>Alexander of Masovia parents</search> ->
<search>Siemowit IV death</search> [Outcome]: Correct
[Sibling Rollout]: <search>Alexander of Masovia father date of death</search>
[Outcome]: Incorrect
[Sibling Rollout]: <search>father of Alexander of Masovia</search> ->
<search>Siemowit IV biography</search> [Outcome]: Incorrect
[Sibling Rollout]: <search>Alexander of Masovia</search> -> <search>Siemowit IV
Duke of Masovia</search> -> <search>Siemowit IV January 1426</search> [Outcome]:
Correct
```

(3) Focal future (own search spans only, future-masked $\mathcal{M}$)

```
<search>Siemowit IV Duke of Masovia date of death</search>
```

(4) Focal outcome label

```
\n[Outcome]: Correct\n
```

(5) Supervised action span $\tau_{i,p:p+|a|}$ *JSD applied at $\mathcal{Q}_\tau$ tokens*

```
<search> Siemowit IV Duke of Masovia date of death </search>
```

Figure 7: Teacher input layout for one supervised step. Blocks (1) and (5) are the standard student input split around the action being supervised. Blocks (2)–(4) are inserted between them and constitute the hindsight context $h(\mathcal{G})$: the $G{-}1$ sibling rollouts in (2) (search spans only, each tagged with its outcome label, with $G{=}5$ in our default configuration), the focal trajectory's own future search spans in (3) (other span types removed by future masking), and the focal trajectory's own outcome label in (4). Blue tokens are search-query content, the positions $\mathcal{Q}_\tau$ at which the JSD objective in Eq. 6 is applied; outcome labels are colored green for CORRECT and red for INCORRECT. At inference time the yellow-shaded blocks (2)–(4) are absent and the teacher view collapses back to the student view.

Three details of this layout are worth noting. First, this is the concrete token-level realization of the abstract context $c^{\text{tch}}(\tau_i; \mathcal{G}) = [h(\mathcal{G}); \tau_i]$ in Eq. 4: the hindsight block is placed immediately before the supervised action positions while keeping the question prefix unchanged relative to the student input, so the diff between teacher and student contexts at each supervised position is exactly the highlighted hindsight block, and every supervised position has causal access to all of $h(\mathcal{G})$. Second, the focal trajectory's own future search spans (block 3) are included in the hindsight, separated from the sibling rollouts; this is what makes the leave-one-out variant in §3.3.1 a meaningful probe, since it removes exactly this block while preserving block 2. Third, future masking $\mathcal{M}$ is applied uniformly to both the sibling skeletons (block 2) and the focal future (block 3), so that THINK reasoning, retrieved DOCUMENTS, and final ANSWER spans are all stripped out before reaching the teacher; without this masking the teacher would see passages whose content makes the gold answer extractable directly, collapsing the search-position distributions onto retrieval-skipping continuations as quantified in §3.3.1.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and §1 accurately reflect the paper's contributions: a hindsight self-distillation framework requiring no external teacher, a token-level JSD objective at search-query positions, and empirical validation across seven QA benchmarks at multiple scales. All numerical claims in the abstract are supported by Tables 1 and 2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: §4 explicitly discusses limitations: the dependence on reliably scorable gold answers for outcome labels, and signal degeneration when all rollouts in a group fail.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper presents an empirical method and does not contain formal theoretical results, theorems, or proofs requiring stated assumptions. Formulations in §2 are definitional (objective and conditioning constructions).

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The method is fully specified in §2 (trajectory format, hindsight block construction, JSD objective, total loss). The experimental setup is detailed in §3.1, with complete hyperparameters in Appendix D, dataset splits in Appendix C, training cost breakdown in Appendix E, and a teacher-input schematic in Appendix I.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Codes will be open-sourced upon acceptance. All datasets used are publicly available through the FlashRAG collection, and the implementation is built on the open-source VeRL framework, with the complete configuration provided in Appendix D to support reproducibility.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: §3.1 specifies the training data, evaluation benchmarks, retrieval setup, and backbone models. Appendix D (Table 6) provides the complete hyperparameter configuration including optimizer settings, learning rate, batch size, group size, and all four SD-Search-specific hyperparameters, along with the retrieval configuration.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Appendix B (Table 4) reports mean and standard deviation across five random seeds for SD-Search-Base, AutoRefine-Base, and Thinker-Instruct on all seven benchmarks. The error bars are 1-sigma standard deviations across seeds (capturing variability from random initialization and rollout sampling), and the appendix discusses how the seed-induced variance compares to the gaps between methods, including which per-benchmark gaps remain consistent across seeds.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The Computational Cost paragraph in §2.4 reports the GPU type and end-to-end training wall-clock for both SD-Search and the AutoRefine baseline. Appendix E (Table 7) further provides a per-step wall-clock breakdown across all training stages and contrasts the cost profile with the process-supervision baselines.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics. The work uses publicly available QA datasets and open-source language models, involves no human subjects, no personally identifiable information, and no scraped data beyond standard academic benchmarks.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper proposes a training methodology for credit assignment in search-augmented reasoning agents on standard QA benchmarks. It is foundational research on reinforcement learning supervision and does not introduce new capabilities or applications with direct paths to societal impacts beyond those already present in retrieval-augmented LLMs.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release new pre-trained models, image generators, or scraped datasets. The artifact to be released upon acceptance is training infrastructure for an existing class of methods (RL-trained retrieval agents) and poses no specific high-risk misuse potential beyond the underlying open-source backbone models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets used in this paper are properly cited, including the Qwen2.5 backbones, the E5-base-v2 retriever, the VeRL framework, and the seven QA datasets, as listed in §3.1 and Appendix C. All baseline methods compared against in Tables 1 and 2 are cited with their original publications.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new datasets or model checkpoints are released with this submission. The training code will be released upon acceptance and will be documented through the method description in §2, the hyperparameters in Appendix D, and the teacher-input schematic in Appendix I.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

Justification: This work does not involve any crowdsourcing or research with human subjects. All experiments use publicly available QA datasets with pre-existing question-answer pairs.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This work does not involve any research with human subjects, and no IRB approval is required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs (Qwen2.5 backbones) are the subject of study in this research, not a non-standard tool used as part of the methodology. We did not use LLMs for code development or writing assistance in this work.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.