When Should LLMs Search? Counterfactual Supervision for Search Routing

Minho Kim [1 2]

Abstract

Search-augmented language models can use external evidence to compensate for limitations in parametric knowledge, but search is not uniformly beneficial: models may call search for questions they can already answer, or rely on noisy evidence when correction, clarification, or abstention would be more appropriate. We formulate this as an instance-level search-routing problem: deciding whether search is needed to improve task success relative to a no-search execution. To derive supervision, we compare no-search and forced-search outcomes for the same question and construct an oracle over `NO_SEARCH`, `SEARCH`, and `UNSOLVED` based on task-specific success. Using this oracle as both an evaluation criterion and a learning signal, we train search-routing policies with supervised fine-tuning and preference optimization, improving routing macro-F1 on oracle-eligible examples from 0.7082 to 0.8235 for Gemma E2B and from 0.7053 to 0.8365 for Qwen3.5-4B. Further analysis shows that the learned policies reduce model-specific routing failures: Gemma primarily learns no-search restraint, while Qwen further reduces missed search; residual `UNSOLVED` cases reveal heterogeneous bottlenecks involving model capacity, retrieval budget, evidence use, and policy behavior.

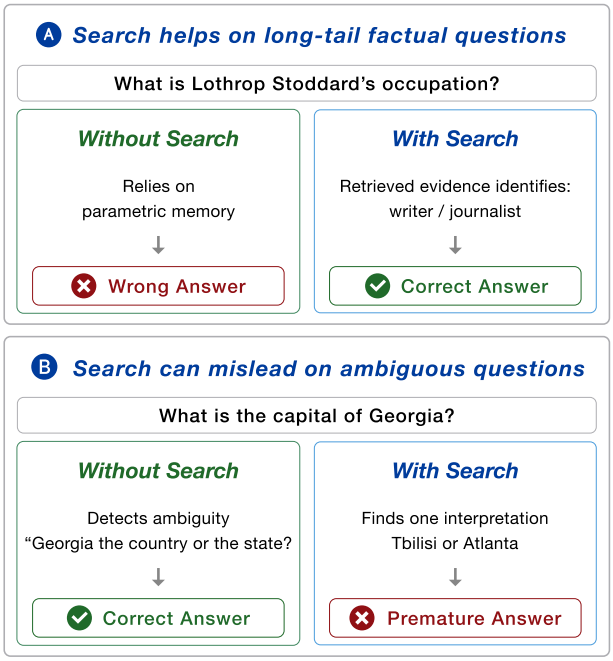

Figure 1. Motivation for instance-level search routing. Search can help, be unnecessary, or mislead, motivating instance-level routing.

1. Introduction

Language models equipped with search tools can use external evidence to answer questions that are difficult to resolve from parametric knowledge alone, including queries about long-tail factual knowledge, recently updated information, or facts that are unlikely to be reliably stored in model parameters (Lewis et al., 2020; Izacard et al., 2023; Mallen et al., 2023). However, the availability of a search tool does not imply that search should be used for every question. For questions the model can already answer, responding without search may be cheaper and less vulnerable to noisy retrieved evidence. For questions with false premises or missing context, the appropriate response may be to correct the premise, ask for clarification, or abstain, rather than retrieve external evidence (Amayuelas et al., 2024; Xie et al., 2026). In practice, the correct first action depends on the instance: a long-tail factual question may require search to recover evidence, whereas an underspecified question may be better resolved by clarification than by retrieving generic search results, as illustrated in Figure 1.

We therefore view search-augmented language models not merely as systems that generate search queries, but as systems that must first perform *search routing*: deciding, for each input question, whether to respond without search or

[1]Sangmyung University [2]DMTLABS. Correspondence to: Minho Kim <ilw2y123@gmail.com>.

Accepted at the FAGEN Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026. *Copyright 2026 by the author.*

call a search tool. This first-action decision is important because routing errors arise in two opposing directions. A model that is too reluctant to search may answer directly even when its parametric knowledge is insufficient, producing plausible but incorrect responses. Conversely, a model that relies too heavily on search may invoke retrieval for questions it can already answer, or for questions that search cannot resolve, increasing cost and exposing the final answer to irrelevant or misleading evidence. The objective is not aggregate search frequency, but calling search only when it improves task success.

This framing differs from standard tool-use evaluation. Many tool-calling benchmarks focus on whether a model calls the correct tool with the correct arguments once tool use is known to be appropriate (Patil et al., 2025). Recent work on when-not-to-call behavior broadens this view by evaluating whether models can avoid unnecessary tool calls, ask follow-up questions, or admit inability to answer (Ross et al., 2025). Our setting focuses specifically on search-augmented question answering, where the same tool may be helpful for one factual question, unnecessary for another, and actively unhelpful for a search-boundary question requiring correction, clarification, or abstention. This motivates an instance-level supervision signal that is grounded in observed response outcomes rather than in a direct judgment of whether search "seems useful."

We address this problem through two research questions.

RQ1. How accurately can a search-enabled language model identify the instances for which search improves task success relative to responding without search?

RQ2. Can models be trained to improve this instance-level search-routing behavior, reducing both missed search and unnecessary search?

To operationalize search need without relying on a direct human or judge annotation of search usefulness, we construct paired counterfactual traces for each question. In the no-search trace, denoted $N(q)$, the model responds without access to the search tool. In the forced-search trace, denoted $S(q)$, the model must first issue a search call and then generate a final response using the returned evidence. A task-specific evaluator maps each final response to binary success: factual correctness for answerable questions, and adequate resolution for false-premise or underspecified questions, including correction, clarification, or abstention.

We then derive a routing oracle from the paired outcomes. If the no-search trace succeeds, we assign `NO_SEARCH`, since search is not necessary even if the forced-search trace also succeeds. If no-search fails but forced-search succeeds, we assign `SEARCH`, since search recovers a no-search failure. If both traces fail, we assign `UNSOLVED`. These examples do not provide a reliable binary routing target: the failure may arise from retrieval quality, query formulation, evidence use, answer synthesis, model capacity, or the question itself. We therefore exclude `UNSOLVED` examples from routing training and routing-accuracy computation, while retaining them as a diagnostic subset.

This outcome-based oracle serves both evaluation and training. For evaluation, we collect a free-policy trace $P(q)$ in which the search tool is available but not forced, and compare the model's first action against the oracle label on examples labeled `NO_SEARCH` or `SEARCH`. This free-policy trace is a prompt-only selective-search baseline: the base model is explicitly instructed to call search only when it would materially improve the response, and otherwise to answer, correct, clarify, or abstain without using the tool. This lets us distinguish missed search, where the model answers without search despite search being necessary, from unnecessary search, where the model calls search despite no-search being sufficient. For training, we use the same oracle to construct supervised fine-tuning examples and preference pairs. Supervised fine-tuning teaches the model to imitate oracle-consistent first actions, while Preference Optimization encourages the model to prefer the appropriate first assistant completion for the same question.

Empirically, search need is model-dependent even with the same question set and search tool: parametric knowledge and search-conditioned solving ability can change whether an instance is labeled `NO_SEARCH`, `SEARCH`, or `UNSOLVED`. Prompted base models nevertheless exhibit both under-search and over-search, showing that selective-search instructions alone do not recover this boundary. Training on model-specific counterfactual outcomes improves routing: on held-out oracle-eligible examples, SFT and Preference Optimization raise macro-F1 from 0.7082 to 0.8235 for Gemma E2B and from 0.7053 to 0.8365 for Qwen3.5-4B; error-direction and `UNSOLVED` analyses separate corrected routing errors from residual bottlenecks beyond first-action routing.

These findings rest on three contributions.

- **Instance-level routing.** We frame search use as an instance-level `NO_SEARCH`/`SEARCH` decision, where `NO_SEARCH` includes direct answering, premise correction, clarification, and abstention.
- **Counterfactual oracle.** We derive an outcome-based oracle from paired no-search and forced-search traces and use it for both routing evaluation and training.
- **`UNSOLVED` diagnostics.** We retain dual-failure cases as a diagnostic subset, showing residual retrieval, evidence-use, synthesis, and model-capacity bottlenecks beyond the initial search decision.

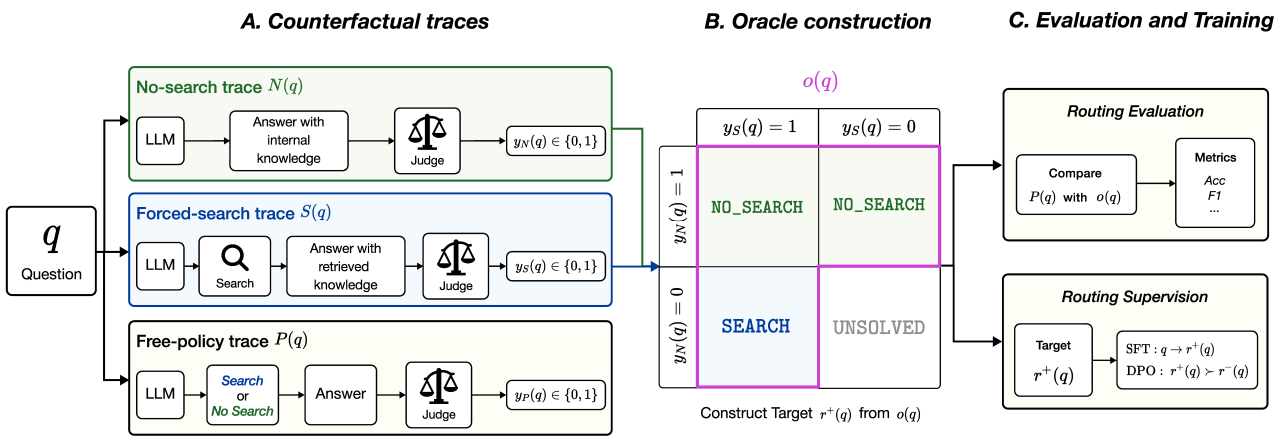

Figure 2. Counterfactual search-routing pipeline. (A) We collect controlled no-search and forced-search traces, together with a free-policy trace collected under a selective-search prompt. (B) Paired no-search and forced-search outcomes induce a routing oracle over NO_SEARCH, SEARCH, and UNSOLVED. (C) The oracle is used both for routing supervision and for free-policy routing evaluation.

2. Method

We formalize search routing as a first-action decision. Given a question q, the policy selects $a(q) \in \{\texttt{NO_SEARCH}, \texttt{SEARCH}\}$, where NO_SEARCH denotes a no-tool first response and SEARCH denotes a first-turn search call. Figure 2 gives an overview; below we define the traces, oracle, evaluation target, and training objectives.

For each question q, we collect two controlled counterfactual traces: $N(q)$, generated with search disabled, and $S(q)$, generated under a forced first-turn search call. We evaluate only the final response in each trace:

$$y_N(q) = \mathrm{Eval}(q, N(q)), \qquad y_S(q) = \mathrm{Eval}(q, S(q)),$$

where $y_N, y_S \in \{0, 1\}$. Eval applies task-specific success criteria; details and prompt templates are in Appendices A and C.

We derive a routing oracle from the paired outcomes:

$$o(q) = \begin{cases} \texttt{NO_SEARCH}, & y_N(q) = 1, \\ \texttt{SEARCH}, & y_N(q) = 0 \wedge y_S(q) = 1, \\ \texttt{UNSOLVED}, & y_N(q) = 0 \wedge y_S(q) = 0. \end{cases}$$

Rows with $o(q) = \texttt{UNSOLVED}$ are excluded from routing supervision and routing metrics because paired failures do not identify a reliable first-action target. We retain them for diagnostic analysis in Section 5 and Appendix B. Table 1 summarizes the induced buckets and labels.

Because the no-search and forced-search outcomes can differ across models, the induced oracle is model-specific rather than a fixed question-level annotation. Supervision therefore targets the corresponding base model's empirical, trace-conditional boundary between cases it can solve without search and cases recovered by search. If no-search already succeeds, the instance is labeled NO_SEARCH even when forced search also succeeds, so SEARCH labels mark only observed no-search-to-search recoveries.

Table 1. Oracle labels induced by counterfactual no-search and forced-search outcomes.

$y_N(q)$	$y_S(q)$	Bucket	Oracle
1	1	$N1_S1$	NO_SEARCH
1	0	$N1_S0$	NO_SEARCH
0	1	$N0_S1$	SEARCH
0	0	$N0_S0$	UNSOLVED

For free-policy evaluation, we collect $P(q)$ under a selective-search prompt: search is available but should be called only when it materially improves the response; otherwise the model should answer, correct, clarify, or abstain without the tool. We parse the first action as $p(q) = \texttt{SEARCH}$ if it calls search and $p(q) = \texttt{NO_SEARCH}$ otherwise. When collected from the base model before routing training, this trace is denoted $P_{\mathrm{pre}}(q)$, and the corresponding prompted base routing policy is P_{pre}. Routing evaluation measures agreement between $p(q)$ and $o(q)$ on examples with $o(q) \in \{\texttt{NO_SEARCH}, \texttt{SEARCH}\}$. We call $o(q) = \texttt{SEARCH}, p(q) = \texttt{NO_SEARCH}$ a missed search, and $o(q) = \texttt{NO_SEARCH}, p(q) = \texttt{SEARCH}$ an unnecessary search. Generation details and prompts are in Appendices A and C.

The free-policy prompt and search interface are held fixed for P_{pre} collection, training inputs, and post-training rollouts, so comparisons among P_{pre}, SFT, and Preference Optimization reflect learned routing changes rather than instruction changes. SFT uses only oracle-eligible exam-

ples: NO_SEARCH targets are successful $N(q)$ responses, and SEARCH targets are first-turn search calls from successful $S(q)$ traces, focusing supervision on the routing action rather than the full post-search trajectory.

Thus, the SEARCH-labeled SFT examples do not require the model to imitate an entire searched answer trajectory. They supervise the decision to enter the search path, while leaving query refinement, evidence use, and final synthesis outside the main routing objective. This choice is consistent with our evaluation, which scores the first action rather than the quality of all subsequent search-conditioned steps.

For Preference Optimization, each eligible q has a chosen response $r^+(q)$ beginning with the oracle-consistent first action and a rejected response $r^-(q)$ beginning with the opposite action, preferably drawn from an actual P_{pre} mistake and otherwise from the controlled counterfactual trace.

We optimize a DPO objective with chosen-response NLL regularization (Rafailov et al., 2023; Pang et al., 2024):

$$\mathcal{L} = \mathcal{L}_{\text{DPO}} + \alpha\, \mathcal{L}_{\text{NLL}}(r^+).$$

Preference Optimization is initialized from the selected SFT checkpoint and uses that checkpoint as the reference model.

3. Experimental Setup

Data and evaluation. We instantiate the counterfactual routing protocol on two complementary question regimes. PopQA provides answerable entity-centric factual questions, for which search may recover long-tail facts; we use 13,763 deduplicated and processed PopQA examples (Mallen et al., 2023). KUQ provides search-boundary questions for which search may or may not be useful: some cases may benefit from external evidence, while others are better resolved through premise correction, clarification, or abstention (Amayuelas et al., 2024). We use the False Assumption category, where the model should reject or correct a false premise, and the Ambiguous category, where the model should recognize missing context and ask for clarification or abstain. The KUQ subset contains 774 False Assumption questions and 850 Ambiguous questions.

Importantly, these dataset groups are not treated as fixed routing labels. PopQA questions are not assumed to require search: if the no-search trace answers correctly, the instance is labeled NO_SEARCH. Likewise, KUQ false-assumption or ambiguous questions are not assumed to forbid search: if no-search fails but forced search enables the model to resolve the premise or ambiguity, the instance is labeled SEARCH. The mixed benchmark therefore tests instance-level routing rather than dataset-level classification.

PopQA is evaluated with a PopQA-adapted SimpleQA-style factuality judge that performs alias-aware correctness checking (Wei et al., 2024), while KUQ is evaluated by category-specific resolution adequacy. Evaluator settings and binary success mappings are provided in Appendix A; full judge prompts are provided in Appendix C.

Models and search environment. We evaluate Qwen3.5-4B (Qwen Team, 2026) and Gemma E2B (Google DeepMind, 2026), comparing P_{pre}, SFT, and Preference Optimization on the test split. P_{pre} is the base-model rollout under the same selective-search prompt and `search(query)` interface used for post-training evaluation. The interface is backed by Brave LLM Context and returns source-backed snippets. We fix the backend, interface, and routing instruction to isolate first-action search routing; budgets and forced-search drop handling are in Appendix A.

Trace collection. For each question, we collect the three traces defined in Section 2. The no-search trace disables tools; the forced-search trace uses prompt-level forcing and acceptance filtering for a valid first-turn search call; and the free-policy trace uses the fixed selective-search prompt without first-action forcing. Retry/drop handling and prompt templates are in Appendices A and C.

Training and metrics. We split data into train, dev, and test before training. SFT and Preference Optimization use only oracle-eligible train rows, formatted with the same free-policy prompt used for P_{pre} and post-training evaluation. UNSOLVED rows are excluded from training and routing metrics but retained for diagnostics. Dev selects checkpoints; test is used only for final reporting. Because oracle labels are model-specific, comparisons are primarily within model across policies rather than direct cross-model rankings.

On oracle-eligible examples, we report routing accuracy and macro-F1, the unweighted average of SEARCH and NO_SEARCH F1. Directional error rates use SEARCH as the positive class, so false positives are unnecessary search and false negatives are missed search. Over-search, or unnecessary-search rate, is

$$\text{OverSearch} = \frac{FP}{TN + FP},$$

the false-positive rate conditioned on oracle NO_SEARCH examples. Under-search, or missed-search rate, is

$$\text{UnderSearch} = \frac{FN}{FN + TP},$$

the false-negative rate conditioned on oracle SEARCH examples.

For category-level reporting, accuracy is computed within each dataset/category on oracle-eligible examples only; FA Acc. and Amb. Acc. are KUQ routing accuracies, not

final-response accuracies. Split sizes, conditional-error denominators, and hyperparameters are in Appendix A.

4. Results

This section answers the two research questions. We first compare the counterfactual outcomes of no-search and forced-search traces, showing that the effect of search depends strongly on the question type and instance. We then analyze the under-search and over-search errors exhibited by the base free-policy execution P_{pre}, and evaluate whether SFT and Preference Optimization improve first-action search routing.

4.1. Counterfactual Outcomes Show Instance-Dependent Search Utility

We first compare task success between controlled no-search and forced-search traces. Table 3 shows that search has opposite effects across question groups. In PopQA, forced search achieves substantially higher success rates than no search for both models. Qwen3.5-4B improves from 0.1831 under no search to 0.5096 under forced search, while Gemma E2B improves from 0.1504 to 0.4813. PopQA therefore supplies many cases in which search is beneficial: factual failures that are not resolved without search can often be recovered when the model sees external evidence.

By contrast, for KUQ False Assumption and Ambiguous questions, forced search yields lower success rates than no search. Qwen3.5-4B decreases from 0.6441 to 0.5376, and Gemma E2B decreases from 0.6022 to 0.4483. For these questions, no-search behaviors such as correcting a false premise, requesting missing information, or abstaining may satisfy the task-specific success criterion more reliably than searching for additional evidence.

The oracle-bucket distributions in Tables A.5 and A.6 (Appendix A) further show that search utility varies by instance across both PopQA and KUQ search-boundary questions.

4.2. SFT and Preference Optimization Improve Search Routing

Table 2 compares the search-routing performance of P_{pre}, SFT, and Preference Optimization on the test split. Evaluation is conducted on oracle-eligible test examples, i.e., examples whose oracle label is either `NO_SEARCH` or `SEARCH`. We report routing macro-F1, routing accuracy, conditional over-search, conditional under-search, and subset routing accuracies for KUQ False Assumption and Ambiguous questions.

Because P_{pre}, SFT, and Preference Optimization are all evaluated under the same selective-search prompt and search interface, these comparisons do not conflate training gains with prompt changes. Moreover, P_{pre} is a prompt-only selective-search baseline rather than an unconstrained tool-available rollout: the base model is already instructed to use search only when it would materially help and to avoid unnecessary tool use. The remaining P_{pre} errors therefore reflect failures to apply this selective-use instruction at the instance level.

SFT substantially improves routing performance for both models beyond this prompt-only baseline. Gemma E2B improves from a P_{pre} macro-F1 of 0.7082 and routing accuracy of 0.7574 to 0.8207 and 0.8292 after SFT. Qwen3.5-4B similarly improves from a P_{pre} macro-F1 of 0.7053 and routing accuracy of 0.7148 to 0.8207 and 0.8263 after SFT. The counterfactual oracle therefore provides a usable learning signal for sharpening the first-action routing boundary beyond prompt-only selective search.

Preference Optimization further shifts the routing boundary after SFT. For Qwen3.5-4B, macro-F1 increases from 0.8207 to 0.8365 relative to SFT, and routing accuracy increases from 0.8263 to 0.8427. The conditional under-search rate decreases from 0.1339 to 0.1034, further reducing cases where the model omits search for questions that require it. For Gemma E2B, the aggregate gain is smaller, but macro-F1 increases from 0.8207 to 0.8235 and routing accuracy from 0.8292 to 0.8329. Preference Optimization also improves the KUQ search-boundary subsets, especially Ambiguous, where accuracy increases from 0.8868 to 0.9434.

4.3. Error Directions Differ Across Model Families

The error directions in Table 2 show that the two models have different initial search-routing tendencies. Gemma E2B exhibits pronounced over-search on `NO_SEARCH` oracle rows under P_{pre}. SFT greatly reduces this over-search, although it introduces a higher under-search rate. Preference Optimization does not completely remove this trade-off, but it slightly reduces under-search and yields additional gains on the KUQ subsets, especially Ambiguous.

By contrast, Qwen3.5-4B exhibits both over-search and under-search before training, and SFT mitigates both error types. Preference Optimization further reduces under-search while keeping over-search nearly unchanged, suggesting that it makes the policy more willing to search on `SEARCH`-oracle cases without substantially increasing unnecessary search on `NO_SEARCH`-oracle cases.

A single global search-rate target would fail for both models. For a search-heavy model such as Gemma E2B, learning no-search restraint is important; for a model such as Qwen3.5-4B, which also misses questions requiring search, repairing missed search is important. Search-routing training should therefore not simply increase or decrease the search-call rate globally. Instead, it should jointly correct under-search

Table 2. Main search-routing results on oracle-eligible test examples. Over-search and under-search are conditional error rates.

Model	Policy	Macro-F1	Routing Acc.	Cond. Over-search	Cond. Under-search	FA Acc.	Amb. Acc.
Gemma E2B	P_{pre}	0.7082	0.7574	0.5425	**0.0598**	0.6897	0.7170
Gemma E2B	SFT	0.8207	0.8292	**0.1928**	0.1574	0.7759	0.8868
Gemma E2B	Preference Opt.	**0.8235**	**0.8329**	0.2059	0.1434	**0.8103**	**0.9434**
Qwen3.5-4B	P_{pre}	0.7053	0.7148	0.3649	0.2272	0.8732	0.6531
Qwen3.5-4B	SFT	0.8207	0.8263	**0.2284**	0.1339	**0.9296**	0.8776
Qwen3.5-4B	Preference Opt.	**0.8365**	**0.8427**	0.2312	**0.1034**	**0.9296**	**0.9184**

Table 3. Counterfactual search outcomes. Task success under no-search and forced-search conditions.

Model	Question group	No-search	Forced-search
Qwen3.5-4B	PopQA	0.1831	0.5096
Gemma E2B	PopQA	0.1504	0.4813
Qwen3.5-4B	KUQ FA + Amb.	0.6441	0.5376
Gemma E2B	KUQ FA + Amb.	0.6022	0.4483

Table 4. Exact-only rescue signatures on `NO_SO` hard subsets. Each column reports examples solved only under that diagnostic probe while the other three probes fail. Examples rescued by multiple probes are excluded. Entries show rescue rate within each model's hard subset.

Model	M_N only	M_S only	R_{TR} only	P_{POST} only
Qwen3.5-4B	0.0202	0.0527	0.0299	0.0255
Gemma E2B	0.0180	0.0445	0.0156	0.0211

and over-search according to each model's initial routing tendency.

5. Diagnostic Analysis: Heterogeneous Failures in `UNSOLVED` Cases

The oracle intentionally excludes `UNSOLVED` examples from routing training and routing evaluation. These are questions for which both the no-search and forced-search traces fail, $y_N(q) = 0$ and $y_S(q) = 0$. Such failures do not identify a reliable first-action target. A failed no-search trace may reflect insufficient parametric knowledge or generation capacity, while a failed forced-search trace may reflect query formulation, retrieval coverage, evidence use, answer synthesis, or model capacity (Krishna et al., 2025). We therefore treat `UNSOLVED` as a diagnostic category rather than as either a `NO_SEARCH` or `SEARCH` routing label.

To focus on persistent controlled-trace failures, we additionally evaluate the base selective-search free-policy response $P_{\text{pre}}(q)$. Cases in which $y_N(q) = 0$, $y_S(q) = 0$, but P_{pre} succeeds are assigned to an escape subset. We focus on the hard subset: examples for which the no-search trace, forced-search trace, and base free-policy trace all fail.

On this hard subset, we test four diagnostic probes: a larger model without search M_N, a larger model with forced search M_S, an expanded retrieval-budget condition using the same base model R_{TR}, and the selected post-training policy rollout P_{POST}. For Qwen3.5-4B, the larger diagnostic model used in M_N and M_S is Qwen3.5-9B (Qwen Team, 2026); for Gemma E2B, it is Gemma E4B (Google DeepMind, 2026). These probes are diagnostic signatures rather than independent causal factors. For example, M_S jointly changes model capacity, query formulation, evidence use, and answer synthesis. We therefore examine which probe uniquely rescues each hard case.

Table 4 reports the exact-only rescue signatures. These exclusive rescue patterns show that the hard `NO_SO` subset is not a single failure type. Some cases are recovered only by larger-model no-search inference, suggesting failures associated with parametric knowledge or no-search generation capacity. A larger share is recovered only by larger-model forced search, suggesting failures that require search-conditioned larger-model inference rather than larger no-search inference alone. Expanded retrieval also uniquely recovers a nontrivial share of cases, indicating that retrieval budget or search-context coverage can be the limiting factor even when the base forced-search trace fails. Finally, post-training policy rollout uniquely recovers some hard cases, showing that routing-related behavior can coexist with other sources of failure even within the `NO_SO` region.

These exclusive rescue signatures should not be read as a causal decomposition of mutually exclusive error sources. Instead, they show why collapsing `NO_SO` into either `NO_SEARCH` or `SEARCH` would be misleading: the bucket contains a mixture of model-capacity limitations, search-conditioned synthesis failures, retrieval-budget failures, and policy-related failures. `UNSOLVED` is therefore better treated as a separate diagnostic category for failures beyond reliable first-action routing supervision.

6. Related Work

Adaptive retrieval and selective search. Retrieval augmentation is a standard way to compensate for limitations in LLMs' parametric knowledge (Lewis et al., 2020; Izacard et al., 2023). Prior work on long-tail factual QA shows that language models struggle to recall less frequent entity

knowledge, and that retrieval can substantially improve performance on such questions (Mallen et al., 2023). However, retrieval is not uniformly beneficial: for questions the model can already answer, retrieved context may instead distract the model or degrade the response (Mallen et al., 2023; Wang et al., 2023). Self-knowledge-guided retrieval is similarly motivated by the observation that retrieved knowledge is not always helpful, and proposes selective retrieval based on whether a model appears to know the answer (Wang et al., 2023). Other adaptive retrieval systems trigger retrieval during generation or select among retrieval strategies using confidence, reflection, or question-complexity signals (Jiang et al., 2023; Asai et al., 2024; Jeong et al., 2024). Our work shares the view that search should be used selectively, but differs in how the routing signal is constructed. Rather than predicting search necessity with a separate classifier, heuristic, or self-knowledge prompt, we compare no-search and forced-search task outcomes for the same question and derive instance-level routing supervision from observed success and failure.

Tool calling and when-not-to-call behavior. Tool-use and tool-calling work primarily evaluates whether models can generate the correct tool name, arguments, and executable function calls (Schick et al., 2023; Li et al., 2023; Qin et al., 2024; Zhang et al., 2024; Lu et al., 2025; Patil et al., 2025). BFCL is a representative benchmark that evaluates function-calling accuracy across single-turn, multi-turn, and agentic settings (Patil et al., 2025). When2Call moves closer to the question of when tools should not be called by asking models to choose among tool calling, direct answering, follow-up questioning, and being unable to answer (Ross et al., 2025). However, When2Call is formulated as a multiple-choice benchmark, and its questions are often structured so that direct answering is an inappropriate or hallucination-prone option. By contrast, `NO_SEARCH` in our setting is not defined as a failure behavior. In PopQA, it may correspond to a correct factual answer produced without search; in KUQ-style boundary cases, it may correspond to premise correction, clarification, or abstention. Thus, our work learns a model-specific search-routing boundary from free-form execution traces, based on whether an actual `search(query)` execution recovers task success.

Over-searching and uncertainty-aware no-search behavior. Recent work on over-searching shows that search can improve answer accuracy for answerable queries while reducing abstention accuracy and increasing cost for unanswerable or underspecified queries (Xie et al., 2026). Such work highlights the risks of unnecessary search, especially when retrieved evidence is noisy or when the question should instead be corrected, clarified, or rejected. This also relates to work on aligning assistants to refuse or say "I don't know" on model-specific unknown questions (Cheng et al., 2024). Our work shares this concern, but does not focus only on over-search. We jointly address over-search, where the model calls search for questions where it is unnecessary, and under-search, where the model omits search for questions where search would recover a no-search failure. In addition, we convert observed no-search and forced-search outcomes into supervision for SFT and Preference Optimization, directly training policies to correct bidirectional search-routing errors.

Known-unknown questions and response-level evaluation. Known-unknown question datasets such as KUQ categorize sources of uncertainty, including false assumptions and ambiguous or underspecified questions (Amayuelas et al., 2024). These categories are useful for our setting because they expose cases in which the value of search is itself uncertain. A false-premise or ambiguous question is not automatically a `NO_SEARCH` case: in some instances, external evidence may help the model detect the false premise, identify the missing context, or recover the intended interpretation. In other instances, searching may be unnecessary or even counterproductive, because the appropriate response is to correct the premise, ask for clarification, or abstain rather than to retrieve more evidence. We therefore use these questions as search-boundary cases rather than as fixed no-search examples. Their routing labels are determined by the same counterfactual outcome rule as factual QA: if the no-search trace resolves the question, the instance is labeled `NO_SEARCH`; if no-search fails but forced search succeeds, it is labeled `SEARCH`; and if both fail, it is excluded from routing supervision as `UNSOLVED`.

Our framework also separates response-level evaluation from routing-label construction. We adapt a SimpleQA-style factuality judge to PopQA with alias-aware correctness checking (Wei et al., 2024), while KUQ-style false-premise and ambiguous questions are evaluated with resolution-oriented rubrics that reward appropriate correction, clarification, or abstention when these resolve the user query. Because our labels depend on free-form final responses, this evaluation setup also relates to LLM-as-judge work, although we restrict the judge to task-success grading rather than asking it to infer search helpfulness directly (Liu et al., 2023; Zheng et al., 2023). Each trace-level outcome is then converted into binary task success before routing labels are constructed. This separation allows factual QA and search-boundary QA to use different task-specific success criteria while still sharing the same first-action routing space, `NO_SEARCH` versus `SEARCH`.

7. Limitations

This study has several limitations. First, the oracle should not be interpreted as an absolute search-helpfulness label, but as a trace-conditional routing oracle based on observed counterfactual traces. Whether a question is labeled

NO_SEARCH, SEARCH, or UNSOLVED depends on the success of $N(q)$ and $S(q)$ under the current model, search tool, prompt, evidence budget, and judge configuration. With a better query, a different retrieval backend, a longer tool-use trajectory, or a larger model, the outcome for the same question may change. Therefore, the oracle should be interpreted as search-routing supervision observed under specific model and tool conditions, rather than as an absolute judgment that a question intrinsically requires search.

Second, this study focuses on first-action search routing rather than the full search-augmented answering pipeline. We evaluate and train whether the model calls search as its first action or responds without search, but we do not directly optimize query refinement, multi-hop retrieval, evidence selection, or answer synthesis after search. In real search-augmented systems, several stages of decision-making may follow the first search call, and such multi-step search behavior is beyond the scope of this work (Yao et al., 2023; Jiang et al., 2023; Krishna et al., 2025). The results should therefore be understood as improving the initial routing boundary that determines whether search is used, rather than as solving tool use or search-augmented answering as a whole.

Third, although the UNSOLVED diagnostic separates failures beyond routing, it does not provide full causal attribution. Through the M_N, M_S, R_{TR}, and P_{POST} interventions, we compare whether larger-model no-search capacity, larger-model search-conditioned solving, expanded retrieval evidence, and post-training policy rollout recover some hard cases. However, these interventions do not isolate mutually independent causes. For example, M_S jointly changes model capacity, query formulation, evidence use, and answer synthesis. This analysis should therefore be interpreted as a diagnostic comparison of the additional conditions under which UNSOLVED cases can be recovered, rather than as a causal decomposition of failure sources. More precise failure attribution and pipeline-level optimization are left for future work.

8. Conclusion

We study search routing in search-augmented LLMs: the problem of deciding, for each question, whether to call search as the first action. Rather than asking a judge to label search usefulness directly, we compare no-search and forced-search responses for the same question and construct an outcome-based oracle over NO_SEARCH, SEARCH, and UNSOLVED. This oracle enables both evaluation and learning of search-call decisions from observed task success.

Our results show that search has different effects across question regimes. In factual QA such as PopQA, search often recovers no-search failures caused by limitations in parametric knowledge. In contrast, for questions with false premises or insufficient context, no-search behaviors such as premise correction, clarification, or abstention may be more appropriate than retrieving evidence. Before routing training, both models exhibit search-routing errors in both directions: under-search, where search is omitted for questions that require it, and over-search, where search is called for questions where it is unnecessary. SFT substantially improves routing macro-F1, but the direction of repair differs by model: Gemma primarily learns no-search restraint, while Qwen reduces both unnecessary search and missed search. Preference Optimization further adjusts the remaining routing boundary after SFT.

The UNSOLVED analysis further justifies leaving paired failures outside binary routing supervision. Exclusive rescue patterns show that N0_S0 hard cases contain heterogeneous bottlenecks: some are recovered only by larger no-search inference, some only by larger forced-search inference, some only by expanded retrieval, and some only by the post-training policy rollout. Because most hard cases nevertheless remain unresolved, collapsing N0_S0 into either NO_SEARCH or SEARCH would turn mixed pipeline failures into noisy routing supervision.

Overall, this work does not reduce failures in search-augmented systems to routing errors alone. Instead, it identifies the subset of questions for which the NO_SEARCH/SEARCH boundary can be reliably defined from observed outcomes, and uses that signal to reduce model-specific routing failures. By separating first-action routing errors from broader search-augmented pipeline failures, the framework provides a practical basis for learning when LLMs should search and when they should respond without search.

Impact Statement

This paper studies when language models should use external search tools. Potential societal impacts include improved factuality and reduced unnecessary tool use, but also possible risks from reliance on imperfect retrieval systems and automated judgment. We discuss these limitations and recommend careful evaluation before deployment in high-stakes settings.

Acknowledgments

This work was supported by the National IT Industry Promotion Agency (NIPA) under the Ministry of Science and ICT (MSIT), and the Seoul Creative Economy & Innovation Center.

References

Amayuelas, A., Wong, K., Pan, L., Chen, W., and Wang, W. Y. Knowledge of Knowledge: Exploring Known-Unknowns Uncertainty with Large Language Models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6416–6432, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.383.

Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations*, 2024.

Cheng, Q., Sun, T., Liu, X., Zhang, W., Yin, Z., Li, S., Li, L., He, Z., Chen, K., and Qiu, X. Can AI Assistants Know What They Don't Know? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8184–8202. PMLR, 21–27 Jul 2024.

Google DeepMind. google/gemma-4-E2B-it and google/gemma-4-E4B-it. https://huggingface.co/google/gemma-4-E2B-it and https://huggingface.co/google/gemma-4-E4B-it, 2026. Hugging Face model cards. Accessed: 2026-06-29.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. Atlas: Few-shot Learning with Retrieval Augmented Language Models. *Journal of Machine Learning Research*, 24(251):1–43, 2023.

Jeong, S., Baek, J., Cho, S., Hwang, S. J., and Park, J. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.389.

Jiang, Z., Xu, F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., and Neubig, G. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495.

Krishna, S., Krishna, K., Mohananey, A., Schwarcz, S., Stambler, A., Upadhyay, S., and Faruqui, M. Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4745–4759, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.243.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Li, M., Zhao, Y., Yu, B., Song, F., Li, H., Yu, H., Li, Z., Huang, F., and Li, Y. API-Bank: A Comprehensive Benchmark for Tool-Augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.187.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., and Zhu, C. G-Eval: NLG Evaluation using Gpt-4 with Better Human Alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153.

Lu, J., Holleis, T., Zhang, Y., Aumayer, B., Nan, F., Bai, H., Ma, S., Ma, S., Li, M., Yin, G., Wang, Z., and Pang, R. ToolSandbox: A Stateful, Conversational, Interactive Evaluation Benchmark for LLM Tool Use Capabilities. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.65.

Mallen, A., Asai, A., Zhong, V., Das, R., Khashabi, D., and Hajishirzi, H. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.546.

Pang, R. Y., Yuan, W., Cho, K., He, H., Sukhbaatar, S., and Weston, J. Iterative Reasoning Preference Optimization. In Globerson, A., Mackey, L., Belgrave, D., Fan, A.,

Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 116617–116637. Curran Associates, Inc., 2024. doi: 10.52202/079017-3702.

Patil, S. G., Mao, H., Yan, F., Ji, C. C.-J., Suresh, V., Stoica, I., and Gonzalez, J. E. The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., and Sun, M. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In Kim, B., Yue, Y., Chaudhuri, S., Fragkiadaki, K., Khan, M., and Sun, Y. (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. doi: 10.52202/075280-2338.

Ross, H., Mahabaleshwarkar, A. S., and Suhara, Y. When2Call: When (not) to Call Tools. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3391–3409, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.174.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. doi: 10.52202/075280-2997.

Wang, Y., Li, P., Sun, M., and Liu, Y. Self-Knowledge Guided Retrieval Augmentation for Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10303–10315, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.691.

Wei, J., Nguyen, K., Chung, H. W., Jiao, Y. J., Papay, S., Glaese, A., Schulman, J., and Fedus, W. Measuring short-form factuality in large language models, November 2024. URL https://arxiv.org/abs/2411.04368v1.

Xie, R., Gopinath, D., Qiu, D., Lin, D., Sun, H., Potdar, S., and Dhingra, B. Over-Searching in Search-Augmented Large Language Models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7714–7739, Rabat, Morocco, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.eacl-long.361.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023.

Zhang, Y., Chen, J., Wang, J., Liu, Y., Yang, C., Shi, C., Zhu, X., Lin, Z., Wan, H., Yang, Y., Sakai, T., Feng, T., and Yamana, H. ToolBeHonest: A Multi-level Hallucination Diagnostic Benchmark for Tool-Augmented Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11388–11422, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.637.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023.

A. Additional Experimental Details

This appendix provides additional implementation details for evaluation, trace collection, training, and routing metrics.

A.1. Evaluators and Success Criteria

The main text describes the three question groups and their task-specific success criteria. Table A.1 summarizes the dataset groups and binary success criteria used to convert final responses into binary success labels.

Table A.1. Dataset groups and binary success criteria.

Group	Rows	Binary success criterion
PopQA	13,763	The final response is factually correct under the gold alias set.
KUQ False Assumption	774	The response rejects or corrects the false premise.
KUQ Ambiguous	850	The response recognizes missing context, asks for clarification, or avoids unsupported commitment.

PopQA is evaluated with an alias-aware SimpleQA-style factuality judge (Mallen et al., 2023; Wei et al., 2024). The judge receives the question, the model response, and the set of acceptable answer aliases. It returns `CORRECT`, `INCORRECT`, or `NOT_ATTEMPTED`; only `CORRECT` maps to success.

KUQ False Assumption and KUQ Ambiguous are evaluated with resolution-oriented judges derived from the KUQ uncertainty categories (Amayuelas et al., 2024). The KUQ judge returns a strict JSON object with a binary `SUCCESS/FAIL` label. Only `SUCCESS` maps to success. Table A.2 lists the judge model, decoding, output-format, and failure-handling settings for both judge families.

Table A.2. Judge settings.

Setting	PopQA	KUQ False Assumption / Ambiguous
Judge model	*gpt*-5.4	*gpt*-5.4
Reasoning setting	`reasoning.effort = none`	`reasoning.effort = none`
Temperature	0.0	0.0
Max output tokens	512	256
Output format	A/B/C classification	Strict JSON
Failure handling	No-match maps to `NOT_ATTEMPTED`.	Strict JSON extraction and schema validation; request or parser exceptions are retried.

A.2. Search Tool, Budget, and Trace Collection

All experiments use a single model-facing `search(query)` tool backed by Brave LLM Context. The model receives source-backed snippets. We do not expose URL clicking, scrolling, a separate fetch function, or a multi-tool browser interface. The backend-generated direct answer is not used.

Table A.3. Default search and generation budget.

Setting	Value
Brave result count	10
Maximum URLs	5
Maximum evidence tokens	2,048
Maximum tool rounds	4
Maximum snippets	10
Maximum tokens per URL	512
Maximum snippets per URL	2

The no-search trace disables tool access. The forced-search trace uses prompt-level forcing and acceptance filtering: the model must produce a valid first-turn `search(query)` call, search is executed, and the model then produces a final response. Forced search is not runtime-constrained decoding. Invalid first-turn tool calls are retried up to two times and then excluded as invalid forced-search traces.

Rows that begin a valid search loop but exceed the maximum tool-round limit are counted as search-path failures, i.e., $y_S(q) = 0$. Table A.4 reports the resulting invalid first-tool-call drops and max-tool-round failures.

Table A.4. Forced-search drop handling.

Slice	Invalid first-tool-call drops	Max-tool-round failures
PopQA, Qwen3.5-4B	0	0
PopQA, Gemma E2B	0	0
KUQ False Assumption / Ambiguous, Qwen3.5-4B	0	9
KUQ False Assumption / Ambiguous, Gemma E2B	0	0

A.3. Oracle-bucket distributions

Table A.5 reports the PopQA four-way partition induced by paired no-search and forced-search outcomes for each model. These counts are consistent with the PopQA marginals reported in the main text: for each model, no-search success is $N1_S1 + N1_S0$ and forced-search success is $N1_S1 + N0_S1$.

Table A.5. PopQA oracle-bucket distribution from paired no-search and forced-search outcomes over 13,763 deduplicated and processed PopQA examples.

Model	$N1_S1$	$N1_S0$	$N0_S1$	$N0_S0$
Qwen3.5-4B	2,222	298	4,791	6,452
Gemma E2B	1,736	334	4,888	6,805

Table A.6 reports the corresponding distribution for the KUQ search-boundary subset. This subset contains false-assumption and ambiguous-context questions, and is included in the mixed train/dev/test splits used for the main routing evaluation. These counts are likewise consistent with the KUQ marginals reported in the main text: for each model, no-search success is $N1_S1 + N1_S0$ and forced-search success is $N1_S1 + N0_S1$.

Table A.6. KUQ search-boundary oracle-bucket distribution from paired no-search and forced-search outcomes over 1,624 false-assumption and ambiguous-context questions.

Model	Slice	$N1_S1$	$N1_S0$	$N0_S1$	$N0_S0$
Qwen3.5-4B	False Assumption	534	104	55	81
Qwen3.5-4B	Ambiguous Context	221	187	63	379
Gemma E2B	False Assumption	325	169	74	206
Gemma E2B	Ambiguous Context	296	188	33	333

A.4. Training Denominators and Hyperparameters

Training uses only oracle-eligible rows, i.e., rows labeled `NO_SEARCH` or `SEARCH`. `UNSOLVED` rows are excluded from SFT, Preference Optimization, and routing-accuracy computation.

For `NO_SEARCH`, SFT uses the successful no-search response as the target. For `SEARCH`, SFT uses the first-turn search call from a successful forced-search trace. Preference Optimization uses the oracle-consistent behavior as the chosen response and an opposite-action response as the rejected response, preferring actual pre-policy mistakes when available. For parameter-efficient tuning, all SFT and Preference Optimization runs update LoRA adapters rather than the full model weights (Hu et al., 2022). We use LoRA adapters with rank $r = 16$, LoRA alpha 32, dropout 0.05, and no bias terms, applied to the attention projections `q_proj`, `k_proj`, `v_proj`, `o_proj` and the MLP projections `gate_proj`, `up_proj`, and `down_proj`. All training runs were conducted on NVIDIA RTX PRO 6000 Blackwell GPUs.

For the test split in Table A.7, the conditional over-search rate is normalized by the test `NO_SEARCH` count, and the conditional under-search rate is normalized by the test `SEARCH` count. Thus, the main-text test denominators are 306 and 502 for Gemma E2B, and 359 and 493 for Qwen3.5-4B.

Table A.8 reports the main SFT and Preference Optimization hyperparameters and selected checkpoints.

Table A.7. Split sizes and oracle eligibility. "Used" denotes rows with oracle label NO_SEARCH or SEARCH.

Model	Split	NO_SEARCH	SEARCH	UNSOLVED	Used
Gemma E2B	Train	2,438	3,995	5,874	6,433
Gemma E2B	Dev	304	498	733	802
Gemma E2B	Test	306	502	737	808
Qwen3.5-4B	Train	2,852	3,926	5,528	6,778
Qwen3.5-4B	Dev	355	490	690	845
Qwen3.5-4B	Test	359	493	694	852

Table A.8. Main training settings.

Model	Stage	LR	β	α	Training length	Batch	Selected checkpoint
Qwen3.5-4B	SFT	2×10^{-5}	n/a	n/a	3 epochs	8	400
Qwen3.5-4B	Preference Opt.	5×10^{-7}	0.03	0.1	1 epoch	8	450
Gemma E2B	SFT	2×10^{-5}	n/a	n/a	3 epochs	8	800
Gemma E2B	Preference Opt.	2×10^{-6}	0.05	0.05	1 epoch	8	100

For DPO, both models use $\texttt{max_prompt_length} = 512$ and $\texttt{max_completion_length} = 1024$. Preference Optimization is initialized from the selected SFT checkpoint and uses the same checkpoint as the reference model. Checkpoints are selected by dev routing behavior and then evaluated on the held-out test split.

B. Diagnostic Details for UNSOLVED Cases

This appendix provides aggregate any-rescue and persistent rates on the UNSOLVED hard subset, together with the additional definitions and breakdowns needed to interpret that diagnostic.

B.1. Hard Subset Construction

The raw UNSOLVED subset consists of rows where both controlled traces fail. We further divide this subset using the base free-policy final response $P_{\text{pre}}(q)$. If P_{pre} succeeds, the row is assigned to the escape subset. If P_{pre} also fails, the row is assigned to the hard subset. Table B.1 reports the resulting raw, escape, and hard counts.

Table B.1. Raw UNSOLVED, escape, and hard subsets.

Model	Raw ($N0_S0$)	Escape	Hard
Qwen3.5-4B	6,912	516	6,396
Gemma E2B	7,344	438	6,906

The hard subset is the target of the diagnostic analysis.

B.2. Overall Rescue on the Hard Subset

Table B.2 reports the aggregate any-rescue diagnostic with counts, complementing the exact rescue-signature breakdown in the main text. Here, *Any rescue* means rescue by at least one of M_N, M_S, R_{TR}, or P_{POST}, and *Persistent* means rescue by none of the four probes.

B.3. Diagnostic Interventions

Table B.3 defines the four diagnostic probes used in the hard-subset analysis.

For Qwen3.5-4B, the larger diagnostic model is Qwen3.5-9B (Qwen Team, 2026). For Gemma E2B, the larger diagnostic model is Gemma E4B (Google DeepMind, 2026).

Relative to the default search budget in Table A.3, the R_{TR} condition is designed to widen retrieval coverage while holding the base model fixed. It adds query rewrite/decomposition and allows up to 8 tool rounds; it also increases Brave result count from 10 to 20, maximum URLs from 5 to 8, maximum evidence tokens from 2,048 to 3,072, and maximum snippets from

Table B.2. Overall rescue on hard subsets across all four diagnostic probes.

Model	Hard (n)	Any rescue	Persistent
Qwen3.5-4B	6,396	1,212 (0.1895)	5,184 (0.8105)
Gemma E2B	6,906	1,003 (0.1452)	5,903 (0.8548)

Table B.3. Diagnostic interventions.

Intervention	Definition
M_N	Larger model, no search.
M_S	Larger model, forced search.
R_{TR}	Same base model with expanded retrieval budget.
P_{POST}	Selected post-training policy rollout.

10 to 12, while keeping the per-URL limits unchanged at 512 tokens and 2 snippets per URL. This makes the diagnostic intent explicit: if R_{TR} succeeds where the default forced-search trace fails, the failure is more plausibly due to insufficient retrieval breadth or evidence budget than to the first-action routing decision itself.

These interventions are operational diagnostics, not independent causal factors. For example, M_S changes model capacity, query formulation, evidence use, and answer synthesis together.

B.4. Source-Level Recovery

Table B.4 breaks the hard-subset recovery outcomes down by source slice. Here, *Any rescue* again means rescue by at least one of the four probes, and *Persistent* means rescue by none.

Table B.4. Source-level recovery on hard subsets.

Model hard set	Slice	(n)	Any rescue	Persistent
Qwen3.5-4B	KUQ	431	97 (0.2251)	334 (0.7749)
Qwen3.5-4B	PopQA	5,965	1,115 (0.1869)	4,850 (0.8131)
Gemma E2B	KUQ	465	169 (0.3634)	296 (0.6366)
Gemma E2B	PopQA	6,441	834 (0.1295)	5,607 (0.8705)

KUQ hard cases are more often recovered than PopQA hard cases, especially for Gemma. This suggests that some KUQ failures are recoverable through improved correction, clarification, or abstention behavior, whereas many PopQA hard cases remain limited by factual retrieval, evidence use, or answer synthesis. These trends are diagnostic rather than causal.

C. Prompt Templates

This appendix reports the prompt families used for no-search, forced-search, free-policy, and judging. Prompt metadata, internal notes, and implementation-only overrides are omitted unless they affect the paper-facing experimental condition. To keep the appendix readable while preserving the full prompt text, we render each prompt family as a compact card and include it inline. Figures C.1, C.2, C.3, C.4, C.5, C.6, C.7, and C.8 provide the prompt cards included in this appendix.

C.1. No-search prompts

C.1.1. PopQA no-search prompt

```
System:
You are answering a closed-book factual question using only your internal
knowledge.
Answer directly with your single best factual answer.
Provide a specific short noun phrase or one short sentence, not just a vague label.
Do not ask for clarification, do not request more context, and do not mention
uncertainty or limitations.
If the name could refer to multiple entities, answer for the most likely public
entity.
User:
{question}
```

Figure C.1. Prompt card for the PopQA no-search condition.

C.1.2. KUQ no-search prompt

```
System:
You are answering a question without using any external tools.
If you know the answer or can responsibly answer from your internal knowledge,
answer directly and briefly.
If the question has a false premise, explicitly correct the premise instead of
answering it as stated.
If the question cannot be answered as written, briefly explain why instead of
forcing an answer.
If the question is underspecified and requires missing context, ask a brief
clarification question or state the missing context.
Do not fabricate facts. Do not mention hidden policies or internal confidence.
User:
{question}
```

Figure C.2. Prompt card for the KUQ no-search condition.

C.2. Forced-search prompts

C.2.1. KUQ FORCED-SEARCH PROMPT

```
System:
You are collecting a search-assisted answer trace with access to a single search
tool.
On the first assistant turn, you must call the search tool before giving any final
answer.
Do not answer the question directly on the first turn.
After receiving search evidence, do the most appropriate thing:
answer directly if the evidence supports an answer,
correct a false premise if the question is based on one,
say briefly that the question cannot be answered as written if the evidence does
not resolve it,
or ask a brief clarification question if required context is still missing.
Do not fabricate facts.
User:
{question}
```

Figure C.3. Prompt card for the KUQ forced-search condition.

C.2.2. POPQA FORCED-SEARCH PROMPT

```
System:
You are collecting a factual QA search trace with access to a single search tool.
On the first assistant turn, you must call the search tool before giving any final
answer.
Do not answer the question directly on the first turn.
After receiving search evidence, answer the user's question directly and concisely.
User:
{question}
```

Figure C.4. Prompt card for the PopQA forced-search condition.

C.3. Free-policy prompts

C.3.1. FREE-POLICY SEARCH-ROUTING PROMPT

```
System:
You are a QA assistant with access to a single search tool.
Use search only when it would materially help produce a better final response.
If you already know the answer or can responsibly respond without search, do not
use the tool.
If the question has a false premise, is fundamentally unknowable, future-dependent,
unsolved, or cannot be resolved without missing user context, do not use search
unless search would clearly help verify or clarify the issue.
If search would help answer, verify, correct a premise, or resolve ambiguity, call
the search tool first.
Avoid unnecessary tool use. Do not mention internal confidence; either respond
without search or use the tool.
User:
{question}
```

Figure C.5. Prompt card for the free-policy search-routing condition.

C.4. Judge prompt text

C.4.1. PopQA judge prompt

The PopQA judge uses the SimpleQA-style grader template (Wei et al., 2024) with the following PopQA-specific alias-aware suffix. The complete grader is a three-way classification into CORRECT, INCORRECT, and NOT_ATTEMPTED.

```
Additional note for this new example:
- The gold reference is an alias set for one answer, not multiple facts that all
  must appear.
- Grade CORRECT if the predicted answer semantically matches any one gold alias and
  does not add a contradiction.
- Grade INCORRECT if the predicted answer commits to a wrong answer, or includes a
  correct alias plus contradictory wrong information.
- Grade NOT_ATTEMPTED if the predicted answer stays too vague, refuses, or does not
  commit to a verifiable answer.
Here is a new example. Simply reply with either CORRECT, INCORRECT, NOT_ATTEMPTED.
Don't apologize or correct yourself if there was a mistake; we are just trying to
grade the answer.
Question: {question}
Gold aliases (any one can be correct):
{answers_block}
Predicted answer: {predicted_answer}
Grade the predicted answer of this new question as one of:
A: CORRECT
B: INCORRECT
C: NOT_ATTEMPTED
Just return the letters "A", "B", or "C", with no text around it.
```

Figure C.6. Prompt card for the PopQA judge. Binary success is 1 for CORRECT and 0 otherwise.

C.4.2. KUQ FALSE ASSUMPTION JUDGE PROMPT

```
You are grading one final response to a search-boundary question.
The question may contain a false premise, be underspecified, or still be answerable
after the issue is resolved.
Decide whether the response resolves the question appropriately. Do not treat
abstention as automatically correct.
Return exactly one JSON object with this schema:
{"label":"SUCCESS|FAIL","resolution_type":"ANSWER|ABSTAIN|CORRECT_PREMISE|CLARIFY|O
THER","error_taxonomy":"ANSWERABLE_FACTUAL|FALSE_PREMISE|UNDERSPECIFIED_CONTEXT|MIX
ED|OTHER","reason":"short reason"}
Grading rules:
- SUCCESS if the response appropriately handles the question as written.
- Do not require exact wording match to the gold reference.
- Judge the final response only; ignore whether search was or was not used
  internally.
- A response may succeed by answering, correcting a false premise, asking for
  clarification, or abstaining only when the issue cannot be resolved.
- If a response uses evidence to correct a false premise or resolve missing
  context, it can be SUCCESS.
- If a response accepts a false premise, invents missing context, or over-commits
  to one unsupported interpretation, label FAIL.
- The reference guidance may be generic; interpret it in the context of the
  question.
- Category guidance: False Premise: the question contains a false or unsupported
  assumption. Success can mean correcting the premise, rejecting the question as
  written, or giving a corrected answer if the response has enough evidence to do
  so.
- For FP questions, use resolution_type CORRECT_PREMISE when the response
  identifies and fixes or rejects the false assumption.
- For UC questions, use resolution_type CLARIFY when the response asks for missing
  context, and ABSTAIN when it correctly says the answer cannot be determined from
  the given information.
- Use CORRECT_PREMISE when it explicitly fixes the false assumption.
- Use CLARIFY when it requests the missing disambiguating context.
- Use ANSWER when the response gives a concrete answer that is adequately supported
  after resolving the issue in the question.
- Use ABSTAIN only when the response appropriately refuses to commit because the
  premise or missing context cannot be resolved from the available information.
- Set error_taxonomy to the dominant issue in the question, not the model's
  behavior.
Category: FP
Question: {question}
Reference guidance: {gold_answer}
Model response: {predicted_answer}
```

Figure C.7. Prompt card for the KUQ False Assumption judge.

C.4.3. KUQ AMBIGUOUS JUDGE PROMPT

The KUQ Ambiguous judge uses the same schema and grading rules as the False Assumption judge, with the category guidance replaced by the prompt card below.

```
Category guidance: Underspecified Context: the question is ambiguous, missing
required context, or underdetermined by the provided constraints. Success can mean
requesting clarification, stating that the information is insufficient, or
resolving the ambiguity only when the response has enough evidence to avoid
committing to a single unsupported interpretation.
Category: UC
Question: {question}
Reference guidance: {gold_answer}
Model response: {predicted_answer}
```

Figure C.8. Prompt card for the KUQ Ambiguous judge.