Ada-RS: Adaptive Rejection Sampling for Selective Thinking

Yirou Ge[*], **Yixi Li**[*], **Alec Chiu**[†], **Shivani Shekhar**[†],
Zijie Pan[†], **Avinash Thangali**[†], **Yun-Shiuan Chuang**[†], **Chaitanya Kulkarni**[†],
Uma Kona, **Linsey Pang**, **Prakhar Mehrotra**
PayPal AI

Abstract

Large language models (LLMs) are increasingly being deployed in cost- and latency-sensitive settings. While chain-of-thought improves reasoning, it can waste tokens on simple requests. We study selective thinking for tool-using LLMs and introduce Adaptive Rejection Sampling (Ada-RS), an algorithm-agnostic sample filtering framework for learning selective and efficient reasoning. For each given context, Ada-RS scores multiple sampled completions with an adaptive length-penalized reward then applies stochastic rejection sampling to retain only high-reward candidates (or preference pairs) for downstream optimization. We demonstrate how Ada-RS plugs into both preference pair (*e.g.* DPO) or grouped policy optimization strategies (*e.g.* DAPO). Using Qwen3-8B with LoRA on a synthetic tool call-oriented e-commerce benchmark, Ada-RS improves the accuracy–efficiency frontier over standard algorithms by reducing average output tokens by up to $\sim$80% and reducing thinking rate by up to $\sim$95% while maintaining or improving tool call accuracy. We further demonstrate that these gains generalize across model scales (Qwen3-1.7B, 8B, 14B) and domains (τ^2-Bench airline and telecom). These results highlight that training-signal selection is a powerful lever for efficient reasoning in latency-sensitive deployments.

1 Introduction

Large language models (LLMs) are increasingly deployed inside cost- and latency-sensitive systems that facilitate human interactions such as customer service assistants and e-commerce copilots that must respond within tight service-level agreements while handling a large volume of queries. To navigate complex requests, LLMs often rely on explicit chain-of-thought (CoT) (Wei et al., 2022) style reasoning to ensure high quality; however, generating long reasoning traces can often introduce substantial overhead and degrade the user experience, especially when many requests are routine or can be easily handled with short responses (*e.g.* small talk, quick clarifications). As a result, a key practical question in real deployments is not whether models can reason and break down complex tasks, but how to allocate reasoning budget only when it aids in resolving a user's request.

Recent *selective thinking* work has begun to tackle this matter by training or prompting models to modulate reasoning depth through strategies such as learning when to invoke CoT (Lou et al., 2025), pruning or compressing reasoning (Xiang et al., 2025; Fang et al., 2025; Hou et al., 2025), or balancing accuracy against token usage via reward shaping (Yang et al., 2025) and multi-objective optimization (Xiang et al., 2026). These approaches make important progress, but they often rely on boundary tuning or specialized training objectives.

In this work, we study selective thinking through the lens of model training. However, our key design choice is to target a different and complementary lever from existing work: the candidate samples used for learning. We introduce Adaptive Rejection Sampling (Ada-RS), an algorithm-agnostic sampling mechanism that applies rejection sampling over candidates, probabilistically retaining responses that are most informative for learning selective and efficient thinking while downsampling uninformative or unnecessarily verbose samples. Due to its algorithm-agnostic nature, Ada-RS can complement several tuning algorithms such as direct preference optimization (DPO) (Rafailov et al., 2024) to construct higher-quality preference pairs for preference optimization or group relative policy optimization-style methods (*i.e.,* GRPO-style updates) (Shao et al., 2024; Yu et al., 2025).

We apply our method to a tool call oriented e-commerce domain setting and find our approach yields favorable accuracy–efficiency trade-offs,

[*]Joint first author.
[†]Joint second author.

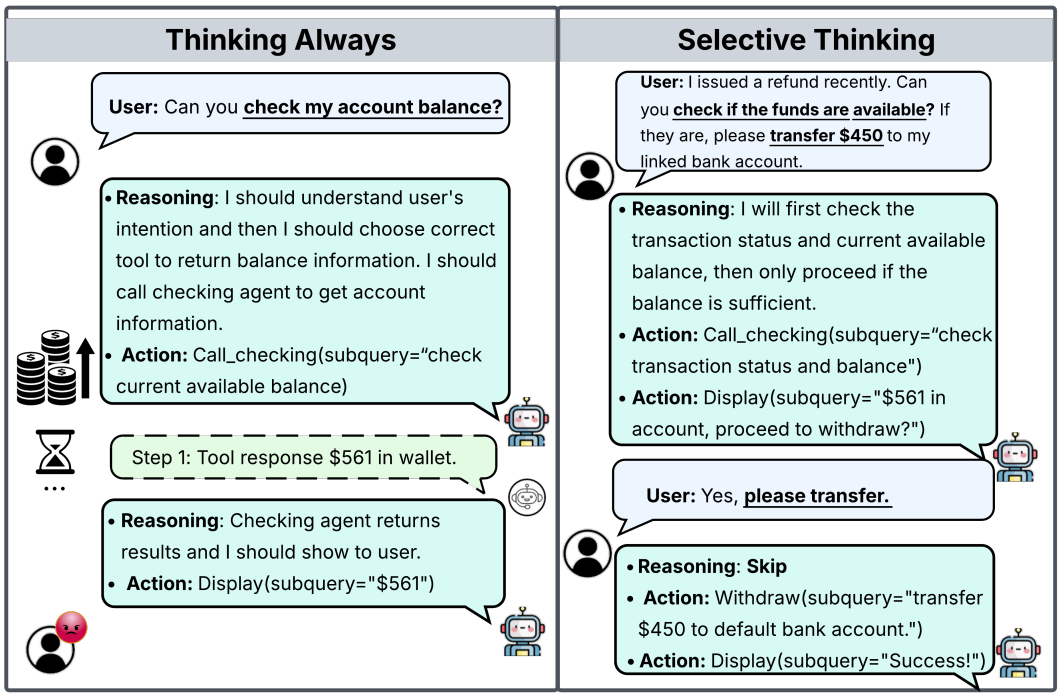

Figure 1: **Comparison between thinking always reasoning and selective reasoning in a tool-calling LLM agent.** The agent performs explicit reasoning even for a simple user query, resulting in unnecessary inference cost and latency (left). The agent selectively skips reasoning and directly calls the appropriate tool (right).

substantially reducing token usage by 70%-80% without sacrificing performance. These results highlight the importance of how we construct and filter the training signal and how that signal can be leveraged to improve the efficiency of generation, especially for systems where latency and inference cost are first-order product constraints.

Contributions.

1. We formalize and evaluate *selective thinking* for latency-sensitive systems, emphasizing the practical need to decide *when not to think*.

2. We propose Ada-RS, an algorithm-agnostic rejection-sampling mechanism for training sample candidates that improves sample efficiency and discourages unnecessary verbosity while preserving long reasoning when useful.

3. We empirically study different optimization strategies and show that Ada-RS enables stronger accuracy–efficiency trade-offs on tool call oriented tasks, with gains that generalize across domains (e-commerce, airline, telecom) and model scales (1.7B to 14B parameters).

2 Related Work

CoT prompting can substantially improve reasoning (DeepSeek-AI, 2025; OpenAI, 2024), but in production settings it often incurs unnecessary latency and cost when long explanations are generated for otherwise simple inputs. Consequently, recent work on efficient reasoning has focused on controlling *when* a model should engage in explicit reasoning (selective thinking) (Lou et al., 2025; Zhang et al., 2025) and *how much* reasoning it should produce, via either (i) explicit, prompt-level control (Ma et al., 2025) or (ii) training-time objectives that encourage selective thinking (Sui et al., 2025).

2.1 Explicitly Controlled Reasoning

A straightforward way to reduce overthinking is to *explicitly* control reasoning behavior at inference time, for example with special prompt instructions, formatting constraints, or difficulty-aware prompting (Ma et al., 2025; Tu et al., 2025). While effective and easy to deploy, such approaches typically rely on external prompt control and do not necessarily teach the model to internally learn when reasoning is warranted; performance can also be sensitive to prompt phrasing and may not generalize across domains (Sui et al., 2025).

2.2 Selective Thinking Through Training

A complementary line of work aims to *learn* selective thinking through training, typically by modifying rewards (Yang et al., 2025; Lou et al., 2025) or objectives (Zhang et al., 2025; Xiang et al., 2026), to trade off task success against reasoning cost. Compared to prompt-only control, these training-based methods can yield a single model that better internalizes the decision of when to reason; however, they often require careful tuning of penalty strengths or multi-stage training to avoid degenerate solutions (*e.g.* collapsing to always-think or never-think behavior) (Sui et al., 2025).

2.3 Relation to Our Work

Our work targets the same goal of selective thinking but focuses on **training-signal construction and selection**. While prior approaches emphasize reward design or alternative training objectives, we propose *Adaptive Rejection Sampling* (Ada-RS) to stochastically retain the most informative samples (or preference pairs) under an adaptive length penalty. This design aims to reduce the influence of unnecessarily verbose trajectories during training while preserving explicit reasoning on difficult inputs.

3 Preliminaries

3.1 Task Overview

We consider a *tool calling* LLM agent that resolves a user request by optionally invoking e-commerce-related tools (*e.g.*, product search, account information retrieval, transaction look-ups) and then producing a final response. An example of a task in the e-commerce setting we explore is given in Figure 1.

At any decision point, the agent observes a context x consisting of the conversation history and any tool outputs observed so far. Given x, the model produces a response $y = (\texttt{<think>}\ t\ \texttt{</think>}, a)$ where t is an optional reasoning trace and a is the final answer (including tool calls when applicable). The goal is to learn a policy $\pi_\theta(y \mid x)$ that emits little-to-no reasoning on easy or simple instances while still using reasoning when it materially improves correctness.

4 Methods

4.1 Adaptive Rejection Sampling (Ada-RS)

Ada-RS is a lightweight *sample selection* mechanism that can be plugged into both off-policy and on-policy training objectives that rely on sampled completions. An overview of the framework and how we apply it are illustrated in Figure 2.

For each context x, we draw K training sample candidates $\{y_i\}_{i=1}^{K} \sim \pi_\phi(\cdot \mid x)$, assign each candidate an adaptive efficiency-aware reward, and then apply stochastic rejection sampling to retain only the most informative candidates (or candidate pairs) for downstream optimization.

4.1.1 Adaptive length penalty (ALP)

Given rollouts $\{y_i\}_{i=1}^{K} \sim \pi_\theta(\cdot \mid x)$, we define a composite reward that trades off task success and reasoning cost:

$$r(y_i, x) = \mathbb{1}(y_i, x) - \alpha \cdot s_K(x) \cdot |t_i| \qquad (1)$$

Here $\mathbb{1}(y_i, x) \in \{0, 1\}$ indicates whether the rollout y_i solves the task for prompt x (*i.e.* correct tool call). $|t_i|$ penalizes the length of the reasoning trace in the rollout (we use the number of sentences inside <think> as a proxy; $|t_i| = 0$ when the <think> block is empty). The key adaptive component is

$$s_K(x) = \frac{1}{K} \sum_{i=1}^{K} \mathbb{1}(y_i, x) \qquad (2)$$

an online estimate of how easy the prompt is under the current policy. When $s_K(x)$ is high, Ada-RS applies a stronger length penalty, discouraging unnecessary reasoning on easy prompts; when it is low, the penalty shrinks, allowing longer reasoning on harder prompts. This follows the spirit of (Xiang et al., 2025), which scales a length cost by an online solve-rate estimate.

4.1.2 Rejection sampling over training sample candidates

Given rewards $\{r_i\}_{i=1}^{K}$ for context x, Ada-RS performs rejection sampling to preferentially retain higher-reward (more correct and/or more efficient) samples while keeping stochasticity for diversity. We support:

Pair-wise rejection sampling (for preference learning). For a candidate pair (i, j), define $\Delta_{ij} = r_i - r_j$ and accept the pair with probability

$$p_{ij} \;=\; \exp\!\left(\frac{\Delta_{ij} - \Delta_{\max}}{\beta_{\text{rs}}}\right), \qquad (3)$$

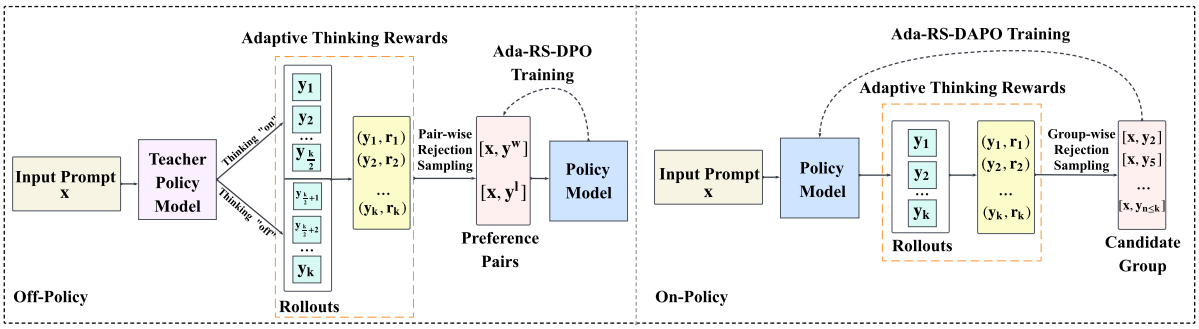

Figure 2: **Overview of the proposed Ada-RS training framework.** ***Off-policy Ada-RS-DPO*** training pipeline (left). Given an input context x, a teacher policy model generates multiple rollouts. Ada-RS performs pair-wise rejection sampling based on adaptive thinking reward signals for selective thinking to construct high-quality preference pairs, which are then used to optimize the student policy model via DPO training. ***On-policy Ada-RS-DAPO*** training pipeline (right). The current policy model generates multiple rollouts for the input context. Ada-RS applies group-wise rejection sampling based on adaptive thinking rewards to select informative candidate subsets. The resulting candidate group is used to update the policy through an on-policy DAPO training.

where β_{rs} is a temperature controlling selectivity hyperparameter and $\Delta_{\text{max}} = \max_{i<j}(\Delta_{ij})$. The accepted pair is converted into a preference example (x, y^w, y^l) by setting $y^w = \arg\max(r_i, r_j)$ and $y^l = \arg\min(r_i, r_j)$. This builds on recent successes found with rejection sampling (Liu et al., 2023) and utilizing reward gaps for preference pair optimization (Khaki et al., 2024).

Group-wise rejection sampling (for grouped policy optimization). Alternatively, we accept each candidate y_i independently based on its standardized reward within the group:

$$p_i = \min\left(\exp\left(\frac{(r_i - \mu)/\sigma}{\beta_{\text{rs}}}\right), 1\right), \quad (4)$$

where μ and σ are the mean and standard deviation of $\{r_i\}_{i=1}^K$ for the prompt. Smaller β_{rs} concentrates training on above-mean samples; larger values keep more diverse candidates.

4.1.3 Ada-RS-DPO

Ada-RS-DPO uses pair-wise rejection sampling to construct higher-quality preference pairs for DPO. For each context x, we sample K candidates of equal amounts across thinking-enabled/disabled examples, compute rewards via Eq. 1, and accept candidate pairs with Eq. 3. Using the accepted preference pairs, we then optimize a DPO objective that includes an auxiliary negative log-likelihood (NLL) loss term to stabilize learning and preserve language modeling quality as described in (Pang et al., 2024). Pseudocode for this algorithm can be found in Appendix Algorithm 1.

4.1.4 Ada-RS-DAPO

We also integrate Ada-RS with Decoupled Clip and Dynamic Sampling Policy Optimization (DAPO) (Yu et al., 2025), which we use as our grouped policy optimization backbone. For each context x, we sample a group of K on-policy candidates $\{y_i\}_{i=1}^K \sim \pi_\theta(\cdot \mid x)$, compute rewards $\{r_i\}_{i=1}^K$ using Eq. 1, and apply group-wise rejection sampling using the acceptable probability from Eq. 4 to stochastically filter candidates before the DAPO update. The DAPO objective is then computed on the retained candidates, leaving the underlying DAPO loss unchanged while concentrating gradient updates on trajectories that are both correct and efficiency-favorable. This yields a simple plug-in mechanism to bias grouped on-policy learning toward selective thinking. Pseudocode for this algorithm can be found in Appendix Algorithm 2.

5 Experiments

All experiments use Qwen3-8B as the reasoning base model (Qwen Team, 2025) with a LoRA adapter (Hu et al., 2022) for all training runs. Additional details on training algorithm hyperparameters can be found in Appendix A and Appendix B.

5.1 Evaluation Setting

We evaluate our methodology on a synthetic multi-turn, multi-step e-commerce dataset designed to mirror common user personas and tasks modeled on general themes observed on e-commerce platforms. The tools available in the dataset mirror those in the τ^2-Bench retail benchmark (Barres

et al., 2025). Our benchmark was specifically designed to reflect the interaction structure, tool schemas, and task distributions characteristic of internal production systems, enabling controlled study of selective reasoning under conditions closer to planned deployment. We additionally validate on τ^2-Bench airline and telecom domains to demonstrate cross-domain generalization (Section 6.4). An example task is shown in Figure 1. Overall, our base training dataset consists of 15,000 tool invocations across 8,026 conversations, which span across 121 tasks and 8 user personas. Our evaluation dataset consists of 2,510 tool calls from 367 conversations, which span across 48 tasks and 8 user personas.

5.2 Metrics

The key metrics we evaluate are as follows:

- **Thinking Rate**: the percentage of instances in which the model produces a non-empty reasoning trace. This metric measures how often the model chooses to engage in explicit reasoning.
- **Output Token Length**: the average number of generated output tokens produced by the model (inclusive of reasoning) as a measure of token efficiency. This serves as a deployment-relevant proxy for inference cost and latency.
- **Tool Call Accuracy**: whether the model selects the correct tool and produces the correct arguments, ensuring functional correctness when reasoning is skipped.

5.3 Baselines and Ablation Studies

We compare prompt-only baselines, supervised fine-tuning (SFT), DPO, and DAPO as well as ablations and hyperparameter sweeps for Ada-RS.

5.3.1 Prompt-only and SFT baselines

We include two *no-fine-tuning* (NFT) baselines that differ only by prompting: *NFT (Thinking-On)*, where reasoning is always invoked, and *NFT (Thinking-Off)*, where explicit reasoning never occurs (*i.e.* forced empty `<think>`). An SFT baseline is established by applying SFT on a dataset of 75% reasoning and 25% non-reasoning data.

5.3.2 DPO baselines and ablations

We evaluate DPO-based training with and without the components used in Ada-RS-DPO: ALP reward, NLL auxiliary loss, and rejection sampling. When not using the ALP reward, we utilize a simple reward strategy to construct preference pairs where correct responses are preferred over incorrect responses and subsequently more concise responses are preferred amongst the correct responses.

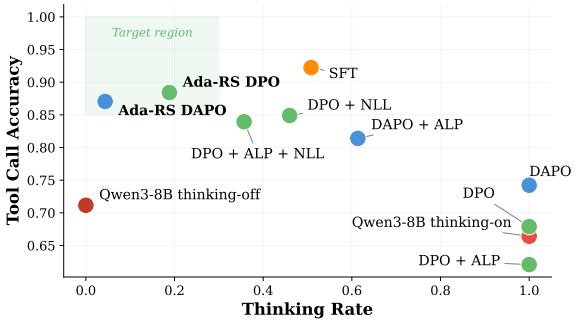

Figure 3: **Tool call accuracy versus Thinking Rate across methods.** The most favorable target region (high accuracy, low thinking rate) is highlighted. Points are colored by algorithm: DPO (green), DAPO (blue), SFT (orange), and no-fine-tuning/base model (red).

5.3.3 DAPO baselines and ablations

We also evaluate Ada-RS in a grouped policy optimization setting using DAPO as the backbone. We utilize DAPO with only accuracy as a reward function and DAPO with the ALP reward function, both without rejection sampling, as baseline ablations for grouped policy optimization.

6 Results

6.1 Learning When to Reason

We first examine whether different training strategies induce the ability to selectively think. SFT is able to successfully induce thinking to trigger only about half the time (Figure 3) but with high verbosity (Figure 4). In contrast, DPO applied directly to the base model with a simple preference pair generation strategy fails to remove the model's default *always-think* behavior at all, suggesting that preference optimization alone does not reliably teach the model *when* to deliberate (Figure 3).

Ada-RS changes this behavior by filtering training sample candidates using an efficiency-aware reward. Ada-RS drastically reduces how often explicit reasoning is invoked overall (Figure 3), while still allocating concise reasoning budgets to maintain accuracy (Figure 4).

6.2 Accuracy-Efficiency Trade-offs

We can first focus on the two best extremes in terms of accuracy and generated output tokens.

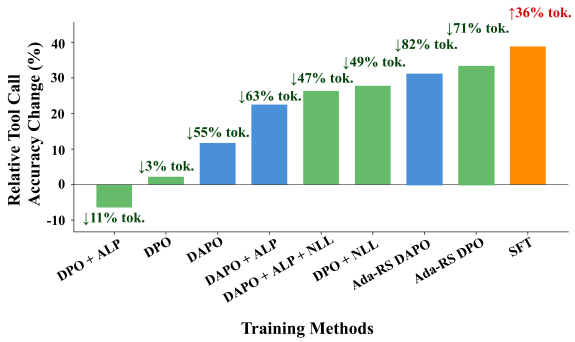

Figure 4: **Tool call accuracy and average Output token across methods relative to the Qwen3-8B base model with thinking enabled.** Numbers above the bars show percentage decrease in average amount of output tokens relative to the base model. Bars are colored by algorithm: DPO (green), DAPO (blue), and SFT (orange).

While SFT attains the highest overall tool call accuracy, it does so with the largest average output length (Figure 4), reflecting high inference cost in latency-sensitive deployments. Meanwhile, the extreme of *NFT Thinking-Off* produces short outputs and never triggers reasoning but yields substantially lower tool call accuracy (Figure 4), indicating that simply suppressing reasoning at inference time is insufficient for reliability. Applying the Ada-RS framework substantially improves the accuracy-efficiency frontier. Ada-RS-DPO achieves a markedly better operating point: attaining similar tool call accuracy to SFT but with low output tokens and a very low thinking rate (Figure 4). Ada-RS-DAPO further improves the frontier: reaching similar accuracy and token cost (Figure 4) as Ada-RS-DPO while reducing the thinking rate even further (Figure 3). Together, these results suggest Ada-RS is complementary to both pairwise preference optimization and grouped policy optimization.

6.3 Ablation Studies

To further understand the effect of different components in Ada-RS, we ablate the specific components that drive selective thinking and efficiency. We further provide the hyperparameter analysis in Appendix B.

Rejection sampling without stabilization fails to learn selective thinking. Applying rejection sampling on top of the ALP reward without the auxiliary NLL stabilization term leads to degenerate behavior (poor accuracy and an always-think policy) (Table 1). This highlights that naive filtering alone can destabilize learning and that stabilization is important for maintaining correctness while optimizing efficiency.

Table 1: **Ablation study results for the effect of using NLL loss with DPO loss.** For both experiments, $\beta_{rs} = 0.1$ and $\alpha = 0.01$. RS denotes the use of the rejection sampling procedure used in Ada-RS for preference pairs.

Method	Accuracy (%)	Avg Output Tokens	Thinking Rate (%)
DPO ALP + RS	63.82	451.64	100.00
Ada-RS-DPO	89.24	87.81	6.10

NLL induces selectivity; ALP improves efficiency; Ada-RS amplifies. DPO alone fails to induce selective thinking (Figure 3). Adding the NLL term yields large improvements in selective thinking and accuracy (Figure 3), while the ALP reward improves token efficiency by discouraging verbosity (especially on prompts that have a high solve rate under the current policy) (Figure 4). The addition of rejection sampling from Ada-RS then further amplifies this effect beyond what is obtained by the reward and auxiliary objectives by concentrating updates on high-reward (correct and efficient) trajectories (Figure 3, 4).

6.4 Cross-Domain Generalization

To assess whether Ada-RS generalizes beyond our e-commerce setting, we apply the full pipeline to the τ^2-Bench (Barres et al., 2025) telecom and airline domains, which differ substantially in tool schemas, task types, and conversation structure.

Table 2: **Generalization to τ^2-Bench domains.** Ada-RS delivers consistent accuracy and efficiency gains across domains unseen during development.

Domain	Model	Acc. (%)	Avg Output Tokens	Thinking Rate (%)
Telecom	NFT-8B	50.88	1091.62	100.00
	Ada-RS-DAPO	60.99	118.41	34.64
	Ada-RS-DPO	64.66	127.35	25.27
Airline	NFT-8B	64.10	1169.48	100.00
	Ada-RS-DAPO	66.67	98.47	9.19
	Ada-RS-DPO	72.44	107.00	10.26

In the telecom domain, Ada-RS improves accuracy by over 10% compared to the base model while reducing output tokens by approximately 89% and lowering thinking rate to 25–35%. In the airline domain, Ada-RS-DPO improves accuracy by ∼8%, reduces output tokens by ∼91%, and lowers thinking rate to ∼10%. These trends are con-

sistent with our original e-commerce findings and demonstrate that Ada-RS is not domain-specific.

6.5 Model Scale Generalization

We further evaluate Ada-RS on both a smaller (Qwen3-1.7B) and larger (Qwen3-14B) model to assess whether the method is tied to a specific model capacity.

Table 3: **Scaling across model sizes.** Ada-RS preserves accuracy–efficiency trade-offs from 1.7B to 14B parameters on our e-commerce benchmark.

Size	Model	Acc. (%)	Avg Output Tokens	Thinking Rate (%)
1.7B	NFT	76.73	311.44	100.00
	Ada-RS-DPO	84.50	130.14	25.50
	Ada-RS-DAPO	87.37	83.99	10.20
8B	NFT	66.41	450.63	100.00
	Ada-RS-DPO	88.41	130.79	18.84
	Ada-RS-DAPO	87.05	81.90	4.34
14B	NFT	83.63	322.01	100.00
	Ada-RS-DPO	87.65	128.73	15.98
	Ada-RS-DAPO	87.53	131.60	67.21

Across scales, Ada-RS consistently improves both accuracy and token efficiency relative to the base model. At 1.7B, Ada-RS-DAPO achieves a $\sim$11% accuracy gain while reducing output tokens by 73% and thinking rate to 10%. At 14B, Ada-RS-DPO improves accuracy by $\sim$4% with a 60% reduction in output tokens. We note that Ada-RS-DAPO at 14B exhibits a higher thinking rate (67%), suggesting that hyperparameter sensitivity may increase at larger scales and warrants further investigation.

7 Limitations and Future Work

Our study has several limitations that point to promising directions for future work. First, while we demonstrate generalization across three domains (τ^2-Bench telecom, airline, and our e-commerce setting) and three model scales (1.7B, 8B, 14B), all evaluations are conducted in simulated environments. The gap between synthetic benchmarks and live production traffic, where query distributions shift, user behavior is less predictable, and system context evolves, remains unverified. We are actively exploring opportunities to evaluate Ada-RS in deployed settings. Second, our evaluation emphasizes per-step tool call accuracy, which may not fully capture end-to-end task success in multi-turn settings (e.g. whether a user goal is ultimately satisfied). Future work should include goal-completion metrics and other user-facing outcomes. Third, the elevated thinking rate observed with Ada-RS-DAPO at 14B scale (Table 3) suggests that the interplay between rejection sampling temperature and model capacity may require scale-aware tuning, which we leave to future work.

8 Conclusions

We studied selective thinking for latency- and cost-sensitive LLM deployments, where the practical objective is not simply to reason well, but to allocate explicit reasoning only when it materially improves tool behavior. To this end, we introduced **Adaptive Rejection Sampling (Ada-RS)**, an algorithm-agnostic mechanism that filters training sample candidates using an adaptive efficiency-aware reward, downweighting unnecessarily verbose trajectories while preserving reasoning on harder inputs. Across tuning backbones, Ada-RS consistently improved the accuracy-efficiency frontier: it reduced overall output length and the frequency of explicit reasoning while maintaining strong tool call accuracy. These gains hold across domains (e-commerce, airline, telecom) and model scales (1.7B to 14B parameters). Overall, these findings highlight that *how* we construct and filter training signal can be a first-order lever for deploying reasoning-capable models under strict product constraints, and that selective thinking can be induced without relying on inference-time gating or prompt switches.

References

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. τ^2-bench: Evaluating conversational agents in a dual-control environment. *Preprint*, arXiv:2506.07982.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Thinkless: Llm learns when to think. *Preprint*, arXiv:2505.13379.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *Preprint*, arXiv:2504.01296.

Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Saeed Khaki, JinJin Li, Lan Ma, Liu Yang, and Prathap Ramachandra. 2024. Rs-dpo: A hybrid rejection sampling and direct preference optimization method for alignment of large language models. *Preprint*, arXiv:2402.10038.

Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, and Jialu Liu. 2023. Statistical rejection sampling improves preference optimization. *arXiv preprint arXiv:2309.06657*.

Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. 2025. Adacot: Pareto-optimal adaptive chain-of-thought triggering via reinforcement learning. *Preprint*, arXiv:2505.11896.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. 2025. Reasoning models can be effective without thinking. *Preprint*, arXiv:2504.09858.

OpenAI. 2024. Openai o1 system card. *OpenAI Technical Report*.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *Preprint*, arXiv:2404.19733.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Preprint*, arXiv:2503.16419.

Songjun Tu, Jiahao Lin, Qichao Zhang, Xiangyu Tian, Linjing Li, Xiangyuan Lan, and Dongbin Zhao. 2025. Learning when to think: Shaping adaptive reasoning in r1-style models via multi-stage rl. *Preprint*, arXiv:2505.10832.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang T. Truong, Chelsea Finn, and Nick Haber. 2025. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. In *NeurIPS 2025 Workshop on Efficient Reasoning*.

Violet Xiang, Rafael Rafailov, Chase Blagden, Nathan Lile, and Nick Haber. 2026. Self-guided thinking: Enabling LLMs to decide when to think.

Junjie Yang, Ke Lin, and Xing Yu. 2025. Think when you need: Self-adaptive chain-of-thought learning. *Preprint*, arXiv:2504.03234.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025. Adaptthink: Reasoning models can learn when to think. *Preprint*, arXiv:2505.13417.

A Additional Training Details

All algorithms initialize from base Qwen3-8B model weights and are optimized using a standard Adam-based optimization. In all experiments, we used the same LoRA configuration: $r = 32$, $\alpha = 32$, and dropout $= 0.05$ with LoRA applied to the target modules q, k, v, and o.

For SFT, we used a learning rate of 5×10^{-4}, while the DPO experiments employ 5×10^{-5}, and DAPO experiments use 5×10^{-6}. A warm-up ratio of 0.03 is applied in all three setups. Different learning rates are selected for each method due to their varying sensitivities to learning dynamics. Extensive hyperparameter tuning was not performed for any of the individual approaches.

In experiments involving a combination of DPO and NLL loss, we set $\lambda_{\text{NLL}} = 1$. For both DPO and DAPO experiments utilizing rollouts, the temperature is set to 1 for generation, with $K = 6$ for DPO and $K = 8$ for DAPO. A smaller K is chosen for DPO due to its pairwise design, which would otherwise result in an exponentially larger number of pairs.

B Hyperparameter Sweeps

We present our full hyperparameter sweeps results in Table 4. We first fix α and a hyperparameter sweep over β_{rs}. For Ada-RS-DPO, we observe a moderate range of β_{rs} (0 - 0.1) can provide the best trade-off between accuracy and selective reasoning. Larger β_{rs} values lead to excessive thinking and lower the tool calling accuracy. Similar trend is observed for Ada-RS-DAPO, a slightly higher range of β_{rs} (0.1 - 0.5) better supports selecting thinking, yielding more consistent improvement.

We further examine the effect of decreasing α. Since α controls the strength of the length pressure; weaker penalties increase the frequency of thinking and can increase total output length. We observe in both experiment groups that overly small α significantly degrade performance and impair the model's adaptive thinking capability, often resulting in unstable or excessive reasoning behaviors.

Table 4: **Hyperparameter sweep for β_{rs} and α.** Sample numbers are reported to the nearest thousand. RS Acceptance Rate denotes the empirical acceptance rate from rejection sampling. Training time decrease denotes the fold change relative to using all produced samples (120k samples or preference pairs).

Method	β_{rs}	α	**Acc.**	**Avg Output Tokens**	**Avg Reason. Tokens**	**Thinking Rate**	**Training Samples**	**RS Accept. Rate**	**Train Time Decrease**
Ada-RS-DPO	1	0.01	84.10	236.52	126.54	50.60	96k	79.00	0.79x
Ada-RS-DPO	0.5	0.01	86.26	233.06	114.25	50.48	86k	71.00	0.71x
Ada-RS-DPO	0.1	0.01	89.24	87.81	0.94	6.10	53k	43.00	0.43x
Ada-RS-DPO	0.1	0.001	87.69	88.00	3.77	28.64	53k	43.00	0.43x
Ada-RS-DPO	0.01	0.01	89.68	84.81	0.83	16.97	28k	23.00	0.23x
Ada-RS-DPO	$\approx$0	0.01	90.04	87.21	0.87	13.03	12k	10.00	0.1x
Ada-RS-DAPO	1	0.005	86.33	152.25	75.32	6.69	115k	96.50	1.3x
Ada-RS-DAPO	0.5	0.005	87.05	81.90	36.31	4.34	78k	65.20	0.7x
Ada-RS-DAPO	0.1	0.005	87.97	81.66	37.68	4.22	54k	45.50	0.5x
Ada-RS-DAPO	0.1	0.001	73.39	118.64	41.55	93.98	54k	45.50	0.5x

Table 5: Performance breakdown by difficulty level with $\alpha = 0.005$ for both Ada-RS methods

Method	**Difficulty**	**Thinking Rate**	**Accuracy**	**Avg Total Tokens**	**Avg Reasoning Tokens**
NFT					
(Thinking-On)	Easy	100.00%	66.10%	458.18	224.31
	Medium	100.00%	67.19%	434.82	224.48
	Hard	100.00%	66.04%	456.62	231.95
NFT					
(Thinking-Off)	Easy	0.00%	73.76%	77.70	0.00
	Medium	0.00%	71.26%	87.09	0.00
	Hard	0.00%	67.52%	88.02	0.00
SFT	Easy	0.40%	93.34%	78.00	3.79
	Medium	71.92%	91.47%	826.50	626.77
	Hard	97.44%	91.51%	1111.73	906.68
DPO	Easy	100.00%	70.38%	445.43	258.70
	Medium	100.00%	66.93%	407.92	244.08
	Hard	100.00%	65.50%	463.14	250.70
Ada-RS-DPO					
($\beta_{rs} = 0.1$)	Easy	17.59%	90.46%	120.60	40.05
	Medium	16.80%	86.22%	134.78	44.76
	Hard	22.64%	87.87%	140.50	48.73
DAPO ALP					
	Easy	39.56%	85.39%	148.09	73.58
	Medium	42.26%	79.00%	150.78	74.78
	Hard	42.86%	78.44%	159.40	78.18
Ada-RS-DAPO					
($\beta_{rs} = 0.5$)	Easy	4.87%	90.76%	78.06	21.39
	Medium	4.99%	83.86%	83.07	24.55
	Hard	2.97%	85.31%	85.92	19.89

Algorithm 1 Ada-RS-DPO (Off-Policy)

Require: Dataset $\mathcal{D}$ of contexts x; teacher policy π_ϕ; student policy π_θ; rollout count K; ALP weight α; RS temperature β_{rs}; NLL weight λ_{NLL}.

$\mathcal{P} \leftarrow \emptyset$ ▷ preference pairs for DPO
for each context $x \in \mathcal{D}$ **do**
  Generate mixed candidates (thinking-on/off):
  Sample $\{y_i^{\text{on}}\}_{i=1}^{K/2} \sim \pi_\phi(\cdot \mid x,\ \texttt{think} = 1)$
  Sample $\{y_i^{\text{off}}\}_{i=1}^{K/2} \sim \pi_\phi(\cdot \mid x,\ \texttt{think} = 0)$
  $\{y_i\}_{i=1}^{K} \leftarrow \{y^{\text{on}}\} \cup \{y^{\text{off}}\}$
  Compute correctness $c_i \leftarrow \mathbb{1}(y_i, x) \in \{0, 1\}$ for each y_i
  Extract think trace t_i from y_i and set $\ell_i \leftarrow |t_i|$ ▷ $|t_i| = 0$ if empty <think>
  $s_K(x) \leftarrow \frac{1}{K} \sum_{i=1}^{K} c_i$ ▷ solve-rate estimate, Eq. (2)
  $r_i \leftarrow c_i - \alpha \cdot s_K(x) \cdot \ell_i$ ▷ ALP reward, Eq. (1)
  Pair-wise rejection sampling to build preferences:
  $\Delta_{\max} \leftarrow \max_{i<j}(r_i - r_j)$
  for each unordered pair (i, j) with $i < j$ **do**
    $\Delta_{ij} \leftarrow r_i - r_j$
    $p_{ij} \leftarrow \exp((\Delta_{ij} - \Delta_{\max})/\beta_{\text{rs}})$ ▷ Eq. (3)
    Draw $u \sim \text{Uniform}(0, 1)$
    if $u < p_{ij}$ **then**
      $y_w \leftarrow \arg\max\{r_i, r_j\}; \quad y_\ell \leftarrow \arg\min\{r_i, r_j\}$
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{(x, y_w, y_\ell)\}$
    end if
  end for
end for
Update θ with DPO on $\mathcal{P}$ plus auxiliary NLL:
  minimize $\mathcal{L}_{\text{DPO}}(\mathcal{P}; \theta) \ + \ \lambda_{\text{NLL}} \mathcal{L}_{\text{NLL}}(\mathcal{P}; \theta)$

Algorithm 2 Ada-RS-DAPO

Require: Dataset $\mathcal{D}$ of contexts x; current policy π_θ; rollout count K; ALP weight α; RS temperature β_{rs}.

for each training step **do**
  Sample minibatch $\{x_b\}_{b=1}^{B} \sim \mathcal{D}$
  for each context x in minibatch **do**
    On-policy rollout group:
    Sample $\{y_i\}_{i=1}^{K} \sim \pi_\theta(\cdot \mid x)$
    Compute correctness $c_i \leftarrow \mathbb{1}(y_i, x)$ and think-length $\ell_i \leftarrow |t_i|$ for each y_i
    $s_K(x) \leftarrow \frac{1}{K}\sum_{i=1}^{K} c_i$ ▷ solve-rate estimate, Eq. (2)
    $r_i \leftarrow c_i - \alpha \cdot s_K(x) \cdot \ell_i$ ▷ ALP reward, Eq. (1)
    Group-wise rejection sampling:
    $\mu \leftarrow \frac{1}{K}\sum_{i=1}^{K} r_i; \quad \sigma \leftarrow \sqrt{\frac{1}{K}\sum_{i=1}^{K}(r_i - \mu)^2}$
    $\mathcal{Y}' \leftarrow \emptyset$ ▷ retained candidate group
    for each candidate y_i **do**
      $p_i \leftarrow \min\Big(\exp\big(((r_i - \mu)/\sigma)/\beta_{\mathrm{rs}}\big),\ 1\Big)$ ▷ Eq. (4)
      Draw $u \sim \mathrm{Uniform}(0, 1)$
      if $u < p_i$ **then**
        $\mathcal{Y}' \leftarrow \mathcal{Y}' \cup \{(y_i, r_i)\}$
      end if
    end for
    DAPO update on retained group:
    Apply one DAPO optimization step using $(x, \mathcal{Y}')$
  end for
end for