π-PLAY: MULTI-AGENT SELF-PLAY VIA PRIVILEGED SELF-DISTILLATION WITHOUT EXTERNAL LABELED QA DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep search agents remain difficult to train at scale due to limited labeled data and weak credit assignment in long-horizon search. Self-play offers a scalable way to reduce data dependence, but conventional approaches optimize models only with sparse outcome rewards, limiting learning efficiency. We find that the task-generation process in self-play can itself serve as a scalable source of privileged supervision for self-distillation. Specifically, self-play naturally produces a Question Construction Path (QCP) during task generation, which captures the reverse solution process and provides rich intermediate signals for self-distillation without curated privileged information. Leveraging this insight, we propose **P**rivileged **I**nformation Self-**Play** (π-Play), a self-evolution framework that couples self-play with self-distillation by directly exploiting privileged information generated during task construction. In π-Play, self-play generates training tasks together with QCPs for self-distillation in a low-cost and scalable manner, while self-distillation converts QCPs into token-level supervision to accelerate self-play evolution. This closed-loop interaction enables more efficient model improvement without relying on labeled QA data or curated privileged information. Across multiple benchmarks and model scales, data-free π-Play surpasses fully supervised search agents and improves evolutionary efficiency by **1.5–3.1**$\times$ over conventional self-play.

1 INTRODUCTION

Deep search agents leverage the reasoning capabilities of large language models (LLMs) and external search engines to perform multi-turn retrieval and analysis for complex questions, emerging as a promising paradigm for information acquisition (Shao et al., 2024; Guo et al., 2025; Zheng et al., 2025b). Recent advances have shown that reinforcement learning (RL) can substantially improve both reasoning and search behaviors, enabling LLM agents to tackle increasingly challenging information-seeking tasks (Shao et al., 2024; Guo et al., 2025). However, training strong search agents at scale remains fundamentally bottlenecked by data (Yue et al., 2026b; Jin et al., 2025; Wu et al., 2025; Li et al., 2025; Gao et al., 2025). Supervised pipelines rely on labeled data and costly expert trajectories, while outcome-supervised RL often suffers from sparse rewards and poor credit assignment, especially in multi-turn search scenarios (Jin et al., 2025; Zheng et al., 2025a).

A promising direction for alleviating data dependence is self-evolution through *self-play*. In existing self-play frameworks, models of the same scale alternately play the roles of examiner and student: the examiner autonomously constructs training tasks, while the student learns by solving those tasks (Yue et al., 2026b; Lu et al., 2026; Chen et al., 2024). This paradigm allows the model to bootstrap its own curriculum without relying on manually collected training data, offering a promising route toward scalable and autonomous improvement. However, existing self-play methods typically optimize the student using only sparse outcome rewards, resulting in inefficient credit assignment for multi-turn search tasks (Huang et al., 2026; Yue et al., 2026b). Notably, self-play produces more than just the final question-answer (QA) pair $(q, o^\star)$. As shown in Fig. 2, the examiner also produces an informative yet previously overlooked intermediate artifact, the *question construction path* (QCP), denoted by c, which records the multi-turn interaction process through which factual evidence is progressively composed into a question. Hence, the examiner's output is more accurately represented as a triplet $(q, c, o^\star)$, where the c captures a reverse solution process from supporting

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

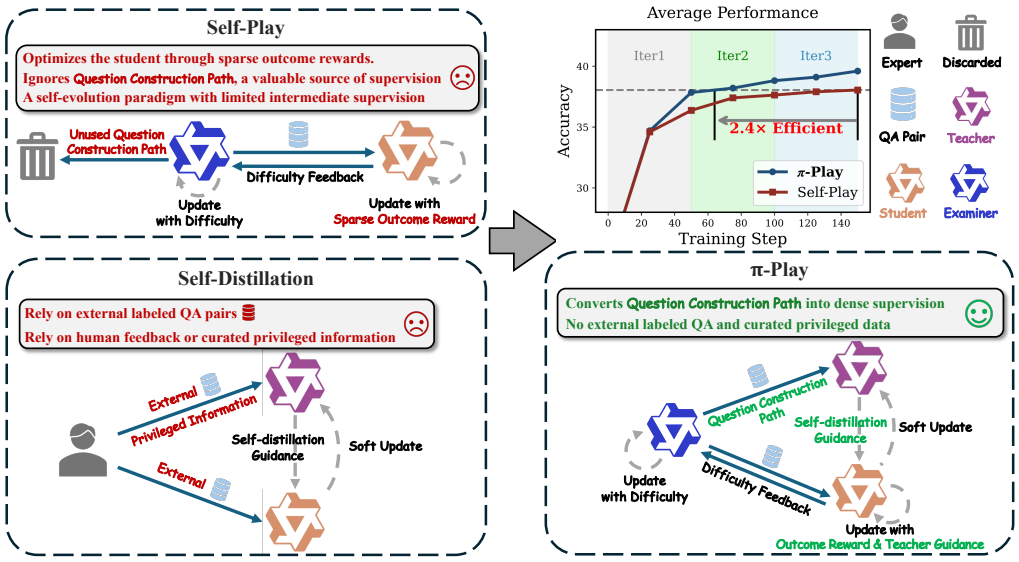

Figure 1: **Comparison of π-Play with other self-evolution frameworks.** Self-play lacks sufficient intermediate supervision, while self-distillation depends on labeled QA data and curated privileged information. π-Play integrates their complementary strengths in a closed-loop self-evolution framework. In π-Play, self-play generates training tasks together with their QCPs, providing both training data and privileged information for self-distillation in a low-cost and scalable manner, while self-distillation converts QCPs into token-level supervision to accelerate self-play evolution. This integration enables more efficient model improvement without relying on external labeled QA data.

evidence to the question, providing rich intermediate supervision for reasoning and search. Unfortunately, existing self-play methods largely ignore the QCP (Yue et al., 2026b; Huang et al., 2026), as its reverse construction trace cannot directly serve as an SFT target for the student, leaving this potentially dense supervisory signal unexploited.

Another line of self-evolution, *self-distillation*, addresses the credit assignment problem by employing high-quality privileged information (Hübotter et al., 2026; Shenfeld et al., 2026; Zhao et al., 2026; Ye et al., 2026; Penaloza et al., 2026). Unlike conventional on-policy distillation (Agarwal et al., 2024; Gu et al., 2024; Yue et al., 2025; Lu & Lab, 2025), which often relies on an external teacher, self-distillation typically uses a teacher of the same scale as the student, augmented with privileged information to provide dense token-level supervision. Common sources of such privileged information include expert demonstrations (Penaloza et al., 2026; Shenfeld et al., 2026), external (human) feedback (Hübotter et al., 2026; Wang et al., 2026), and prior knowledge (Ye et al., 2026; Sang et al., 2026). Prior studies have shown that such privileged supervision can significantly enhance learning efficiency. However, obtaining high-quality privileged information is often costly and difficult to scale, as it may require human experts or stronger models (Shenfeld et al., 2026; Zhao et al., 2026; Ye et al., 2026). Moreover, existing self-distillation methods typically still rely on curated QA pairs for optimization. Thus, while self-distillation provides dense supervision, its reliance on externally sourced privileged information and curated training data limits its scalability.

This work is motivated by a key observation: **the task-generation process in self-play can itself serve as a scalable source of privileged supervision for self-distillation.** Since QCP records how factual evidence is progressively composed into a question, this privileged information can help a same-scale teacher to provide informative token-level guidance to the student. Based on this insight, we propose **P**rivileged **I**nformation Self-**Play** (π-Play), a multi-agent self-evolution framework in which an examiner first generates training tuples $(q, c, o^\star)$, and a teacher conditioned on both the question q and its construction path c provides token-level distillation signals to guide the student (Fig. 2). By converting QCP into dense supervision, π-Play complements sparse outcome rewards with fine-grained learning signals, enabling more efficient credit assignment for long-horizon search and reasoning. Across multiple training iterations, π-Play consistently surpasses supervised baselines and self-play baselines and exhibits stronger evolutionary efficiency (Fig.1). In summary, our main contributions are as follows:

- We propose π-Play, a novel self-evolution framework that **unifies self-play and self-distillation to exploit their complementary strengths**. By combining autonomous task generation with dense teacher supervision, π-Play enables more efficient model improvement without relying on external labeled training data.
- Furthermore, we **propose Question Construction Path (QCP) as a new source of high-quality privileged information**. Self-play can generate such privileged information during task construction **in a low-cost and scalable manner**. This enables **scalable self-distillation** without relying on human feedback or curated privileged data.
- Extensive experiments show that π-Play outperforms fully supervised search agents and improves evolutionary efficiency by **1.5–3.1**$\times$ over conventional self-play. On Qwen3-4B, π-Play **surpasses the converged performance of self-play baselines after only one iteration**.

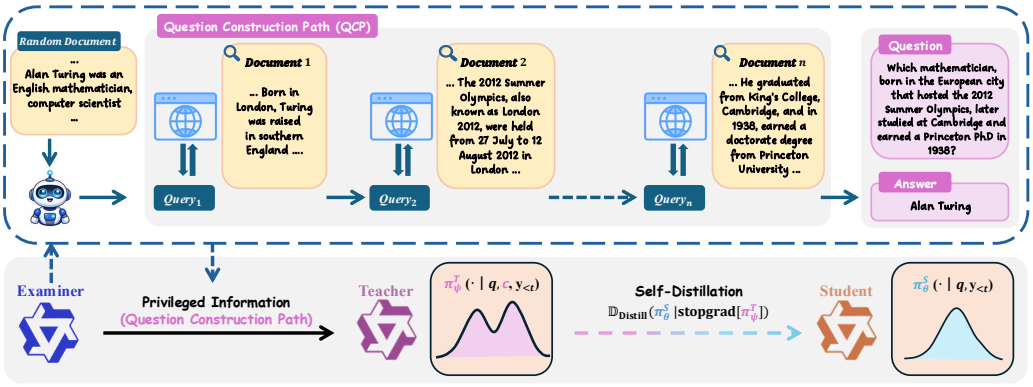

Figure 2: **Overview of QCP-guided self-distillation in π-Play.** The examiner is equipped with search tools and interacts with the search engine to obtain factual information, ensuring the correctness of both the synthesized QA pairs and their construction paths c. The teacher policy π_ψ^T leverages QCP as additional context to provide token-level supervision to the student policy π_θ^S along the student's rollout y, by minimizing the per-token reverse KL divergence $\mathbb{D}_{\text{Distill}}\left(\pi_\theta^S \,\|\, \text{stopgrad}[\pi_\psi^T]\right)$.

2 Related Work

2.1 Deep Search Agents

Deep search agents leverage the reasoning capabilities of large language models and external search engines to perform multi-turn retrieval and analysis for complex questions, emerging as a promising paradigm for information acquisition (Song et al., 2025). Recent work has employed RL to further enhance both reasoning and access to up-to-date knowledge, enabling LLMs to tackle complex tasks more effectively (Shao et al., 2024; Guo et al., 2025; Fu et al., 2026). Some agentic RL works, including R1-Searcher (Song et al., 2025), Search-R1 (Jin et al., 2025), and DeepResearcher (Zheng et al., 2025b) further enhance question-answering capabilities but remain constrained by limited training data. To scale agentic RL, some pipelines (Wu et al., 2025; Li et al., 2025; Gao et al., 2025) employ offline question-synthesis strategies, yet they do not explicitly couple task generation with the evolving capability of the solver. In contrast, self-play enables search agents to jointly generate and solve tasks without human annotations (Huang et al., 2026; Lu et al., 2026; Yue et al., 2026b), reducing reliance on manual supervision and allowing agentic RL to scale to broader scenarios.

2.2 Self-play for LLMs

Self-play enables LLMs to autonomously improve their reasoning and problem-solving capabilities by iteratively generating tasks and learning from their own experiences (Liu et al., 2026; Huang et al., 2026; Lu et al., 2026; Liu et al., 2025; Yue et al., 2026b). Early approaches leverage the LLM as both generator and evaluator, refining its policy without human supervision (OpenAI et al., 2021;

Chen et al., 2024). For instance, self-rewarding LLMs employ iterative training loops where the model judges its own outputs to construct preference data for optimization (Yuan et al., 2024). More recent frameworks, such as R-Zero (Huang et al., 2026), SQLM (Chen et al., 2026) and Dr.Zero (Yue et al., 2026b), typically involve only two roles: an *examiner*, which generates QA pairs, and a *student*, which is optimized via outcome-based RL on these self-generated tasks. These methods typically rely on sparse outcome rewards and do not exploit the construction paths produced during question generation. Such sparse feedback makes it difficult to distinguish effective from ineffective tool-use behaviors (Zheng et al., 2025a; Zhang et al., 2026; Yue et al., 2026a), leading to inefficient credit assignment and slow policy improvement. In contrast, π-Play incorporates these construction paths into student optimization as token-level supervision.

2.3 Privileged Information Self-distillation for LLMs

Self-distillation is a training paradigm that enables a student model to improve by learning from its own generated outputs. In this process, a teacher model evaluates students' rollouts and provides supervision signals at the token level to guide students. Unlike on-policy distillation (Agarwal et al., 2024; Gu et al., 2024; Yue et al., 2025; Lu & Lab, 2025), self-distillation does not rely on a larger external teacher. Instead, the teacher typically shares the same architecture and scale as the student, but is augmented with privileged information to provide reliable supervision (Hübotter et al., 2026; Shenfeld et al., 2026; Zhao et al., 2026; Ye et al., 2026; Penaloza et al., 2026). Such dense supervision has been shown to effectively enhance the student model's learning efficiency (Hübotter et al., 2026; Shenfeld et al., 2026; Ye et al., 2026; Penaloza et al., 2026). However, obtaining high-quality privileged information is often nontrivial. In several prior works, privileged information is typically constructed with the help of human experts or stronger models (Shenfeld et al., 2026; Zhao et al., 2026; Ye et al., 2026), including expert demonstrations (Penaloza et al., 2026; Shenfeld et al., 2026) and prior knowledge (Ye et al., 2026; Sang et al., 2026), which limits the scalability of self-distillation. In π-Play , the privileged signal provided to the teacher model is derived from the task-construction process in self-play, thereby avoiding dependence on externally provided privileged information.

3 Methods

3.1 Self-Play with Privileged Information

As shown in Fig. 1, we employ a self-evolution framework based on efficient multi-agent collaboration, in which all models function as search agents capable of leveraging external knowledge. Let y denote a full response (or rollout), and $o(y)$ denote the final answer extracted from y; for brevity, we write $o_i := o(y_i)$. The *examiner* π_ϕ^E, the *teacher* π_ψ^T, and the *student* π_θ^S are each optimized according to their respective objectives:

Optimization Objective

- **Examiner** (Sec. 3.2): Learn to generate diverse and challenging questions.

$$\max_\phi \mathbb{E}_{(q,c,o^\star)\sim\pi_\phi^E(\cdot),\{y_k\}_{k=1}^n\sim\pi_\theta^S(\cdot|q)}[r^d(o^\star,\{o_k\}_{k=1}^n)], \tag{1}$$

- **Teacher** (Sec. 3.4): Incorporate privileged information to generate more accurate rollouts while avoiding an excessive discrepancy between the teacher and the student.

$$\max_\psi \mathbb{E}_{(q,c,o^\star)\sim\pi_\phi^E(\cdot),y\sim\pi_\psi^T(\cdot|q,\mathbf{c})}[\mathbb{I}(o=o^\star)-\beta\mathbb{D}_{\mathrm{KL}}\big(\pi_\psi^T(\cdot\mid q,c)\,\|\,\mathrm{stopgrad}[\pi_\theta^S(\cdot\mid q)]\big)], \tag{2}$$

- **Student** (Sec. 3.3): Learns efficiently through a combination of outcome reward and teacher guidance, enhancing its evolutionary efficiency.

$$\max_\theta \mathbb{E}_{(q,c,o^\star)\sim\pi_\phi^E(\cdot),y\sim\pi_\theta^S(\cdot|q)}[\mathbb{I}(o=o^\star)-\lambda\mathbb{D}_{\mathrm{Distill}}\big(\pi_\theta^S(\cdot\mid q)\,\|\,\mathrm{stopgrad}[\pi_\psi^T(\cdot\mid q,c)]\big)], \tag{3}$$

where r^d denotes the difficulty reward and $\mathbb{I}$ is the indicator function. To generate questions of moderate difficulty (i.e., suitable for the student's current capability), the examiner's reward r^d is defined over the distribution of predicted answers. If all predictions (i.e., $\{o_k\}_{k=1}^n$) are correct, the

question is considered trivial, whereas if none are correct, the question is likely too difficult for the student. To jointly optimize the examiner, teacher, and student, we adopt an alternating optimization loop that couples question generation, teacher guidance, and student improvement into a unified self-evolution process. As the student becomes stronger, it drives the examiner to generate increasingly challenging questions. Meanwhile, the student's updated behavior is softly propagated to the teacher, allowing the teacher to track the student while remaining more stable. This iterative process establishes a continuously evolving curriculum. All three agents are initialized from the same base LLM and rely exclusively on the search tool to access external knowledge. The entire training process strictly adheres to a *training data-free setting*, avoiding any demonstrations, questions, or annotated answers from external sources or human experts.

3.2 Examiner Training

To utilize student feedback for training the examiner, we employ a difficulty reward function r^d that encourages both verifiability (the task must be solvable) and difficulty (the task must not be trivial) (Yue et al., 2026b). Specifically, we leverage the student's success rate on the generated questions as a proxy for these properties. Let k denote the number of correct solutions out of n sampled attempts. We penalize cases where the student either fails completely ($k = 0$) or succeeds trivially ($k = n$), encouraging the examiner to generate questions of moderate difficulty. The reward is defined as:

$$r^d(o^\star, \{o_i\}_{i=1}^n) = \mathbb{I}(0 < k < n)\frac{n-k}{n-1} + r^f, \tag{4}$$

where $k = \sum_{i=1}^n \mathbb{I}(o_i = o^\star)$. The difficulty reward is maximized when exactly one solution is correct and decays linearly as the number of correct predictions increases. Here, r^f denotes a format reward that encourages the examiner to interleave reasoning with search during task generation, so that the synthesized questions are both factually grounded and accompanied by an informative construction path.

Following prior work, we employ hop-grouped relative policy optimization (Yue et al., 2026b) to train the examiner. Specifically, we estimate advantages by grouping structurally similar questions within a batch. Generated QA pairs are clustered according to their *cross-hop complexity*, measured by the number of hops $h \in \mathcal{H}$. Intuitively, questions with fewer hops are typically simpler, whereas higher-hop questions demand extensive search and multi-turn reasoning. This hop-specific normalization of returns produces low-variance advantage estimates while avoiding the computational cost of sampling multiple candidate questions per prompt.

$$\mathcal{J}_{\text{Examiner}}(\phi) = \mathbb{E}_{\{(q_i,c_i,o_i^\star)\sim\pi_\phi^E(\cdot),\{y_{i,k}\}_{k=1}^n\sim\pi_\theta^S(\cdot|q_i)\}_{i=1}^N}\left[\frac{1}{N}\sum_{h\in\mathcal{H}}\sum_{i\in\mathcal{I}_h}\log\pi_\phi^E A_{i,h} - \beta\mathbb{D}_{\text{KL}}(\pi_\phi^E\|\pi_{\text{ref}})\right], \tag{5}$$

where N is batch size and β controls the strength of KL regularization. The advantage of each generated QA triplet $(q_i, c_i, o_i^\star)$ is obtained by hop-wise reward normalization $A_{i,h} = \frac{r_i^d - \mathbb{E}_{j\in\mathcal{I}_h}[r_j^d]}{\sqrt{\mathbb{V}\text{ar}_{j\in\mathcal{I}_h}[r_j^d]+\delta}}$, where $\mathcal{I}_h$ is the set of questions in the hop group h, and δ is a small constant for numerical stability.

3.3 Student Training

For student training, we sample training tuples $(q, c, o^\star)$ from the examiner policy π_ϕ^E. The student policy π_θ^S then generates candidate rollouts for each question and is optimized using a hybrid training objective. Importantly, the construction path c is only visible to the teacher model, ensuring that the teacher can produce reliable guidance while the student learns to solve the task without direct access to privileged information. Formally, the student objective is defined as follows:

$$\mathcal{J}_{\pi\text{-play}}(\theta) = \mathbb{E}_{(q,c,o^\star)\sim\pi_\phi^E,\{y_i\}_{i=1}^G\sim\pi_\theta^S(\cdot|q)}\left[\underbrace{\frac{1}{G}\sum_{i=1}^G\sum_{t=1}^{|y_i|}\mathcal{L}_{i,t} - \beta\,\mathbb{D}_{\text{KL}}(\pi_\theta^S\|\pi_{\text{ref}})}_{\text{Learning from outcome reward}} - \underbrace{\lambda\mathbb{D}_{\text{Distill}}\left(\pi_\theta^S\,\|\,\pi_\psi^T\right)}_{\text{Teacher guidance}}\right], \tag{6}$$

$$\mathcal{L}_{i,t} = \min(w_{i,t}A_i, \text{clip}(w_{i,t}, 1-\epsilon, 1+\epsilon)\,A_i)\,, \tag{7}$$

The overall objective consists of two complementary components. The first term corresponds to group relative policy optimization (GRPO), which improves the student policy using outcome rewards derived from answer correctness (i.e., $r^e(q, o_i) = \mathbb{I}(o_i = o^\star)$). The importance weight $w_{i,t}$

and normalized advantage A_i are given by:

$$w_{i,t} = \frac{\pi_\theta^S(y_{i,t} \mid q, y_{i,<t})}{\pi_{\theta_{\text{old}}}^S(y_{i,t} \mid q, y_{i,<t})}, \quad A_i = \frac{r_i^e - \mathbb{E}_{j \in G}[r_j^e]}{\sqrt{\mathbb{V}\text{ar}_{j \in G}[r_j^e] + \delta}}. \tag{8}$$

By computing advantages from group statistics, GRPO reinforces successful rollouts while penalizing failed ones. The second term is a self-distillation objective, where the student policy is aligned with the teacher policy π_ψ^T along student's rollout. The distillation loss is defined as follows:

$$\mathbb{D}_{\text{Distill}}(\pi_\theta^S \,\|\, \pi_\psi^T) = \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \text{KL}(\pi_\theta^S(\cdot \mid q, y_{i,<t}) \| \text{stopgrad}(\pi_\psi^T(\cdot \mid q, \mathbf{c}, y_{i,<t}))) \tag{9}$$

The full formulation of the distillation loss is provided in Appendix D. Since the teacher has access to privileged information in the form of the QCP, it can provide reliable token-level guidance. This dense supervision complements sparse outcome rewards through a favorable bias-variance trade-off (Schulman et al., 2016; Gu et al., 2017): outcome rewards are unbiased but high-variance, whereas teacher guidance may introduce modest bias while substantially reducing variance. Their combination enables more efficient credit assignment and faster policy improvement.

3.4 Teacher Updating

Although Eq. (2) defines the ideal teacher objective, directly optimizing it would introduce additional computational overhead. In practice, to provide stable teacher supervision while allowing the teacher to co-evolve with the student, we approximate this objective by updating the teacher parameters as an exponential moving average (EMA) of the student parameters (Hübotter et al., 2026; Penaloza et al., 2026):

$$\psi \leftarrow (1 - \tau)\psi + \tau\theta, \quad \tau \in (0, 1), \tag{10}$$

where τ controls the teacher update rate. This update keeps the teacher relatively stable while enabling it to gradually track the student's improvements over training.

3.5 The Co-Evolution Procedure of π-Play

In summary, we present π-Play, a data-free self-evolution framework that jointly optimizes the examiner, teacher, and student models (Fig. 1). In each iteration, the examiner generates QA pairs together with QCP, and is trained via student-derived difficulty feedback to produce challenging yet solvable questions. The student improves its search and reasoning abilities through training on QA pairs generated by examiner, while the teacher leverages the QCP as additional context to provide token-level guidance to the student. After each update, the teacher is softly aligned with the student, enabling both models to co-evolve. This alternating optimization loop forms a symbiotic feedback cycle in which stronger students drive examiner toward more challenging questions, while teacher guidance accelerates student improvement. The training process is summarized in Appendix B.

4 Experiments

4.1 Setup

Datasets & Models. We conduct experiments on models from the Qwen-3 series (Yang et al., 2025). The base models are Qwen3-4B, Qwen3-4B-Instruct, and Qwen3-8B. More details on experimental setups and training procedures can be found in Appendix A. We evaluate π-Play on three one-hop benchmarks NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), as well as four multi-hop QA benchmarks, including HotpotQA (Yang et al., 2018), 2WikiMQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023).

Baselines & Evaluation. To demonstrate the efficacy of π-Play, we compare it against a variety of baseline search agents: (1) *training-free*: ReAct (Yao et al., 2023); (2) *supervised RL*: Search-R1 (Jin et al., 2025) and ToolForge (Chen et al., 2025), and (3) *self-play*: Dr.Zero (Yue et al., 2026b) and SQLM* (Yue et al., 2026b), a search version of SQLM (Chen et al., 2026). All models are evaluated using the exact match scores with identical search engine (E5-base (Wang et al., 2022)) and corpus settings (English Wikipedia dump (Karpukhin et al., 2020)).

Table 1: **The main results of π-Play. Bold** values indicate the best result; underline values denote the second-best. π-Play outperforms both supervised and self-play search agents.

	General QA			Multi-Hop QA				
	NQ	**TriviaQA**	**PopQA**	**HotpotQA**	**2WikiMQA**	**MuSiQue**	**Bamboogle**	**Total**
				Qwen3-4B				
ReAct	7.2	16.6	10.3	11.2	12.7	1.6	8.8	68.4
Search-R1	35.2	57.3	39.5	30.7	28.9	10.5	**37.6**	239.7
SQLM*	39.9	60.4	42.1	**33.0**	30.8	**10.8**	29.6	246.6
Dr.Zero	40.4	61.2	44.3	31.0	29.3	7.8	31.2	245.2
π-Play	**41.1**	**62.0**	**44.8**	32.5	**31.5**	9.0	28.8	**249.7**
				Qwen3-4B-Instruct				
ReAct	18.4	45.5	28.4	22.8	18.0	6.3	27.2	166.6
Search-R1	39.5	62.6	41.3	34.9	24.8	12.9	**50.0**	266.0
SQLM*	39.0	62.2	41.9	34.1	27.3	10.2	35.2	249.9
Dr.Zero	41.3	61.6	**44.0**	38.4	29.8	**14.4**	36.8	266.3
π-Play	**41.8**	**63.2**	**44.0**	**38.5**	**32.3**	13.4	44.0	**277.2**
				Qwen3-8B				
ReAct	15.9	37.3	14.9	22.3	22.5	7.4	28.8	149.1
Search-R1	35.7	59.9	40.0	32.0	27.8	9.8	37.6	242.8
ToolForge	30.3	16.8	35.2	-	-	**37.8**	**48.0**	-
SQLM*	38.9	60.5	42.7	33.1	31.7	9.9	32.0	248.8
Dr.Zero	42.1	62.5	45.8	36.6	**34.8**	13.1	40.0	274.9
π-Play	**43.0**	**64.6**	**47.2**	**38.9**	34.2	12.4	40.0	**280.3**

4.2 Main Results

We first analyze the main evaluation results as reported in Table 1 and derive several key observations from them: (1) *Strong Overall Performance:* π-Play achieves substantial improvements in overall performance over base LLM (e.g., ReAct), demonstrating strong robustness and generalization across diverse task types and model scales. The results further demonstrate the effectiveness and superiority of the multi-agent self-evolution framework for search agents. (2) *π-Play surpasses supervised RL methods.* π-Play delivers strong performance without using any training data. In terms of average performance, it surpasses the Search-R1 by 4.3%, 4.2%, and 15.4% on Qwen3-4B, Qwen3-4B-Instruct, and Qwen3-8B, respectively. This self-evolution framework demonstrates greater performance gains when instantiated with a stronger base LLM. (3) *π-Play surpasses self-play methods.* π-Play outperforms the self-play methods (SQLM* and Dr.Zero) across multiple model scales, benefiting from the additional guidance provided by the teacher to the student. Notably, the performance gains are even more substantial on multi-hop benchmarks, where complex multi-step reasoning is required. We attribute this advantage to token-level credit assignment from the teacher, which provides finer-grained and effective supervision for long-horizon reasoning.

4.3 Training Dynamics

To better understand the self-evolving dynamics of π-Play, we analyze model performance across training iterations, with detailed comparisons with Dr.Zero summarized in Table 2. These results lead to several key observations: (1) Across all three iterations, the student model shows a steady upward performance trend, consistently outperforming Dr.Zero after every iteration. This highlights the effectiveness of the examiner–teacher–student interplay and suggests that collaborative self-evolution is more effective than standard self-play by enabling more efficient information sharing among agents. (2) After the first training iteration, the student in π-Play achieves substantial gains in both search and reasoning abilities. In terms of performance, it already matches or even surpasses the converged perfor-

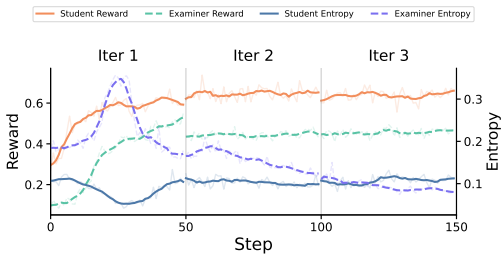

Figure 3: **Iterative reward and entropy dynamics of the examiner and student in π-Play with Qwen3-4B-Instruct**. Both reward and entropy reach a converged state by Iteration 3.

Table 2: **Learning dynamics of π-Play with increasing iterations.** On Qwen3-4B, π-Play surpasses Dr.Zero's converged performance after only a single iteration.

		General QA			Multi-Hop QA				
	Methods	**NQ**	**TriviaQA**	**PopQA**	**HotpotQA**	**2WikiMQA**	**MuSiQue**	**Bamboogle**	**Total**
Qwen3-4B									
Iter1	Dr.Zero	36.1	58.7	42.8	28.6	25.9	6.2	21.6	219.9
	π-Play	**38.6**	**59.4**	**43.5**	**32.2**	**31.4**	**10.7**	**32.8**	**248.6**
Iter2	Dr.Zero	**40.6**	**60.5**	**44.4**	30.8	29.1	8.0	28.8	242.2
	π-Play	38.1	58.4	42.3	**32.3**	**32.5**	**11.2**	**35.2**	**250.0**
Iter3	Dr.Zero	40.4	61.2	44.3	31.0	29.3	7.8	**31.2**	245.2
	π-Play	**41.1**	**62.0**	**44.8**	**32.5**	**31.5**	**9.0**	28.8	**249.7**
Qwen3-4B-Instruct									
Iter1	Dr.Zero	39.8	61.7	42.6	35.4	27.4	11.7	36.0	254.6
	π-Play	**40.7**	**62.1**	**42.9**	**36.7**	**30.1**	**13.2**	**39.2**	**264.9**
Iter2	Dr.Zero	39.3	61.7	43.2	**38.2**	28.4	**14.2**	38.4	263.4
	π-Play	**41.2**	**62.5**	**43.5**	37.8	**32.5**	**14.2**	**40.0**	**271.7**
Iter3	Dr.Zero	41.3	61.6	**44.0**	38.4	29.8	**14.4**	36.8	266.3
	π-Play	**41.8**	**63.2**	**44.0**	**38.5**	**32.3**	13.4	**44.0**	**277.2**

mance of Dr.Zero after three iterations, demonstrating the superior evolutionary efficiency. (3) After the second iteration, the improvement trends begin to vary across model sizes. While Qwen3-4B-Instruct continues to show modest gains, Qwen3-4B drops slightly from 250 to 249.7, suggesting that performance has started to plateau. Beyond this point, further iterations bring only marginal or no additional improvements across model sizes. In summary, these dynamics validate the design of π-Play, demonstrating that the introduction of the privileged teacher brings significant benefits.

5 Further Analysis

In this section, we analyze the importance of the QCP, the contribution of its individual components, the co-evolution dynamic of the examiner, and the search behavior of the student. Further experiments, including **extensive ablation studies, efficiency analysis, multi-seed statistical analysis, and training cost analysis**, are provided in **Appendix C**.

5.1 Ablation Study on the Question Construction Path

Table 3: **Ablation study on the question construction path (QCP).** π-Play w/o Distillation denotes the variant of π-Play without teacher-guided distillation loss. Variants of the form π-Play w/ [Privileged Info] replace QCP with different forms of privileged information for self-distillation.

	General QA			Multi-Hop QA				
	NQ	**TriviaQA**	**PopQA**	**HotpotQA**	**2WikiMQA**	**MuSiQue**	**Bamboogle**	**Total**
π-Play w/o Distillation	41.2	62.3	43.6	37.7	29.0	13.3	38.4	265.5
π-Play w/ GT	42.0	62.0	**44.5**	37.7	29.4	13.3	35.2	264.1
π-Play w/ GT+Hop	**42.1**	61.9	42.9	37.7	29.1	**14.0**	39.2	266.9
π-Play w/ Partial QCP	41.6	62.1	43.3	38.3	31.0	13.6	43.2	273.1
π-Play w/ QCP (Ours)	41.8	**63.2**	44.0	**38.5**	**32.3**	13.4	**44.0**	**277.2**

As shown in Table 3, we further evaluate the effectiveness of QCP as privileged information by comparing it with several alternative forms of teacher-side privileged context, including the ground-truth answer alone (π-Play w/ GT), the ground-truth answer combined with the question hop count (π-Play w/ GT+HOP), and a partial QCP obtained by randomly truncating half of the original QCP (π-Play w/ Partial QCP). Among these variants, using GT yields the worst performance, which is nearly comparable to the variant without a teacher model (π-Play w/o Distillation). This is because GT contains only the final answer and provides little information about the underlying logic used to construct the question, making it difficult for the teacher to provide effective guidance to the student. In contrast, the full QCP achieves the best performance, while Partial QCP performs second best. Moreover, Partial QCP still substantially outperforms both GT and GT+HOP, further highlighting the effectiveness of QCP as privileged information.

5.2 Ablation Study on the Answer Leakage

As shown in Table 4, we separately removed the path and the final answer from the QCP and evaluated the resulting performance changes. As shown in the following table, removing the final answer and answer-bearing snippets (QCP w/o Answer) causes only a minor degradation, whereas removing the path (QCP w/o Path) leads to a larger drop. This suggests that the construction path contributes more to the performance gain than answer-related information in QCP.

Table 4: **Ablation study on the answer leakage.**

	General QA			Multi-Hop QA				
	NQ	**TriviaQA**	**PopQA**	**HotpotQA**	**2WikiMQA**	**MuSiQue**	**Bamboogle**	**Total**
QCP w/o Path	**42.1**	61.9	42.9	37.7	29.1	**14.0**	39.2	266.9
QCP w/o Answer	41.3	62.2	43.9	38.4	32.1	13.4	42.4	273.7
QCP	41.8	**63.2**	**44.0**	**38.5**	**32.3**	13.4	**44.0**	**277.2**

5.3 Evolution of Question Difficulty

To understand the co-evolutionary dynamic of the examiner, we examined how the tasks it generated changed across iterations. After each of the training iterations, we sampled 2000 questions from its policy, creating three distinct evaluation sets: $\mathcal{D}_{\text{step 50}}$, $\mathcal{D}_{\text{step 100}}$, and $\mathcal{D}_{\text{step 150}}$. As shown in Table 5, the examiner generates progressively more challenging questions as training proceeds. This is evidenced by the performance of a fixed solver on these evolving question sets: for instance, the static student (Step 50) drops from 57.1 on $\mathcal{D}_{\text{step 50}}$ to 45.1 on $\mathcal{D}_{\text{step 150}}$. This suggests that examiner successfully increases task difficulty over the course of training.

Table 5: **Co-evolutionary dynamic of the examiner.** We report the student's accuracy on the datasets generated by examiner at different steps.

	Performance of Evaluated Model		
	Base Model	Student(step 50)	Student(step 100)
$\mathcal{D}_{\text{step 50}}$	27.5	57.1	58.4
$\mathcal{D}_{\text{step 100}}$	27.0	51.9	57.3
$\mathcal{D}_{\text{step 150}}$	22.9	45.1	53.3

Table 6: **Search behavior analysis.** We report the average number of search actions and the average query redundancy across seven QA benchmarks.

	Accuracy ↑	**Search Count ↓**	**Query Redundancy ↓**
SQLM*	35.7	2.3	0.43
Dr.Zero	38.0	2.4	0.48
π-Play	**39.6**	**1.9**	**0.37**

5.4 Search Behavior Analysis

To assess the effect of QCP-guided self-distillation on search behavior, we quantitatively analyze search count and query redundancy across seven QA benchmarks. As shown in Table 6, compared with conventional self-play methods such as SQLM* and Dr.Zero, π-Play achieves higher accuracy with fewer search actions, indicating more efficient behavior. The same trend holds for query redundancy. Fine-grained supervision leads to lower redundancy and more effective queries.

6 Conclusion

In this work, we propose π-Play , a novel self-evolution framework that unifies self-play and self-distillation to exploit their complementary strengths. By combining autonomous task generation with dense teacher supervision, π-Play enables more efficient model improvement without relying on external labeled training data. Our results further demonstrate that QCPs naturally generated during self-play can serve as high-quality privileged information for self-distillation, highlighting a promising direction for exploring more efficient and scalable sources of privileged information.

Future work While π-Play provides a new framework for multi-agent self-evolution, our current experiments instantiate it with a foundational self-distillation pipeline. This design helps measure the effectiveness of the QCP-guided self-distillation, but it may not fully exploit recent advances in self-distillation. Consequently, the evolutionary efficiency and supervision quality of π-Play could be further enhanced by incorporating more advanced on-policy distillation and self-distillation techniques. Exploring such integrations is an important direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools solely to improve the clarity and language of the manuscript, assist with literature retrieval, and support the development of experimental code. We did not use generative AI tools for research ideation or execution, interpretation of results, drafting substantive sections of the manuscript, generating synthetic datasets, or proving mathematical claims. All AI-assisted outputs were manually reviewed and verified by the authors to ensure their accuracy and consistency with our intended meaning. We take full responsibility for the final content of this work.

REFERENCES

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 *The Twelfth International Conference on Learning Representations*, pp. 1–18, Vienna, Austria, 2024. OpenReview.net.

Hao Chen, Zhexin Hu, Jiajun Chai, Haocheng Yang, Hang He, Xiaohan Wang, Wei Lin, Luhang Wang, Guojun Yin, and Zhuofeng zhao. Toolforge: A data synthesis pipeline for multi-hop search without real-world apis. *arXiv preprint arXiv:2512.16149*, 2025.

Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Self-questioning language models. In *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*, pp. 1–14, 2026. URL https://openreview.net/forum?id=sWBBJ3WgmO.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 6621–6642, Vienna, Austria, 2024. PMLR.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. In *The Fourteenth International Conference on Learning Representations*, pp. 1–18, Rio de Janeiro, Brazil, 2026. OpenReview.net.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. In *NeurIPS 2025 First Workshop on Multi-Turn Interactions in Large Language Models*, pp. 1–15, 2025. URL https://openreview.net/forum?id=JSYCLJcniH.

Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *The Fifth International Conference on Learning Representations*, pp. 1–13, Toulon, France, 2017. OpenReview.net.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, pp. 1–24, Vienna, Austria, 2024. OpenReview.net.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

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*, pp. 6609–6625, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning LLM from zero data. In *The Fourteenth International Conference on Learning Representations*, pp. 1–18, Rio de Janeiro, Brazil, 2026. OpenReview.net.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. In *Proceedings of the 43rd International Conference on Machine Learning*, pp. 1–45, Seoul, South Korea, 2026. PMLR.

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.

Mandar Joshi, Eunsol Choi, Daniel S. 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*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781, Online, 2020. Association for Computational Linguistics.

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.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. Websailor: Navigating superhuman reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. Spice: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025.

Bo Liu, Simon Yu, Zichen Liu, Leon Guertler, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. SPIRAL: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, pp. 1–28, Rio de Janeiro, Brazil, 2026. OpenReview.net.

Hongliang Lu, Yuhang Wen, Pengyu Cheng, Ruijin Ding, Jiaqi Guo, Haotian Xu, Chutian Wang, Haonan Chen, Xiaoxi Jiang, and Guanjun Jiang. Search self-play: Pushing the frontier of agent capability without supervision. In *The Fourteenth International Conference on Learning Representations*, pp. 1–34, Rio de Janeiro, Brazil, 2026. OpenReview.net.

Kevin Lu and Thinking Machines Lab. On-policy distillation. Thinking Machines Lab: Connectionism, 2025. URL https://thinkingmachines.ai/blog/on-policy-distillation.

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*, pp. 9802–9822, Toronto, Canada, 2023. Association for Computational Linguistics.

OpenAI OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Peter Welinder, Ruben D'Sa, Arthur Petron, Henrique P. d. O. Pinto, Alex Paino, Hyeonwoo Noh, Lilian Weng, Qiming Yuan, Casey Chu, and Wojciech Zaremba. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv preprint arXiv:2101.04882*, 2021.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. In *Proceedings of the 43rd International Conference on Machine Learning*, pp. 1–40, Seoul, South Korea, 2026. PMLR.

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

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. Crisp: Compressed reasoning via iterative self-policy distillation. *arXiv preprint arXiv:2603.05433*, 2026.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *The Fourth International Conference on Learning Representations*, pp. 1–14, San Juan, Puerto Rico, 2016. OpenReview.net.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *Proceedings of the 43rd International Conference on Machine Learning*, pp. 1–17, Seoul, South Korea, 2026. PMLR.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

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.

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*, 2022.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhenglin Wang, Zhengwei Tao, Ding-Chu Zhang, Zekun Xi, Xiangru Tang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webdancer: Towards autonomous information seeking agency. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, pp. 134093—134121, San Diego, CA, 2025. Curran Associates Inc.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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*, pp. 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, pp. 1–33, Kigali, Rwanda, 2023. OpenReview.net.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 57905–57923, Vienna, Austria, 2024. PMLR.

Chuhuai Yue, Chengqi Dong, Yinan Gao, Hang He, Jiajun Chai, Wei Lin, and Guojun Yin. Promoting efficient reasoning with verifiable stepwise reward. In *Proceedings of the Fortieth AAAI Conference on Artificial Intelligence and Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence and Sixteenth Symposium on Educational Advances in Artificial Intelligence*, pp. 34530—34538, Singapore, 2026a. AAAI Press.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, pp. 1–36, San Diego, USA, 2025. Curran Associates Inc.

Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. Dr. zero: Self-evolving search agents without training data. In *Third Conference on Language Modeling*, pp. 1–25, San Francisco, CA, 2026b. OpenReview.net.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Zijie Zhao, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. CriticSearch: Fine-grained credit assignment for search agents via a retrospective critic. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12272–12290, San Diego, California, United States, 2026. Association for Computational Linguistics.

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. In *Proceedings of the 43rd International Conference on Machine Learning*, pp. 1–16, Seoul, South Korea, 2026. PMLR.

Xuhui Zheng, Kang An, Ziliang Wang, Yuhang Wang, and Yichao Wu. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21805–21830, Suzhou, China, 2025a. Association for Computational Linguistics.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. DeepResearcher: Scaling deep research via reinforcement learning in real-world environments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 414–431, Suzhou, China, 2025b. Association for Computational Linguistics.

A IMPLEMENTATION

In our experiments, we implement π-Play through alternating optimization over the examiner and the student-teacher modules. Following prior self-play work, the examiner generates 1-, 2-, 3-, and 4-hop questions with a default ratio of 4:3:2:1. In each iteration, we first train the examiner for 50 steps, then use it to generate QA data from the corresponding prompts, and subsequently train the student on the synthesized data for another 50 steps. Meanwhile, the teacher is soft-updated at every student training step. Consistent with prior self-play settings, we run only three iterations in total, yielding 150 total training steps for each model. Throughout training, we adopt a decayed λ schedule to gradually weaken teacher guidance, allowing the student to progressively move toward regions with higher EM reward. For Qwen3-4B and Qwen3-8B, we set λ to 0.03, 0.003, and 0.002 across the three iterations, respectively. For Qwen3-4B-Instruct, we set λ to 0.1, 0.03, and 0.03. The format reward r^f for the examiner is set in the same way as the proposer's format reward in Dr.Zero (Yue et al., 2026b). Full hyperparameter details are reported in Table 7, Table 8, and Table 9.

Table 7: **Examiner hyperparameter settings.**

Steps	50
Optimizer	AdamW
Optimizer Momentum	β_1, β_2 = 0.9, 0.999
Warmup Ratio	0.03
Weight Decay	0.01
Learning Rate	5e-7, 1e-6
Group size	1
Reward size	5
KL-Div	0
Total Train Batch Size	256
LR Scheduler	Constant with Warmup
Precision (WA)	BF16-mixed
Max Turn in Rollout	5
Max Sequence Length	4096

Table 8: **Student hyperparameter settings.**

Steps	50
Optimizer	AdamW
Optimizer Momentum	β_1, β_2 = 0.9, 0.999
Warmup Ratio	0.03
Weight Decay	0.01
Learning Rate	1e-6
Group size in GRPO	5
KL-Div in GRPO	0, 0.001
ϵ in GRPO	0.2
Total Train Batch Size	256
LR Scheduler	Constant with Warmup
Precision (WA)	BF16-mixed
Max Turn in Rollout	5
Max Sequence Length	8192

Table 9: **Teacher hyperparameter settings.**

Precision (WA)	BF16-mixed
Soft Update Weight τ	0.05
Max Sequence Length	8192

B ALGORITHM

In this section, we describe the complete training procedure of π-Play in detail, with the overall algorithm summarized in Algorithm 1.

Algorithm 1 The Evolution Process of π-Play

Require: Examiner model π_ϕ^E, Teacher model π_ψ^T, Student model π_θ^S, Base LLM π_{ref}

Initialize π_ϕ^E, π_ψ^T, and π_θ^S from Base LLM π_{ref}
for iteration $j \leftarrow 1$ to M **do**
  # Updating Examiner Model
  for step $k \leftarrow 1$ to W **do**
    Sample N QA triplets $\{(q_i, c_i, o_i^\star)\}_{i=1}^N$ from the examiner policy π_ϕ^E
    Update $\pi_\phi^E(\cdot)$ with Eq. (5)
  end for

  # Generate Training Data For Student
  Generate $\mathcal{D} = \{(q_i, c_i, o_i^\star)\}_{i=1}^{N_\mathcal{D}}$ from $\pi_\phi^E(\cdot)$

  # Updating Student and Teacher Model
  for step $k \leftarrow 1$ to W **do**
    Sample a batch $\mathcal{D}_b$ from $\mathcal{D}$
    for each QA triplets $(q, c, o^\star)$ in $\mathcal{D}_b$ **do**
      Sample G rollouts $\{y_i\}_{i=1}^G$ from the student policy $\pi_\theta^S(\cdot|q)$
    end for
    Update π_θ^S with Eq. (6) by teacher guidance; **Then** Soft Update π_ψ^T with Eq. (10)
  end for
end for
return Student model π_θ^S

C FURTHER ANALYSIS

C.1 EFFICIENCY ANALYSIS

We evaluate evolutionary efficiency by comparing the training steps required to reach the converged performance of the self-play baseline. As shown in Fig. 4, π-Play consistently reaches this performance level earlier across all model settings. In particular, the resulting efficiency gains are approximately $3.1\times$, $2.4\times$, and $1.5\times$ on Qwen3-4B, Qwen3-4B-Instruct, and Qwen3-8B, respectively. The gains are most evident in the early training stages, suggesting that QCP-guided self-distillation provides more informative learning signals than sparse outcome rewards. Overall, these results indicate that π-Play improves not only the final performance but also the evolutionary efficiency.

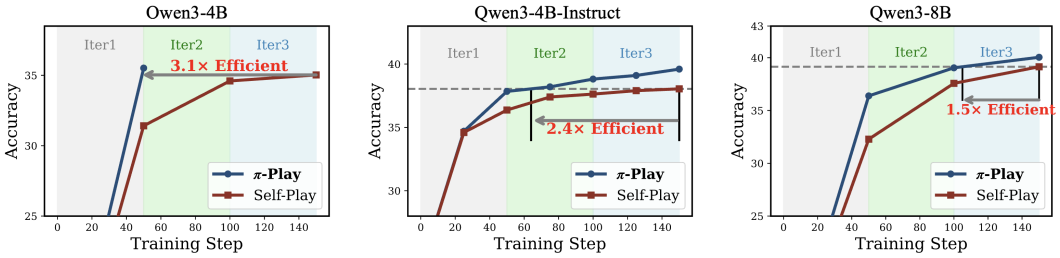

Figure 4: Evolutionary efficiency of π-Play compared with conventional self-play. We measure efficiency by the training steps required to reach the converged performance of self-play. π-Play achieves $3.1\times$, $2.4\times$, and $1.5\times$ higher step efficiency on Qwen3-4B, Qwen3-4B-Instruct, and Qwen3-8B, respectively.

C.2 Multi-Seed Statistical Analysis

As shown in Table 10, we report the average performance over three random seeds to provide a more reliable comparison with Dr.Zero. As shown in the table below, π-Play consistently achieves a higher total score than Dr.Zero, demonstrating stable performance improvements.

Table 10: **Statistical results of π-Play across three random seeds. Bold** values indicate the best result; underline values denote the second-best. π-Play consistently achieves a higher total score than self-play search agents.

	General QA			Multi-Hop QA				
	NQ	TriviaQA	PopQA	HotpotQA	2WikiMQA	MuSiQue	Bamboogle	Total
				Qwen3-4B				
SQLM*	39.9	60.4	42.1	**33.0**	30.8	10.8	29.6	246.6
Dr.Zero	**40.4**	**61.2**	**44.3**	31.0	29.3	7.8	31.2	245.2
π-Play	38.3±0.4	58.8±0.5	42.7±0.4	32.1±0.2	**31.4**±1.0	**11.5**±0.4	**34.9**±0.5	**249.8**±0.3
				Qwen3-4B-Instruct				
SQLM*	39.0	62.2	41.9	34.1	27.3	10.2	35.2	249.9
Dr.Zero	41.3	61.6	**44.0**	**38.4**	29.8	**14.4**	36.8	266.3
π-Play	**41.6**±0.6	**62.8**±0.4	43.8±0.1	**38.4**±0.2	**32.5**±0.3	13.9±0.3	**43.7**±1.2	**276.6**±1.7
				Qwen3-8B				
SQLM*	38.9	60.5	42.7	33.1	31.7	9.9	32.0	248.8
Dr.Zero	42.1	62.5	45.8	36.6	**34.8**	**13.1**	40.0	274.9
π-Play	**43.4**±0.4	**64.6**±0.2	**47.4**±0.2	**37.8**±0.9	33.4±0.8	12.8±0.9	**41.1**±2.6	**280.5**±0.5

Table 11: **Ablation study of λ. Bold** value indicates the top-performing result, while underline value denotes the second-best.

λ	General QA			Multi-Hop QA				
	NQ	TriviaQA	PopQA	HotpotQA	2WikiMQA	MuSiQue	Bamboogle	Total
				Iter 1				
0.100	40.1	61.8	42.5	35.5	30.1	12.8	40.8	263.6
0.050	39.8	**62.1**	42.2	34.9	**31.4**	11.8	**41.6**	263.8
0.030	40.1	61.1	42.7	33.7	30.4	9.85	40.0	257.9
0.010	38.5	61.6	42.5	35.4	26.6	12.4	35.2	252.2
0.005	40.1	61.1	42.0	34.9	31.1	10.6	32.0	251.8
0.000	**40.8**	61.3	42.3	34.3	27.9	11.1	31.2	248.9
Decay	40.7	**62.1**	**42.9**	**36.7**	30.1	**13.2**	39.2	**264.9**
				Iter 2				
0.100	40.1	61.5	42.5	36.0	30.5	13.8	**40.0**	264.4
0.050	40.6	62.2	43.6	36.2	31.3	12.8	35.2	261.9
0.030	40.9	61.7	43.2	37.4	28.9	13.6	38.4	264.1
0.010	41.0	62.3	43.7	37.2	30.4	12.8	36.8	264.2
0.005	**41.2**	**62.6**	**44.0**	37.3	28.1	13.0	36.8	263.0
0.000	**41.2**	61.6	43.4	36.3	26.1	13.8	36.0	258.4
Decay	**41.2**	62.5	43.5	**37.8**	**32.5**	**14.2**	**40.0**	**271.7**
				Iter 3				
0.100	40.8	61.7	42.3	35.6	31.0	13.4	40.0	264.8
0.050	41.0	61.3	42.8	35.7	30.5	14.0	36.0	261.3
0.030	**42.5**	62.0	44.3	38.0	29.8	**14.2**	37.6	268.4
0.010	**42.5**	62.1	**44.5**	36.5	32.0	12.0	36.0	265.6
0.005	42.4	62.4	43.9	37.6	28.6	13.8	38.4	267.1
0.000	41.2	62.3	43.6	37.7	29.0	13.3	38.4	265.5
Decay	41.8	**63.2**	44.0	**38.5**	**32.3**	13.4	**44.0**	**277.2**

C.3 Ablation Study

For the distillation coefficient λ in Eq. 6, we adopt a decaying schedule, as described in Appendix A. Accordingly, we conduct an ablation study on Qwen3-4B-Instruct to verify whether this decaying

strategy is superior to using a fixed λ across three iterations. As shown in Table 11, compared with a constant λ, the decaying schedule not only maintains a higher evolution speed in the early stages, but also converges to better final performance in the later stages of training.

Moreover, we perform an ablation study on the top-K parameter in Eq. 11. The study is conducted on Qwen3-4B-Instruct, with K selected from $25, 50, 100, 200$. As shown in Table 12, the results reveal that larger values of K do not necessarily yield better performance. Instead, the optimal performance is achieved when K is around 50.

Table 12: **Ablation study of K. Bold** value indicates the top-performing result

Top K	General QA			Multi-Hop QA				
	NQ	**TriviaQA**	**PopQA**	**HotpotQA**	**2WikiMQA**	**MuSiQue**	**Bamboogle**	**Total**
	Iter 1							
200	37.8	61.7	42.1	**37.1**	29.0	12.9	35.2	255.8
100	39.4	61.8	42.4	36.8	28.4	12.6	38.4	259.8
50	**40.7**	**62.1**	**42.9**	36.7	**30.1**	**13.2**	**39.2**	**264.9**
25	40.0	61.2	41.0	35.1	29.8	11.5	36.8	255.4
	Iter 2							
200	41.4	61.7	43.2	35.5	29.8	11.4	36.0	259.0
100	**41.9**	62.2	**44.4**	36.8	28.5	14.0	36.8	264.6
50	41.2	**62.5**	43.5	**37.8**	**32.5**	**14.2**	40.0	**271.7**
25	40.5	61.9	43.0	36.9	29.9	12.9	**43.2**	268.3
	Iter 3							
200	42.3	62.1	44.6	37.3	30.3	13.2	39.2	269.0
100	41.9	62.4	**44.8**	37.6	31.2	**13.8**	36.8	268.5
50	41.8	**63.2**	44.0	**38.5**	**32.3**	13.4	**44.0**	**277.2**
25	**43.1**	62.7	42.9	36.7	32.1	12.8	36.8	267.1

C.4 Training Cost

Table 13 reports the average per-step training time of different methods. Across various search agents, we observe that although our approach introduces an additional teacher model compared with Dr.Zero, it does not significantly increase the per-step training time. While the teacher model performs an extra forward pass along the student's rollout to compute token probabilities, π-Play introduces almost no additional training time compared with Dr.Zero. We attribute this to the fact that Dr.Zero lacks fine-grained feedback to penalize ineffective tool-use behaviors, causing the student model to generate redundant search queries (Fig. 5), thereby increasing training time. This effect occurs both during the computation of difficulty rewards in the examiner and during the student model's rollout.

The reported training times were measured on a node equipped with 8 NVIDIA H20 GPUs, and we follow the same training configurations described in Appendix A.

Table 13: **The training time.** We analyze the per-step training time (in seconds) for each iteration under Qwen3-4B-Instruct. Although the teacher model requires an additional forward pass along the student's rollout to compute token probabilities, π-Play introduces only a small per-step training overhead of 4.3% relative to Dr.Zero.

	Iter1		**Iter2**		**Iter3**		**Average**	
	Dr.Zero	π-Play	Dr.Zero	π-Play	Dr.Zero	π-Play	Dr.Zero	π-Play
Examiner	311.6	311.6	407.8	526.3	623.4	529.9	447.6	455.9
Student (Rollout)	88.9	81.1	146.0	159.9	176.8	149.8	137.2	130.3
Teacher (Forward)	-	21.8	-	31.8	-	30.6	-	28.1
Student (Updating)	94.0	94.1	116.2	123.0	123.2	117.8	111.1	111.6
Total Time	494.5	508.6	670.0	841.0	923.4	828.1	695.9	725.9

C.5 CASE STUDY

We present QCP examples generated by the examiner during the π-Play training process. The QCPs for different hop settings are presented in Fig. 6, Fig. 7, and Fig. 8. All cases are obtained from the model trained on Qwen3-4B-Instruct. In particular, we further compare the responses of the student model generated by the baseline Dr.Zero and our π-Play method on the same question, as shown in Fig. 5. Each response contains multi-turn interactions between the model and the external search engine.



Figure 5: Side-by-side trajectories of Dr.Zero (left) and π-Play (right) on the same question. Each trajectory shows multi-turn interactions with the search engine (actions, responses, and final answer). Although both Dr.Zero and π-Play answer the query correctly, our method (right) uses fewer queries and reaches a logically structured answer with minimal redundancy.

Random Document

```
Now, generate a question and its answer with n = 1 hops starting
from the following source document: (Title: "
FC Barcelona Rugby League")\nwidely speculated, but purely a
hypothesis, that this branch of the successful FC Barcelona club
will become members of the Rugby Football League and possibly Super
 League should the venture prove a success in Catalonia, as they
were now playing more competitive rugby, on a more regular basis.
Their first ever game was against CR Sant Cugat on 13 May 2010, a
game which ended in a disappointing 30-20 loss. They earned their
first win of the competition with a 44-30 win at home to GEiEG to
give them a hope of qualifying for the finals. And, despite GEiEG
winning ...
```

Examiner: Generate Task

```
<question>Which club played its first ever rugby game against CR
Sant Cugat on 13 May 2010?</question>
<answer>FC Barcelona Rugby League</answer>
```

Figure 6: **QCP example with hop = 1 provided by the examiner.**

Random Document

```
Now, generate a question and its answer with n = 2 hops starting
from the following source document: (Title: "Luna 19") Luna 19 (a.k
.a. Lunik 19) (E-8-LS series), was an unmanned space mission of the
 Luna program. "Luna 19" extended the systematic study of lunar
gravitational fields and location of mascons (mass concentrations).
 It also studied the lunar radiation environment, the gamma-active
lunar surface, and the solar wind. Photographic coverage via a
television system was also obtained. "Luna 19" was the first of the
 "advanced" lunar orbiters whose design was based upon the same Ye
-8-class bus used for the lunar rovers and the sample collectors.
For these orbiters, designated Ye-8LS, the basic "lander stage" was
 topped off by a ...
```

Examiner: Query 1

```
<tool_call>{"name": "search", "arguments": {"query_list": ["Luna 19
 mission spacecraft class design"]}}</tool_call>
```

Tool Response

```
(Title: "Luna 19") ... Luna 19 was the first of the "advanced"
lunar orbiters whose design was based upon the same Ye-8-class bus
used for the lunar rovers and the sample collectors ...
```

Examiner: Query 2

```
<tool_call>{"name": "search", "arguments": {"query_list": ["Ye-8LS
spacecraft used for lunar rovers and sample collectors"]}}</
tool_call>
```

Tool Response

```
(Title: "Luna 21") ... Luna 21 (Ye-8 series) was an unmanned space
mission, and its spacecraft, of the Luna program, also called Lunik
 21, in 1973. The spacecraft landed on the Moon and deployed the
second Soviet lunar rover, "Lunokhod 2". ...
```

Examiner: Generate Task

```
<question>Which country operated the lunar rover Lunokhod 2, which
was deployed by the Ye-8 series spacecraft used in the Luna 19
mission</question>
<answer>Soviet Union</answer>
```

Figure 7: **QCP example with hop = 2 provided by the examiner**

Random Document

```
Now, generate a question and its answer with n = 3 hops starting
from the following source document: (Title: "Chris Charsley")\
nseason, including the test match against Darwen which won them
promotion to the First Division. He also played for Aston Villa as
a guest in 1886. Charsley had a brief spell with West Bromwich
Albion, whom he joined in August 1891. He made his only appearance
```

for Albion in a 2-1 defeat to Preston North End in November of the same year, before re-joining Small Heath the following month. On 25 February 1893, he became the first Birmingham City player-and the only player who had played for the club in its Small Heath Alliance days-to represent ...

Examiner: Query 1

<tool_call>{"name": "search", "arguments": {"query_list": ["Chris Charsley Birmingham City first player Small Heath Alliance"]}}</tool_call>

Tool Response

(Title: "History of Birmingham City F.C. (1875-1965)") ... Goalkeeper Chris Charsley, a policeman who played as an amateur and went on to serve as chief constable of Coventry, became the first Small Heath player to be capped by England. ...

Examiner: Query 2

<tool_call>{"name": "search", "arguments": {"query_list": ["Chris Charsley England international debut date"]}}</tool_call>

Tool Response

(Title: "1892-93 Small Heath F.C. season") ... Chris Charsley became the first Small Heath player to play for England, and only the second, after Caesar Jenkyns for Wales, to represent his country at full international level, when he took the field against Ireland on 27 February at Aston Villa\'s ground in Perry Barr, Birmingham. ...

Examiner: Query 3

<tool_call>{"name": "search", "arguments": {"query_list": ["Chris Charsley England international match opponent 27 February 1893"]}}</tool_call>

Tool Response

(Title: "1892-93 in English football") ... Chris Charsley of Small Heath, who later went on to be Chief Constable of Coventry, made his solitary England appearance in goal. ...

Examiner: Generate Task

<question>Which team did Chris Charsley play against in his only England international match?</question>
<answer>Ireland</answer>

Figure 8: **QCP example with hop = 3 provided by the examiner.**

D DISTILLATION LOSS

To save GPU memory, we adopt top-K distillation following (Hübotter et al., 2026), where the top-K set is defined with respect to the student distribution:

$$
\begin{aligned}
&\mathbb{D}_{\text{Distill}}\big(\pi_\theta^S \,\|\, \pi_\psi^T\big) = \frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \mathrm{KL}(\pi_\theta^S(\cdot \mid q, y_{i,<t}) \| \mathrm{stopgrad}(\pi_\psi^T(\cdot \mid q, c, y_{i,<t})))\\
&= \frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \sum_{\hat{y}_{i,t}\in\pi_\theta^S} \pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t}) \cdot \log \frac{\pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t})}{\mathrm{stopgrad}(\pi_\psi^T(\hat{y}_{i,t} \mid q, c, y_{i,<t}))}\\
&= \left(\begin{array}{l}
\dfrac{1}{|y_i|}\displaystyle\sum_{t=1}^{|y_i|} \sum_{\hat{y}_{i,t}\in\mathrm{top}_K(\pi_\theta^S)} \pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t}) \cdot \log \frac{\pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t})}{\mathrm{stopgrad}(\pi_\psi^T(\hat{y}_{i,t} \mid q, c, y_{i,<t}))}\\
\underbrace{\dfrac{1}{|y_i|}\displaystyle\sum_{t=1}^{|y_i|} \sum_{\hat{y}_{i,t}\in\pi_\theta^S\setminus\mathrm{top}_K(\pi_\theta^S)} \pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t}) \cdot \log \frac{\pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t})}{\mathrm{stopgrad}(\pi_\psi^T(\hat{y}_{i,t} \mid q, c, y_{i,<t}))}}_{\text{tail}}
\end{array}\right)\\
&\approx \left(\begin{array}{l}
\dfrac{1}{|y_i|}\displaystyle\sum_{t=1}^{|y_i|} \sum_{\hat{y}_{i,t}\in\mathrm{top}_K(\pi_\theta^S)} \pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t}) \cdot \log \frac{\pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t})}{\mathrm{stopgrad}(\pi_\psi^T(\hat{y}_{i,t} \mid q, c, y_{i,<t}))}\\
\underbrace{\Big(1-\sum_{\hat{y}_t\in\mathrm{top}_K(\pi_\theta^S)} \pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t})\Big) \cdot \log \dfrac{1-\sum_{\hat{y}_t\in\mathrm{top}_K(\pi_\theta^S)} \pi_\theta^S(\hat{y}_{i,t} \mid q, y_{i,<t})}{\mathrm{stopgrad}\Big(1-\sum_{\hat{y}_t\in\mathrm{top}_K(\pi_\theta^S)} \pi_\psi^T(\hat{y}_{i,t} \mid q, c, y_{i,<t})\Big)}}_{\text{tail}}
\end{array}\right)
\end{aligned}
\tag{11}
$$

Rather than computing the full KL divergence over the entire vocabulary, we split the distillation loss into two components: the exact contribution from the student's top-K tokens and a tail term corresponding to all remaining tokens. The tail is further approximated by collapsing the non-top-K probability mass into a single residual term. This strategy avoids storing two full copies of vocabulary logits: one for the student and one for the teacher, thereby greatly reducing memory usage. Empirically, this approximation has negligible impact on performance, since most tokens of the vocabulary are not informative at a given time (Hübotter et al., 2026). Further analysis of the choice of K can be found in Appendix C.

E PROMPTS

We provide the system prompts for all models in Section E.1, and the user prompts for the examiner, teacher, and student models in Sections E.2, E.3, and E.4, respectively.

E.1 SYSTEM PROMPTS

System Prompt for the Examiner, Teacher and Student

```
You are a helpful assistant.

# Tools

You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML tags:
<tools>
{"type": "function", "function": {"name": "search", "description": "Searches the web
for relevant information based on the given query.", "parameters": {"type": "object", "
properties": {"query_list": {"type": "array", "description": "A list of fully-formed
semantic queries. The tool will return search results for each query.", "enum": null}},
 "required": ["query_list"]}, "strict": false}}
</tools>

For each function call, return a json object with function name and arguments within <
tool_call></tool_call> XML tags:
```

```
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

Figure 9: **System prompt for the examiner, teacher and student in π-play.** They use the same system prompt.

E.2 EXAMINER PROMPTS

User Prompt for the Examiner

```
You are an expert in question generation. Craft one challenging, deterministic question
 and its single, unambiguous answer based on the provided source document. The logical
path must start from the document and require exactly n hops (i.e., n-1 searches) to
reach the final answer.

### Definitions
1. Hop: A node in the reasoning chain. Hop 1 is the starting entity found in the
document. Hop n is the final answer.

### Inputs
1. n: the exact number of hops in the reasoning chain (requiring n-1 searches).
2. Source document: the full source text.

### Process & Tools
1. Analyze the Document and Select the Starting Point
  - Read and analyze the source document.
  - Select a specific entity, event or detail explicitly mentioned in the text. This
entity becomes Hop 1 (the initial clue).
2. Design the Chain Forwards
  - From Hop 1 to Hop 2: Identify a factual attribute or relation of Hop 1 that is NOT
in the text but can be found via search. The result is Hop 2.
  - Iterate: Continue connecting the current Hop i to the next Hop i+1 using
deterministic, verifiable relation found via search.
  - Stop at Hop n: Continue this process until you have exactly n hops. Hop n must be a
 single, canonical final answer.
3. Reasoning & Search Protocol
  - Always reason inside `<think> ... </think>` when you plan connections or receive
new information.
  - For each hop transition that requires external information, issue search query
using `<tool_call> ... </tool_call>`.
  - Search results will be provided between `<tool_response> ... </tool_response>` by
the system.
4. Output Format
  - Emit a numbered sequence of EXACTLY n-1 search steps. For each search i (1 to n-1),
 produce:
    `<think> Reasoning step i: Identify Hop i in document/search results, formulate
query to reach Hop i+1 </think>`
    `<tool_call> Query to search Hop i+1 </tool_call>`
    `[Wait for search results in <tool_response> from system]`
  - After completing all searches and arriving at Hop n, output the question and final
answer:
    `<think> Final reasoning step: Confirm the chain is complete with Hop n and
formulate the question </think>`
    `<question> A challenging question that provides Hop 1 (the initial clue) and asks
for the final answer (Hop n) </question>`
    `<answer> The single, concise final answer (Hop n) </answer>`

### Examples
1. Example template for Hop n = 1, i.e. no search:
  `<think> [Explain how Hop 1 is selected from the source document and how the question
 is formulated] </think>`
  `<question> [Question based solely on the text entity Hop 1] </question>`
  `<answer> [Answer (Hop 1)] </answer>`
2. Example template for Hop n = 3, i.e. 2 searches:
  `<think> [Reasoning step 1: Find Hop 1 in the source document, formulate the query to
 reach Hop 2] </think>`
  `<tool_call> [Search query to find Hop 2 based on Hop 1] </tool_call>`
  `[Wait for search results in <tool_response> from system]`
  `<think> [Reasoning step 2: Reason on search results to identify Hop 2 and write the
next query to find Hop 3] </think>`
  `<tool_call> [Search query to find Hop 3 based on Hop 2] </tool_call>`
  `[Wait for search results in <tool_response> from system]`
  `<think> [Final reasoning step: Confirm Hop 3 in search results and formulate the
question starting from Hop 1] </think>`
  `<question> [Question starting with Hop 1, requiring the solver to find Hop 2 to
eventually reach the Answer (Hop 3)] </question>`
```

```
  `<answer> [Answer (Hop 3)] </answer>`

### Critical Rules
1. Start in Document: Hop 1 must be explicitly present in the source text. Every
subsequent hop must be supported by the corresponding search results.
2. Search is mandatory for n > 1: Each link between hops beyond Hop 1 must use the
search engine.
3. Exact search count: Emit exactly (n-1) `<tool_call>` entries, no more, no fewer.
4. No spoilers: The question must mention only Hop 1; do not include or hint at
intermediate hops.
5. Clarity: The question is self-contained; the answer is concise and direct (no extra
commentary, formatting or explanation).
6. Chain integrity: Each hop must depend strictly on the previous hop. No hop should be
 skippable or derivable without its immediate predecessor.

Now, generate a question and its answer with n = {hops} hops starting from the
following source document: {document}
```

Figure 10: **Initial instructions for the examiner in π-play.** Our prompt for examiner is developed based on Dr.Zero (Yue et al., 2026b)

E.3 Teacher Prompts

User Prompt for the Teacher (Qwen3-4B-Instruct)

```
You are a helpful assistant. You will be given privileged information about the reverse
 solution process of the question (i.e., construction process of the question). Please
answer the question again with reference to the construction process of the question.

###  General Rules
1. Precise Searching: When generating search queries, ensure they are specific,
semantically complete, and directly target the key information you are missing.
2. Context Retention: Remember prior conversations and search results, maintaining
logical consistency across multiple rounds of searching.
3. Termination Judgment: When information are sufficient to determine the answer,
immediately stop searching and output the answer.
4. Reference to Privileged Information: When outputting search actions and answers,
refer to the question construction process in privileged information, but do not
directly output the ground-truth in the first turn or directly use extra information
from the source document.
5. Search Boundary Constraint: When searching, do not introduce information that is not
 contained in the question or search results, even if it exists in the source document.

### Notice
Most importantly, pretending you do not know the ground-truth answer, the extra
information in the source document, and do not mention the source document when
thinking.

###Begin
This is the reverse solution process  of the question (i.e., the process of generating
the question) given by teacher:

================================================================================
{demonstration}
================================================================================

Now, answer this question again by yourself, pretending you do not know the ground-
truth answer, the source document. You need to start answering this question from
scratch. Most importantly, pretending you do not know the ground-truth answer, the
source document, and do not mention the source document when thinking. If you find you
lack some knowledge, you can call a search engine by <tool_call> query </tool_call> and
 it will return the top searched results between <tool_response> and </tool_response>.
You can search as many times as your want. If you find no further external knowledge
needed, you can directly provide the answer inside <answer> and </answer>, without
detailed illustrations. For example, <answer> Beijing </answer>. Question: {question}
```

User Prompt for the Teacher (Qwen3-4B / Qwen3-8B)

```
You are a helpful assistant. You will be given privileged information about the reverse
 solution process of the question (i.e., construction process of the question). Please
pretend not to know the source document used to construct the question and the ground-
truth in the reverse solution process, and answer the question again with reference to
the construction process of the question.

###  General Rules
Precise Searching: When generating search queries, ensure they are specific,
semantically complete, and directly target the key information you are missing.
```

```
Context Retention: Remember prior conversations and search results, maintaining logical
 consistency across multiple rounds of searching.
Termination Judgment: When information are sufficient to determine the answer,
immediately stop searching and output the answer.
Reference to Privileged Information: When outputting search actions and answers, refer
to the question construction process in privileged information, but do not directly
output the ground-truth in the first turn or directly use extra information from the
source document.
Search Boundary Constraint: When searching, do not introduce information that is not
contained in the question or search results, even if it exists in the source document.

### Notice
Most importantly, pretending you do not know the ground-truth answer, the extra
information in the source document, and do not mention the source document when
thinking.

###Begin
This is the reverse solution process  of the question (i.e., the process of generating
the question) given by teacher:

================================================================================
{demonstration}
================================================================================

Now, answer this question again by yourself, pretending you do not know the ground-
truth answer, the source document. You need to start answering this question from
scratch. Most importantly, pretending you do not know the ground-truth answer, the
source document, and do not mention the source document when thinking. You must conduct
 reasoning inside <think> and </think> first every time you get new information. After
reasoning, if you find you lack some knowledge, you can call a search engine by <
tool_call> query </tool_call> and it will return the top searched results between <
tool_response> and </tool_response>. You can search as many times as your want. If you
find no further external knowledge needed, you can directly provide the answer inside <
answer> and </answer>, without detailed illustrations. For example, <answer> Beijing </
answer>. Question: {question}
```

Figure 11: **Initial instructions for the teacher in π-play**

E.4 STUDENT PROMPTS

User Prompt for the Student (Qwen3-4B-Instruct)

```
Answer the given question. If you find you lack some knowledge, you can call a search
engine by <tool_call> query </tool_call> and it will return the top searched results
between <tool_response> and </tool_response>. You can search as many times as your want
. If you find no further external knowledge needed, you can directly provide the answer
 inside <answer> and </answer>, without detailed illustrations. For example, <answer>
Beijing </answer>. Question: {question}
```

User Prompt for the Student (Qwen3-4B / Qwen3-8B)

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

Figure 12: **Initial instructions for the student in π-play**