Extended Abstract

Motivation Our goal is to enhance Qwen2.5-0.5B's ability to generate high-quality answers with reflection and reasoning. During such reflection, if the model identifies the confidence to be low, it can invoke external tools like Tavily for accurate retrieval.

This advances the traditional one-shot answer generation, which may result in one of the two: 1. the model could simply hallucinate, and produces inaccurate answers 2. the model simply refuses to answer because it does not have the knowledge. Both are not as helpful.

To overcome the challenges above, we have designed two key components in the pipeline: a Generator Model and an Evaluator Model, we experiment with diverse data mixtures to assess performance.

Method We train two components: a generator and an evaluator. The generator is first trained via Supervised Fine-Tuning (SFT) on instruction-following data. We then generate QA pairs, score them using OpenAI's API, and train a reward model to predict those scores. Using this reward model, we apply Proximal Policy Optimization (PPO) to align the generator's outputs with human preferences.

The evaluator is trained via SFT to generate comments from QA pairs, using reasoning traces distilled from DeepSeek-R1-14B. It then analyzes the sentiment of the comment to determine whether to trigger an online search.

At inference, the generator produces an initial answer, and the evaluator generates a comment to assess confidence. If the sentiment is high (positive), the system returns the generator's response. If low (negative), it calls Tavily's API for external retrieval. This reflective-retrieval hybrid approach improves answer accuracy while reducing unnecessary API calls and latency.

Implementation Our dataset merges HotpotQA, focusing on multi-hop and comparative questions, with ScienceQA, emphasizing factual reasoning, providing ground-truth answers for supervised training.

We conducted two experiments to assess reasoning and generalization. The first, using Qwen 411 (bridge-focused), includes 4,187 training and 1,795 test examples. The second, employing Qwen 111 (balanced question types), has 3,149 training and 1,349 test examples. Data was enriched via OpenAI APIs and DeepSeek-R1:14B—OpenAI contributed scalar scores for reward modeling, while DeepSeek provided reflective comments for richer evaluative feedback. Ground truth responses were included as positive supervision during SFT and assigned a reward of 1.0 during PPO training to enhance learning quality.

Results In our quantitative evaluation, we compared the performance of three models—Qwen2.5-0.5B (base), Fine-tuned SFT, and Fine-tuned PPO—across a combined test set containing balanced and bridge-focused distributions. Both fine-tuned models notably outperformed the base model, achieving overall accuracy scores of 0.38 (SFT) and 0.39 (PPO) compared to the base's 0.29. Specifically, the PPO model excelled in comparison questions (0.60 accuracy), showcasing its superior discriminative reasoning capabilities over the SFT (0.53) and base (0.45) models. In open factual questions, both fine-tuned models significantly improved accuracy (0.40) over the base model (0.34). Bridge questions, however, posed greater challenges, yielding modest accuracy gains (base: 0.10, SFT: 0.19, PPO: 0.18). Additionally, we explored the efficiency-accuracy tradeoff using a confidence-based API calling policy. Using DeepSeek-r1:14b as an evaluator, we identified an optimal threshold range (0.2–0.4), balancing high accuracy (around 80%) with efficient external resource usage. These results underline the efficacy of fine-tuning methods, particularly PPO, and highlight the practical advantage of confidence-aware strategies in model deployment.

Discussion Future work includes allowing the model to revise responses using its own generated answers and evaluator comments. We also plan to investigate how data composition and question types influence both generation quality and retrieval efficiency.

Conclusion By leveraging SFT and PPO, our approach achieves consistently better responses across diverse query types. The addition of reflection-based evaluation allows the model to dynamically balance internal reasoning with external search, leading to improved efficiency and robustness.

Reflection-Augmented QA: Reinforcement Learning Meets Online Search

Zhulian Huang
Department of Computer Science
Stanford University
`zlhuang@stanford.edu`

Binbin Li
Department of Computer Science
Stanford University
`binbinli@stanford.edu`

Ying Lu
Department of Computer Science
Stanford University
`yinglu01@stanford.edu`

Abstract

We present a reflective-retrieval framework that enhances the Qwen2.5-0.5B model's ability to generate accurate, reasoned answers while dynamically leveraging external tools when confidence is low. Our system consists of a generator trained via Supervised Fine-Tuning (SFT) and Proximal Policy Optimization (PPO), and an evaluator trained to assess confidence through comment generation. At inference, the evaluator determines whether to return the model's answer or invoke a retrieval API. Experiments on a combined dataset of HotpotQA and ScienceQA show that both SFT and PPO fine-tuning significantly outperform the base model, with PPO especially strong on comparison questions. Moreover, using evaluator-guided retrieval based on DeepSeek-R1:14B, we identify an optimal confidence threshold (0.2–0.4) that balances high accuracy (80%) with efficient API usage. Our approach demonstrates the effectiveness of combining reflection with external knowledge retrieval to improve robustness and response quality.

1 Introduction

Recent advancements in large language models (LLMs) have significantly improved the quality of natural language generation. However, using smaller models to generate high-quality, reliable, and well-grounded responses remains a challenging task. Smaller models often lack the capacity to capture nuanced reasoning or handle complex queries without additional supervision or augmentation.

In this work, we aim to expand the capability of the base model, Qwen2.5-0.5B (with 494M parameters), to generate accurate, reflective, and high-quality responses, while maintaining computational efficiency. Our goal is to enable the model to reason about its outputs and, when necessary, invoke external tools such as online search APIs to improve answer reliability.

To this end, we develop a two-model architecture Figure 1 based on Qwen2.5-0.5B: a *generator model* and an *evaluator model*. The generator model is trained to produce coherent and grounded answers. We begin with supervised fine-tuning (SFT) on curated open-source datasets. Then, we leverage OpenAI-generated quality scores—based on alignment with ground truth responses—to train a reward model using supervised learning. This reward model enables reinforcement learning through Proximal Policy Optimization (PPO), guiding the generator to produce responses with improved alignment to human preferences.

In parallel, the evaluator model is trained to generate reflective comments that assess the quality of the generator's output. It is fine-tuned using distillation traces from the DeepSeek-R1:14B model, learning to emulate reflective thinking. Additionally, we employ a sentiment classifier to evaluate these comments. When low confidence is detected, the evaluator triggers an external search API (e.g., Tavily) to retrieve supplemental information before the generator issues a final response.

This architecture enables the system to combine internal reasoning with external knowledge sources efficiently. When the model is confident, it answers directly. When uncertain, it augments its response using real-time web search. This hybrid approach provides a lightweight yet robust solution for building small-model assistants with dynamic reasoning and retrieval capabilities.

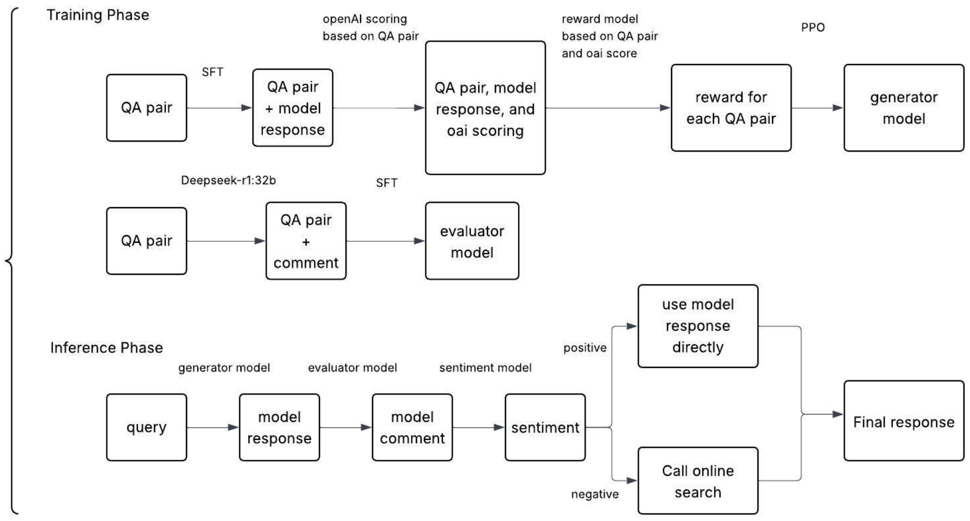


Figure 1: Architecture Overview

2 Related Work

There are a variety of work related to improving search capabilities via RL, especially on how to enable the model to access external information. **R1-searcher** paper has been published to train the model to invoke web search. It is an outcome-based reinforcement learning framework designed to enhance LLMs' ability to autonomously access external knowledge during reasoning via as two-stage process. It involves no supervised fine-tuning or process supervision: Relies solely on RL for training, promoting autonomous exploration and adaptation. It also leverages dynamic integration of reasoning and retrieval and allows LLMs to decide when and how to use external knowledge based on the reasoning context. The paper shows advantages to standard RAG, archives better results than GPT–o4-mini.

Search-R1 paper trains large language models (LLMs) to autonomously conduct online searches as they reason through problems, integrating search engine access directly into the reasoning process. It is Multi-step, interleaved reasoning and retrieval and uses token masking and stable RL training, which ensures robust learning and prevents overfitting to retrieved content.

R1-Searcher++ paper extends the R1-Searcher framework by teaching LLMs to adaptively leverage both internal and external knowledge through a two-stage training strategy. Its key features include two-stage training: Supervised fine-tuning (SFT) for format learning, and RL for dynamic knowledge acquisition and memorization. It encourages the model to use internal knowledge when possible and to efficiently memorize externally retrieved information. It achieves higher accuracy with fewer search invocations compared to vanilla RL-based approaches.

Another very relevant work is **STaR-GATE** (Andukuri et al., 2024), which teaches language models to improve their questioning strategy via self-improvement. STaR-GATE combines active preference elicitation (GATE) with a self-improvement loop (STaR), training a model to ask clarifying questions that significantly enhance its ability to generate personalized and high-quality responses. Through iterative finetuning on conversations that maximize the probability of gold responses, STaR-GATE models become more capable of resolving task ambiguity and tailoring answers to users' needs. This approach aligns closely with our goal of improving search personalization, suggesting that encouraging models to ask targeted clarifying questions could further enhance the effectiveness of personalized search results.

One interesting paper **Adaptive Inference-Time Compute** (Manvi et al, 2025) suggests in the middle of generation, the model could be trained a new capability to predict that restarting the generation will yield better response.

Our problem combines both generating a better answer and also allow the model to learn when to invoke online search, which improves efficiency. Based on the idea of ReAct(Yao et al, 2024), Our approach is through reflection and sentiment decision. We let the model to learn to generate better answers, and in addition to that learn to generate better comment. From the sentiment of the comment, model thinks and reflects, and makes a decision to leverage external search or not.

3 Method

3.1 Generator Model Pipeline

3.1.1 Problem Formulation

We model the fine-tuning of a language model as a two-stage optimization process: Supervised Fine-Tuning (SFT) followed by Reinforcement Learning from Human Feedback (RLHF) using Proximal Policy Optimization (PPO).

Given a dataset

$$\mathcal{D} = \{(x_i, y_i, r_i)\}_{i=1}^{N}$$

where x_i is a user query, y_i is a model-generated response, and $r_i \in [0, 1]$ is a quality score produced by a proprietary scoring model from OpenAI to approximate human preferences, our objective is to learn a policy π_θ that maximizes the expected reward:

$$\mathbb{E}_{x \sim \mathcal{D},\ y \sim \pi_\theta(\cdot|x)}[R(x, y)]$$

Here, $R(x, y)$ is a learned reward function trained to approximate the quality scores from OpenAI's annotator model, serving as a proxy for human preference.

3.1.2 Supervised Fine-Tuning (SFT)

The base model used for SFT was Qwen2.5-0.5B, a 494M parameter autoregressive language model. Training was conducted using a causal language modeling loss on query-response pairs.
Objective:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(x,y) \sim \mathcal{D}_{\text{SFT}}} \left[\sum_{t=1}^{|y|} \log \pi_\theta(y_t \mid x, y_{<t}) \right]$$

where $\mathcal{D}_{\text{SFT}} = \{(x_i, y_i) \mid r_i \geq \tau\}$, $\tau = 0.7$.

Data Enhancement:

$$\mathcal{D}_{\text{enhanced}} = \mathcal{D}_{\text{SFT}} \cup \{(x_i, y_i^*) \mid r(x_i, y_i^*) = 1.0\}$$

SFT was performed with a learning rate of 5e-5, batch size of 4 (with gradient accumulation), and a maximum sequence length of 512 tokens. Training was conducted for 3 epochs using AdamW with gradient clipping and checkpointing enabled for memory efficiency.

3.1.3 Reward Model Training

The reward model architecture consisted of a frozen Qwen2.5-0.5B encoder followed by a multi-layer perceptron reward head. The head featured LayerNorm, dropout (0.1), and a sequence of decreasing hidden dimensions ending in a scalar output. Training used a normalized Mean Squared Error loss and early stopping based on validation RMSE.

Train reward model $R_\phi(x, y)$ using:

$$\mathcal{L}_{\text{reward}}(\phi) = \mathbb{E}_{(x,y,r)\sim\mathcal{D}}\left[\|R_\phi(x,y) - r\|^2\right] + \lambda\|\phi\|_2^2$$

3.1.4 Proximal Policy Optimization (PPO)

PPO was initialized from the SFT-finetuned model and optimized with reward-weighted updates. The reward signal was computed using the trained reward model, and KL divergence with a frozen SFT reference model was used to prevent policy drift.
Objective:

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta}\left[\mathcal{L}_{\text{policy}}(x,y) + \beta \cdot \mathcal{L}_{\text{KL}}(x,y)\right]$$

Components:

- Policy Loss:
$$\mathcal{L}_{\text{policy}}(x,y) = -\sum_{t=1}^{|y|} w(R_\phi(x,y)) \log \pi_\theta(y_t \mid x, y_{<t})$$
- Weight: $w(r) = \frac{1}{1+\max(0,r)}$
- KL Penalty:
$$\mathcal{L}_{\text{KL}}(x,y) = \text{KL}(\pi_\theta(\cdot|x)\|\pi_{\text{ref}}(\cdot|x))$$

PPO Algorithm: The training procedure begins by initializing the policy model π_θ with the parameters of the reference model π_{ref}. For each training batch, the current policy π_θ is used to generate a response y_i conditioned on the input query x_i. The generated response is then scored using the reward model $R_\phi(x_i, y_i)$, which approximates human preferences. Finally, the model parameters θ are updated by applying a gradient step to minimize the PPO loss, denoted as $\mathcal{L}_{\text{PPO}}$, using a learning rate α.

3.2 Evaluator Model Pipeline

3.2.1 Objective

Our goal is to base on Qwen2.5-0.5B as the base model, and prompt it to generate comments on a given QA pair. We use SFT to train this model, where we collect (query, model response, training response) tuples. Deepseek model is used to comment on the (query, model response) pair, as well as (query, training response) pair. As you could imagine, given the 0.5B model, (query, model response) generally do not yield good answers and deepseek comments have quite a lot of negative answers, whereas for (query, training response) pair, since it is ground truth data, the deepseek response are mostly positive.

3.2.2 Prompt

We prompt the Qwen2.5-0.5B using alpaca format with the following:

> **Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.**
> **Instruction:**
> You are a knowledgeable analyst who has world knowledge and is skilled at evaluating question–answer pairs. Please provide a comment on the input.
>
> **Input:** {*query + answer pair*}
>
> **Comment:** {*model-generated evaluation*}

In our experiment, 14,676 synthetic comments are generated according to the (query, model response, comment) tuple and (query, training response, comment) tuples, with the help of distillation from deepseek-r1:14b, including thinking traces and comments. Note that for (query, training response) pair, the comment will naturally be positive, since training response is ground truth data. We observe the same in our data with scores generally above 0.7. These serve as positive training examples in addition to the correct model response. We trained 10 epochs each, with comment data including thinking trace and without respectively. We use this data to train a SFT model that predicts the next tokens of a comment given the instruction and input, with the input containing both query and (model/training) response.

We have done two experiments, one with the thinking trace from deepseek distillation, and one without the distillation. We find that training loss keeps decreasing for both experiments. The experiment without the thinking trace decreases the training loss faster, starting with 2.55, and ending with 0.1279, while the one with the thinking trace ending with 0.325, all other parameters remaining equal as described in the above configuration.

After generating the comment, we assess its sentiment. We take a simple approach and use the base model Qwen2.5-0.5B to judge the sentiment of the comment with prompting, and ask the model to produce a score between 0 (highly negative) and 1 (highly positive).

Based on the sentiment score, the model switches to online search (e.g., Tavily API) if the score is below the threshold; otherwise, the model produces its own response. This enables a form of reflection for smart decision. However, in our training, we find that instruction following to generate comments remains relatively weak and needs improvement.

Another experiment we have done is that we also study an ideal case if we leverage OpenAI-level to give out sentiment. If the small model learns to comment and produce sentiment of the same quality as openAI model, we see a clear gain and tipping point where the model choses to save calling online search API when it is confident, and calls online search otherwise. The saving of calling online search does not compromise answer quality, which is expected.

This hybrid approach shows an efficiency gain while preserving answer quality.

4 Experimental Setup

4.1 Data Distribution

To evaluate the model's performance, we utilized data composed of three distinct question types, each designed to test a different aspect of reasoning Figure 2:

- **Bridge Questions:** These questions require multi-hop reasoning by connecting different pieces of information through an intermediate "bridge" entity. To find the answer, the model must first identify this bridge and then use it to find the final fact.

- **Comparison Questions:** These questions require the model to compare two or more entities along a specific attribute, often involving quantities or qualities. Answering them involves retrieving facts for each entity and then performing a comparative operation.

- **Open Factual Questions:** These are direct, fact-based questions that can typically be answered by retrieving a single piece of information from a broad knowledge source. They test the model's ability to locate and extract specific facts without complex reasoning chains.

To enhance learning quality, we incorporated ground truth responses in both supervised and reinforcement learning stages. Ground truth responses were used as high-quality positive samples during supervised fine-tuning (SFT) to teach the model ideal response patterns. In PPO training, these ground truth responses were assigned a reward score of 1.0, providing a consistent upper bound signal to guide the policy optimization toward preferred answers.

4.2 Model Configuration

Base Model: The base model employed in this study is Qwen2.5-0.5B, a decoder-only transformer architecture with approximately 494 million parameters. It supports a context window of up to 512 tokens and uses FP32 precision to ensure numerical stability during training. The tokenizer is configured with `<|endoftext|>` as both the padding and end-of-sequence token. Padding is applied on the right side of the input, truncation is enabled, and the maximum input length is set to 512 tokens.

4.3 Generator Model Training Hyperparameters

SFT Configuration: The supervised fine-tuning (SFT) phase used a learning rate of 5×10^{-5}, with a batch size of 4 and 4 gradient accumulation steps, resulting in an effective batch size of 16 samples per update. Training was performed over 3 epochs with a warmup ratio of 0.1, weight decay of 0.01, and a maximum gradient norm of 1.0. Training used full-precision (FP32), with gradient checkpointing enabled for memory efficiency.

Reward Model Configuration: The reward model was trained using a learning rate of 2×10^{-5} and a batch size of 8 across 5 epochs. A validation split of 0.2 was applied, and early stopping was triggered with a patience of 3 epochs. Regularization was achieved with a weight decay of 0.01 and dropout rate of 0.1.

PPO Configuration: Proximal Policy Optimization (PPO) training was conducted using a very low learning rate of 1×10^{-7} for stability, with a batch size of 4 and 2 training epochs. The KL divergence penalty coefficient β was set to 0.2. Generation parameters included a maximum of 100 new tokens, temperature of 0.7, top-p sampling threshold of 0.9, and maximum gradient norm of 0.5.

4.4 Evaluation Protocol

Reward Model Evaluation: The reward model was evaluated using Pearson correlation $\rho(R_\phi, r_{\text{true}})$, root mean square error (RMSE) computed as $\sqrt{\mathbb{E}[(R_\phi - r)^2]}$, and the R^2 coefficient of determination.

Policy Evaluation: The improvement in policy quality was measured by the change in expected reward: $\Delta R = \mathbb{E}[R_\phi(x, y_{\text{PPO}})] - \mathbb{E}[R_\phi(x, y_{\text{SFT}})]$. Manual response evaluation was also conducted to assess the relevance and coherence of generated outputs.

4.5 Evaluator Model

We train the evaluator model to generate proper comments and provide sentiment score depends on whether the evaluator thinks the model correctly, partially and incorrectly address the query.

Training:

- Model: Qwen2.5-0.5B
- Optimizer: adamw 8bit
- Batch size: 2
- gradient accumulation step: 4
- Epochs: 10
- Loss: Causal LM loss
- learning rate: 0.001
- max seq length: 1024

We experiment with comment data generation both with and without thinking trace.

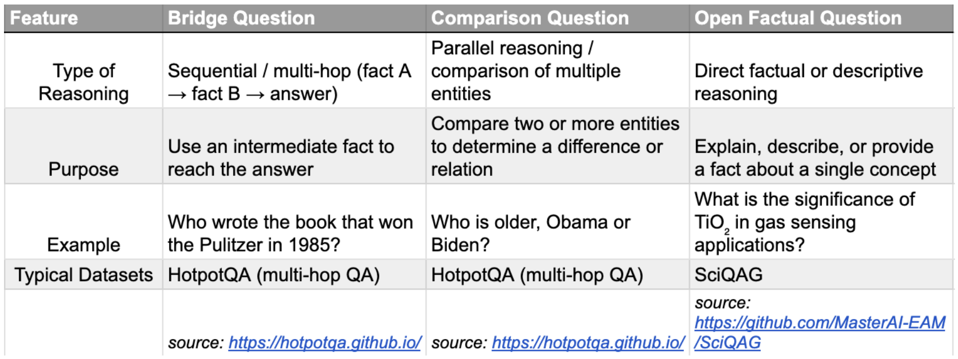
4.6 Implementation Details

Software Environment: The experiments were conducted using the following software versions: `transformers==4.52.4`, `torch==2.7.0`, `datasets==3.6.0`, `pandas==2.2.3`, and `scikit-learn==1.6.1`, `trl==0.15.2`, `unsloth==2025.5.2`, `pandas==2.2.3`, `ollama==0.4.8`.

Reproducibility: To ensure reproducibility, a fixed random seed was used across all training stages. Model checkpoints were saved every 50 steps, and all configurations were logged in JSON format. Library versions were kept consistent throughout the pipeline.

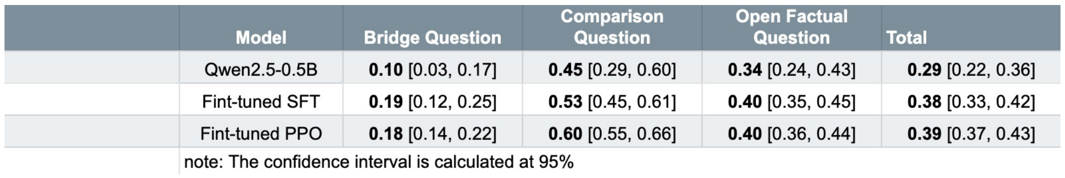
Memory Management: Gradient checkpointing was used to reduce memory consumption. Mixed-precision training was disabled (`fp16 = False`) for stability, and `torch.cuda.empty_cache()` was called between model stages to free up GPU memory.

5 Results



Feature	Bridge Question	Comparison Question	Open Factual Question
Type of Reasoning	Sequential / multi-hop (fact A → fact B → answer)	Parallel reasoning / comparison of multiple entities	Direct factual or descriptive reasoning
Purpose	Use an intermediate fact to reach the answer	Compare two or more entities to determine a difference or relation	Explain, describe, or provide a fact about a single concept
Example	Who wrote the book that won the Pulitzer in 1985?	Who is older, Obama or Biden?	What is the significance of TiO_2 in gas sensing applications?
Typical Datasets	HotpotQA (multi-hop QA)	HotpotQA (multi-hop QA)	SciQAG
	source: https://hotpotqa.github.io/	*source: https://hotpotqa.github.io/*	*source: https://github.com/MasterAI-EAM/SciQAG*

Figure 2: Three types of QA pairs



	Model	Bridge Question	Comparison Question	Open Factual Question	Total
	Qwen2.5-0.5B	**0.10** [0.03, 0.17]	**0.45** [0.29, 0.60]	**0.34** [0.24, 0.43]	**0.29** [0.22, 0.36]
	Fint-tuned SFT	**0.19** [0.12, 0.25]	**0.53** [0.45, 0.61]	**0.40** [0.35, 0.45]	**0.38** [0.33, 0.42]
	Fint-tuned PPO	**0.18** [0.14, 0.22]	**0.60** [0.55, 0.66]	**0.40** [0.36, 0.44]	**0.39** [0.37, 0.43]
	note: The confidence interval is calculated at 95%				

Figure 3: Accuracy Evaluation for Three Models

5.1 Quantitative Evaluation

We evaluate the performance of three models—Qwen2.5-0.5B (base), Fine-tuned SFT, and Fine-tuned PPO—on a combined test set composed of both balanced and bridge-focused distributions. Evaluation scores are reported for three question types: Bridge, Comparison, and Open Factual, as well as an overall score. Figure 3 summarizes the average scores and corresponding 95% confidence intervals. The Accuracy Store is generated by DeepSeek-r1:14b which is used an expert evaluator to assess the correctness of each response relative to the provided ground truth. The model returns the numeric score between 0 and 1, where 1 represents a fully correct response, and 0 represents a completely incorrect answer.

Overall Performance. Both fine-tuned models significantly outperform the base Qwen2.5-0.5B model. The base model achieves an overall score of 0.29 [0.22, 0.36], while the SFT and PPO models reach 0.38 [0.33, 0.42] and 0.39 [0.37, 0.43], respectively. These results demonstrate the effectiveness of both supervised and reinforcement learning in improving response quality.

Comparison Questions. The greatest improvements are observed in comparison questions, where the PPO model performs best with a score of 0.60 [0.55, 0.66]. The SFT model follows closely at 0.53 [0.45, 0.61], both outperforming the base model's 0.45 [0.29, 0.60]. This indicates that preference modeling via reward-based fine-tuning is especially effective in scenarios that require discriminative reasoning.

Open Factual Questions. For open factual questions, both SFT and PPO models perform equally well with scores of 0.40, substantially improving upon the base model's 0.34 [0.24, 0.43]. This shows that both fine-tuning approaches contribute to enhanced factual accuracy in generated responses.

Bridge Questions. Performance on bridge questions remains lower across all models, though fine-tuning still offers gains. The base model scores 0.10 [0.03, 0.17], SFT improves to 0.19 [0.12, 0.25], and PPO reaches 0.18 [0.14, 0.22]. The relatively modest improvements suggest that bridge questions—often requiring multi-hop reasoning—remain a challenge for smaller models.

In summary, PPO shows the strongest results overall, particularly on comparison-type questions, while SFT is competitive and slightly more stable across distribution shifts. Both methods provide robust gains over the base model.

5.2 Qualitative Analysis

To complement the quantitative results, we conducted a qualitative analysis of model-generated responses. We examined a sample of outputs from each model across question types and annotated them based on coherence, relevance, groundedness, and reasoning depth.

Base Model. The base Qwen2.5-0.5B model frequently produces surface-level responses. While it can answer straightforward factual queries, it often fails to deliver consistent or multi-step reasoning, particularly for bridge and comparison questions.

Fine-tuned SFT. The SFT model shows notable improvements in coherence and fluency. Its responses are more structured and grounded, especially in bridge-style queries where linguistic templates help scaffold reasoning. However, it sometimes overfits to seen patterns and may hallucinate confidence in edge cases.

Fine-tuned PPO. The PPO model generates the most nuanced and human-aligned responses. Its use of a reward model appears to guide more cautious and reflective answers, especially in comparison questions. We observe instances where the model explicitly references key distinctions, demonstrating improved decision boundaries. That said, PPO outputs can occasionally be verbose or overly conservative in uncertain contexts.

Overall, qualitative findings align with quantitative metrics: PPO fine-tuning leads to more accurate and preference-aligned behavior, while SFT provides strong improvements with greater simplicity and robustness. Notably, both fine-tuned models significantly improve answer quality and reasoning fluency over the base model.

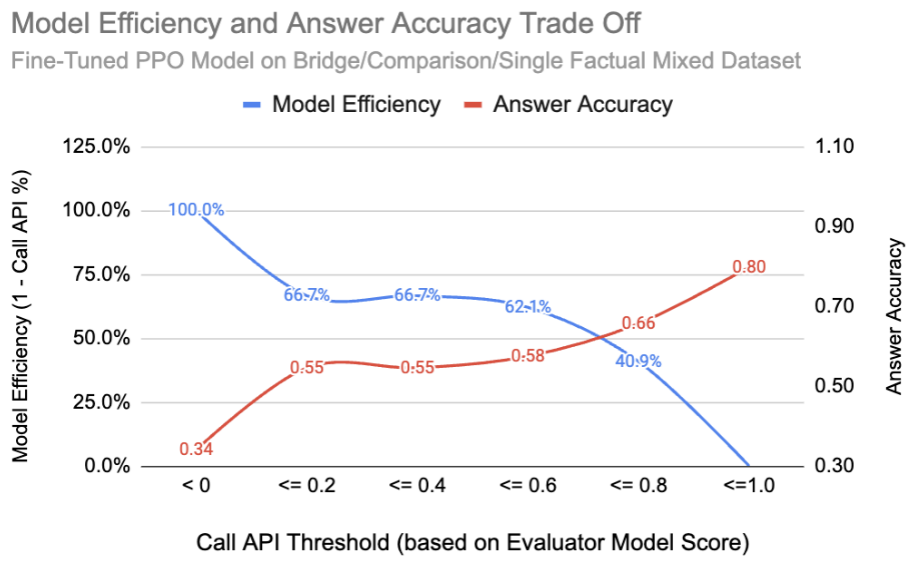


Figure 4: Model Efficiency and Answer Accuracy

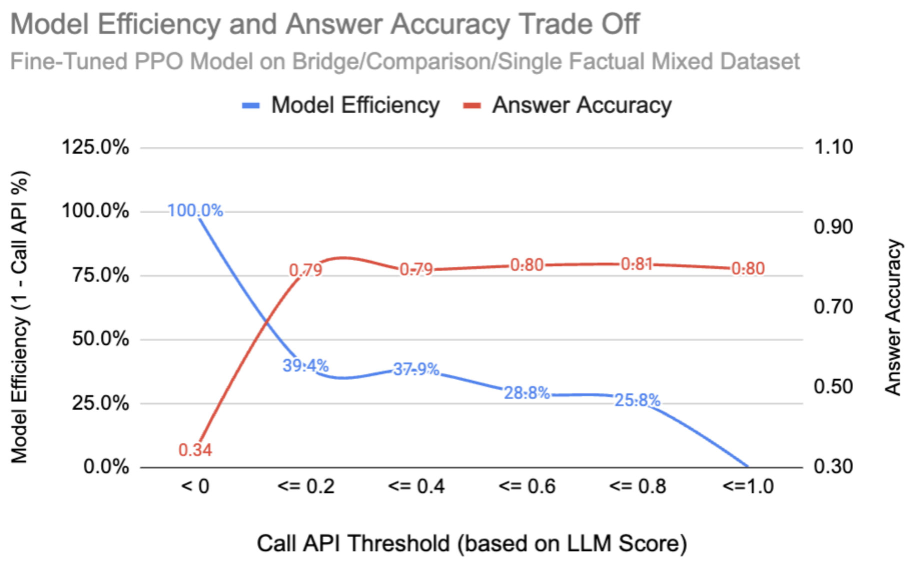


Figure 5: Model Efficiency and Answer Accuracy

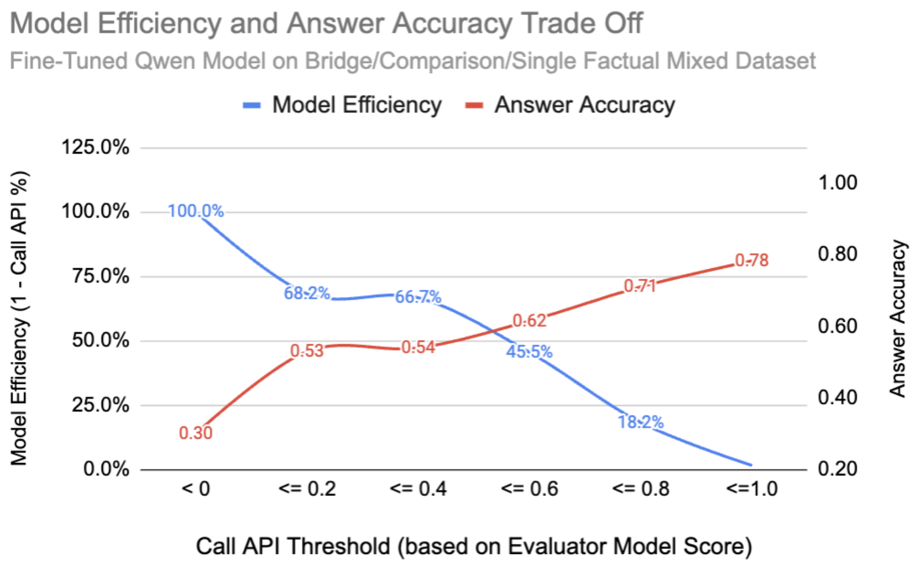


Figure 6: Model Efficiency and Answer Accuracy

5.3 Efficiency-Accuracy Tradeoff via API Calling Policy

To explore the practical utility of external knowledge integration, we evaluate how model confidence can guide selective invocation of an online search API. Specifically, we use DeepSeek-r1:14b as a reference to evaluate the response and provide a confidence level on the response. These scores serve as proxies for response confidence, which in turn inform the decision to either provide a direct answer or call an external search API (e.g., Tavily) for assistance. Note that the scoring mechanism of DeepSeek exhibits slightly non-deterministic behavior, resulting in outputs with certain variance across identical test cases.

Figure 6 is based on model evaluator, and Figure 5 is based on DeepSeek-r1:14b evaluator. The two plots illustrate the tradeoff between model efficiency (defined as API call rate) and answer accuracy as the API-calling threshold is varied based on model response score. The rule here is when the confidence score is below the threshold, invoke online search.

Model Efficiency. For Figure 6 based on model evaluator, at the lowest threshold (close to 0), the model never calls the API, yielding 100% efficiency but poor accuracy (34%). As the threshold increases, the frequency of API invocation rises, reducing model-side computation but improving answer quality. Efficiency drops steadily—from 66% at threshold ≤ 0.2 to 0% at ≤ 1.0—indicating the model relies more heavily on external sources when uncertain.

On the contrary, based on Figure 5 based on DeepSeek-r1:14b evaluator, the starting point is same, but as the threshold increases, efficency drops very quickly to 39.4% between 0.2 and 0.4.

Answer Accuracy. For DeepSeek-r1:14b evaluator, answer accuracy improves rapidly at first, from 0.34 at threshold < 0 to 0.79 at ≤ 0.2. It peaks around 0.81 at thresholds ≤ 0.8, before slightly tapering off. This suggests that the evaluator model's confidence score is well-aligned with response correctness, making it an effective signal for triggering external knowledge retrieval. For model evaluator, accuracy improves to 55% at the beginning, and slowly increases afterwards. This is due to model evaluator tending to produce more extreme scores compared with DeepSeek, and the small model tends to be less conversative. However, the internal knowledge of the 0.5B-parameter model limits its capability, so answer accuracy overall is lower as threshold increases.

Optimal Tradeoff Point. We evaluate based on model's scoring vs DeepSeek scoring. We see a very interesting pattern: for the same set of (query, response) pair, DeepSeek tends to be more conservative, and calls online API more aggressively(60% of calls when score is under 0.2), but the good side is the answer accuracy is much higher, reaching around 80%. The model itself tends to be less conservative, so efficiency is higher (33% of API calls at 0.2 threshold), but accuracy is lower(around 55%) because of its internal knowledge's limitation.

For both Figure 6 and Figure 5, the threshold range between ≤ 0.2 to ≤ 0.4 appears to offer the best balance between efficiency and accuracy.

Implication. These results represent an interesting comparison to study our small model's evaluator vs DeepSeek model evaluator. In practice, our system is designed to use a lightweight internal evaluator to generate and score these comments autonomously. In practice, for client side LLM, we have strong need to deploy small but power LLM models. The findings demonstrate the promise of a confidence-aware API calling mechanism, where model-generated uncertainty signals can effectively balance response accuracy with efficiency and external resource usage.

6 Discussion

For future work, we plan to run another experiment by letting the model to regenerate the response based on the query and its generated response and comments. This may enforce the model to rethink and regenerate the response given additional signals. We could do so by modeling the probability that the model can generate a better response, and invoke that rethinking when the probability is high. This is also suggested by Manvi et al paper.

We also plan to study how different data mixture and types of questions will affect model's performance on answer accuracy, as well as on the efficiency of API call. We currently have more bridge questions in the training data mixture, but it will be interesting to study the performance using balanced training data, or training data that involves more of the other two types.

7 Conclusion

We conclude that by leveraging Supervised Fine-Tuning (SFT) and Proximal Policy Optimization (PPO), we are able to achieve uniformly improved performance across all three question types—bridge, comparison, and open factual—consistently outperforming the base Qwen2.5-0.5B model. Among them, PPO shows particularly strong gains in comparison questions, likely due to its alignment with preference-driven reward signals. SFT, on the other hand, demonstrates more stable performance across different question distributions, making it a robust choice under varying data conditions.

In addition to generation quality, we introduce a reflection and assessment mechanism in which the model generates an evaluation comment for each response. By analyzing both the content and sentiment of this evaluation, the system is able to intelligently determine whether to rely on its internal knowledge or invoke an external online search API. This dynamic decision-making process leads to measurable efficiency gains without compromising answer quality.

Together, these results highlight the effectiveness of combining fine-tuning techniques with reflective self-assessment and selective retrieval. Our approach enables a lightweight language model to deliver high-quality, context-aware responses while efficiently balancing internal reasoning and external knowledge access.

Finally, our code is here for reference: https://github.com/yinglu1985/cs224r/tree/main

8 Team Contributions

- **Zhulian Huang:** was primarily responsible for the reinforcement learning component of the project, including:
  - Adapting the Supervised Fine-Tuning (SFT) checkpoint as the initialization for the policy model.
  - Designing and implementing the reward model to predict response quality based on OpenAI-assigned scores.
  - Applying Proximal Policy Optimization (PPO) to fine-tune the policy model for improved response generation.
- **Binbin Li :**
  - Led the data preparation and processing efforts, ensuring the quality and consistency of both raw and structured inputs.
  - Developed and implemented pipelines for synthetic data generation to supplement limited real-world examples and enhance training coverage.
  - Fine-tuned a Supervised Fine-Tuning (SFT) model using various data mixtures to explore the impact of different training strategies.
  - Designed and implemented an API calling process to generate online search results.
  - Generated model evaluation metrics.
- **Ying Lu:**
  - Overall system design for training and inference.
  - synthetic data generation with deepseek and openAI scoring.
  - SFT training for generator model on QA pair.
  - evaluator model training on (query, response, comment) pair.
  - Model evaluation and efficiency gain analysis.

References

Song, H., et al. *R1-Searcher: Incentivizing the Search Capability in LLMs via Reinforcement Learning*, 2025.

Jin, Bowen, et al. *Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning*, 2025.

Andukuri, C., et al. *STaR-GATE: Teaching Language Models to Ask Clarifying Questions*. 2024.

Song, H, et al. *R1-Searcher++: Incentivizing the Dynamic Knowledge Acquisition of LLMs via Reinforcement Learning*, 2025.

Yao, S, et al. *ReAct: Synergizing Reasoning and Acting in Language Models*, 2024.

Manvi, R, et al, *Adaptive Inference-Time Compute: LLMs Can Predict if They Can Do Better, Even Mid-Generation*