State-Guided Agentic RAG for Multi-Hop Question Answering

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) improves Large Language Models (LLMs) by grounding generation in external knowledge, while agentic RAG further enables models to iteratively retrieve, reason, and refine their outputs. Despite this progress, existing agentic RAG systems often decide their next actions from implicit LLM reasoning histories, lacking an explicit evidence-grounded state that records what has been resolved, what remains missing, and which action is justified next. This makes the agent vulnerable to unsupported queries, hallucinated intermediate entities, and error propagation, especially when the task requires multi-step knowledge acquisition. We propose a **State-Guided Agentic RAG** framework that treats agentic retrieval as an explicit state-driven process. The framework maintains a structured knowledge state with unresolved slots, dependencies, candidate values, confirmed evidence, and conflicts, and incrementally updates this state during retrieval. The evolving state serves as an action-grounding interface for retrieval, evidence completion, retry, stopping, and answer grounding. We evaluate our method on HotpotQA, 2Wiki, and MuSiQue using Qwen3-14B/8B/4B. Results show that our method achieves the best overall average performance across model scales and improves over standard and agentic RAG baselines in most settings. We provide an anonymized implementation for reproducibility[1].

1 Introduction

Large Language Models (LLMs) (Brown et al., 2020) have demonstrated strong language understanding and generation abilities, but their responses can still suffer from outdated knowledge, factual inconsistency, and hallucination (Kalai et al., 2026). Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020; Gao et al., 2024) mitigates this issue by incorporating external knowledge into generation. As shown in Figure 1(a), a typical RAG system retrieves relevant passages once and uses them as contextual evidence for answer generation. Although this retrieve-then-read paradigm improves factual grounding, retrieval is still largely treated as a static context construction step, leaving the LLM to implicitly decide how the retrieved evidence should be used (Izacard and Grave, 2021; Borgeaud et al., 2022; Izacard et al., 2023).

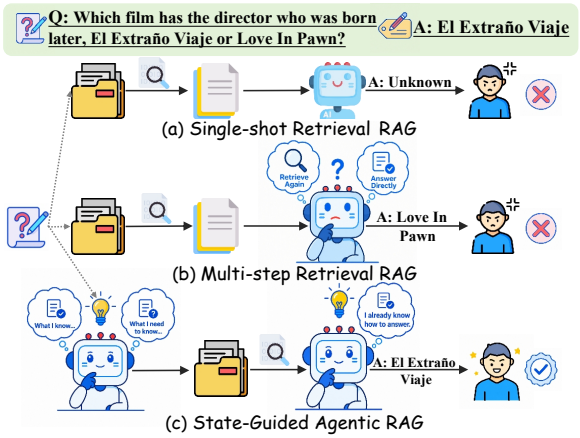

Figure 1: Comparison of single-shot RAG, multi-step retrieval RAG, and the proposed State-Guided Agentic RAG framework. Unlike prior paradigms that rely on retrieved passages or implicit reasoning histories, State-Guided Agentic RAG maintains an explicit slot-level state to track missing, confirmed, and conflicting information and to guide subsequent retrieval actions.

Recent agentic RAG systems extend this paradigm by allowing LLMs to iteratively retrieve, reason, decompose questions, refine queries, and revise answers (Yao et al., 2023; Trivedi et al., 2023; Jiang et al., 2023; Asai et al., 2024; Jeong et al., 2024; Yan et al., 2024; Zhu et al., 2025). As illustrated in Figure 1(b), multi-step retrieval makes RAG more flexible when a single retrieval step is

[1] https://anonymous.4open.science/r/State-Guided-Agentic-RAG-5C3F

insufficient. However, the agent's next action is often determined from the LLM's implicit reasoning history rather than from an explicit, evidence-grounded representation of the current problem-solving state. As a result, the agent may issue unsupported retrieval queries, follow hallucinated intermediate entities, stop prematurely, or continue along an incorrect reasoning path. This limitation is not simply a matter of retrieval depth; it reflects the lack of an explicit basis for controlling agentic actions.

This problem becomes particularly evident in multi-step knowledge-intensive scenarios, such as multi-hop question answering (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022). These tasks require progressively resolving intermediate entities, relations, and evidence dependencies. For example, a question about an attribute of a film director first requires identifying the director and then retrieving evidence about the target attribute. Standard RAG may miss such dependent evidence, while agentic RAG may retrieve repeatedly without knowing what has been confirmed, what remains missing, and which action is justified next.

We argue that agentic RAG needs not only more retrieval or reasoning actions, but an explicit state that grounds and controls those actions. To this end, we propose **State-Guided Agentic RAG**, shown in Figure 1(c), which formulates agentic retrieval as an iterative state resolution process. Our framework maintains a structured knowledge state with unresolved slots, dependencies, candidate values, confirmed evidence, and conflicts, and updates it with retrieved evidence to guide retrieval, evidence completion, retry, stopping, and answer grounding.

The proposed state serves as an action-grounding interface between retrieval and generation. Missing slots trigger focused retrieval, confirmed dependencies support dependency-aware queries, and conflicting evidence triggers retry. As a result, the agent is guided by an inspectable evidence state rather than implicit LLM reasoning alone.

We evaluate our method on HotpotQA, 2Wiki, and MuSiQue (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022) with Qwen3-14B/8B/4B (Yang et al., 2025). Results show strong average improvements over standard and agentic RAG baselines across model scales. Ablations further confirm that explicit state modeling and state-grounded action control are key to the improvements.

Our contributions are:

- We formulate agentic retrieval as a state-guided decision process with explicit action grounding.
- We propose State-Guided Agentic RAG, which uses an evidence-grounded state to control retrieval, retry, stopping, and answer grounding.
- We evaluate across three datasets, three LLM scales, and multiple baselines, with ablation and efficiency analyses.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) improves language models by grounding generation in external knowledge. Early methods such as REALM and RAG introduced retrieval-augmented pre-training or generation with non-parametric memory (Guu et al., 2020; Lewis et al., 2020). Dense Passage Retrieval strengthens open-domain QA with learned dense retrievers (Karpukhin et al., 2020), while FiD improves multi-passage evidence aggregation (Izacard and Grave, 2021). Retrieval-enhanced models such as RETRO and Atlas further show that retrieval can scale language modeling and few-shot knowledge-intensive tasks (Borgeaud et al., 2022; Izacard et al., 2023). Recent work also studies retrieval granularity, such as Mix-of-Granularity, which dynamically selects query-specific retrieval granularity and extends it to graph-based retrieval (Zhong et al., 2025). These methods mainly treat retrieval as context construction, whereas our work formulates retrieval as an iterative state resolution process guided by an explicit evidence-grounded state.

2.2 RAG with Chain-of-Thought Reasoning

Another line of work improves LLM reasoning through intermediate steps. Chain-of-Thought prompting elicits step-by-step reasoning (Wei et al., 2022), while Least-to-Most prompting, Decomposed Prompting, and Self-Ask decompose complex questions into subproblems or follow-up questions (Zhou et al., 2023; Khot et al., 2023; Press et al., 2023). ReAct further combines reasoning traces with external actions (Yao et al., 2023). For retrieval-augmented reasoning, IRCoT interleaves retrieval with chain-of-thought reasoning, ITER-RETGEN uses previous generations to guide iterative retrieval, and Q-DREAM improves multi-hop

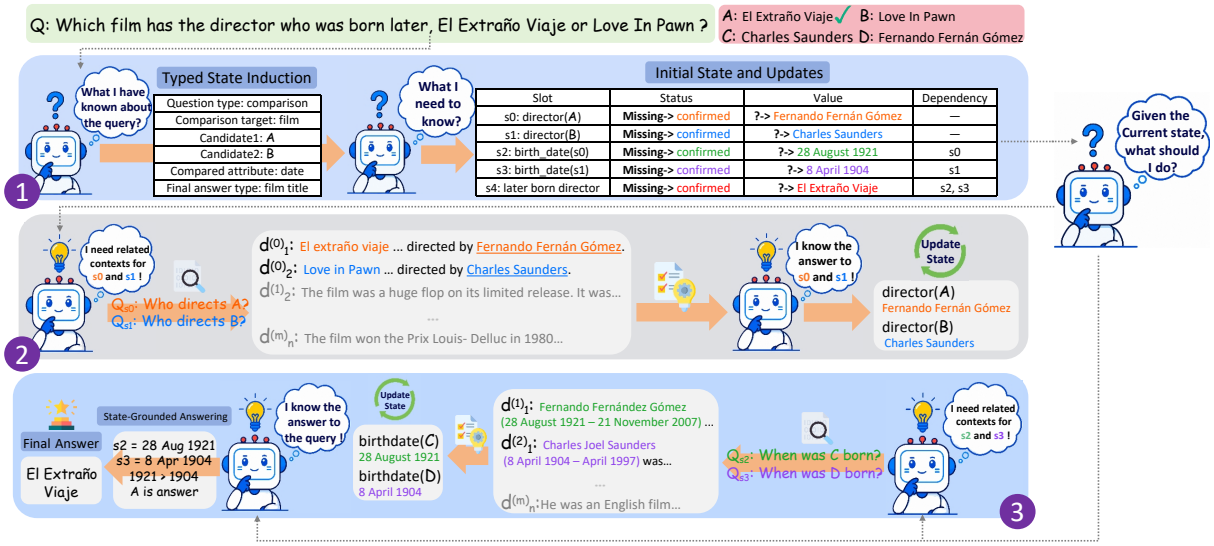

Figure 2: Overview of the proposed State-Guided Agentic RAG framework. Given a multi-hop question, the system first induces a typed slot state, retrieves evidence for unresolved slots, updates slot values with evidence, constructs dependency-aware queries from confirmed slots, and finally generates an answer grounded in the completed state. In the running example, A and B denote the two candidate films, while C and D denote the resolved director entities; functional notations such as `director(A)` and `birthdate(C)` indicate the target attribute to be resolved for the corresponding entity.

RAG through question decomposition and dynamic passage retrieval (Trivedi et al., 2023; Shao et al., 2023). These methods demonstrate the value of intermediate reasoning, but they usually rely on unstructured traces or generated subquestions. In contrast, our method maintains a structured state with unresolved slots, dependencies, candidates, confirmed evidence, and conflicts as the action-grounding interface.

2.3 Agentic RAG

Agentic RAG makes retrieval more adaptive by allowing LLMs to decide when to retrieve, how to reformulate queries, and how to revise outputs. FLARE retrieves when future content is uncertain (Jiang et al., 2023). Self-RAG trains models to retrieve and critique generations with reflection tokens (Asai et al., 2024), while CRAG uses a retrieval evaluator and corrective actions for unreliable evidence (Yan et al., 2024). Adaptive-RAG selects retrieval strategies based on query complexity, and RQ-RAG learns to refine retrieval queries (Jeong et al., 2024; Xu et al., 2024). For multi-hop and structured retrieval, ChainRAG mitigates lost-in-retrieval through progressive rewriting, while RAPTOR and GraphAnchor organize evidence with hierarchical or graph-based structures (Zhu et al., 2025; Sarthi et al., 2024; Liu et al., 2026). Although these methods improve flexibility, many still rely on implicit histories, generated subqueries, reflection signals, or external structures to drive actions. Our work instead focuses on explicit action grounding by using an evidence-grounded state to control retrieval, evidence completion, retry, stopping, and answer grounding.

Compared with prior work, our method differs in three aspects. First, it uses a slot-level state rather than free-form reasoning traces as the main control signal. Second, it continuously updates unresolved dependencies and evidence status during inference instead of decomposing the question only once. Third, the state is an inference-time control structure rather than an offline graph or tree index, making retrieval adaptive and inspectable.

3 Method

3.1 Overview

We propose **State-Guided Agentic RAG**, a retrieval-augmented generation framework that uses an explicit state to ground agentic actions. We define *action grounding* as the property that each retrieval, retry, stopping, or answering decision is triggered by an explicit state condition, such as a missing slot, a confirmed dependency, an unverified candidate, or conflicting evidence.

Given a question q, the system first induces a typed state prior, initializes a slot-level state, re-

Algorithm 1 State-Guided Agentic RAG

Input: question q, retriever R, reranker G, LLM $\mathcal{M}$, maximum steps T
Output: answer y

$\tau \leftarrow$ TypedStateInduction(q)
$S_0 \leftarrow$ InitializeState(q, τ)
$\mathcal{D} \leftarrow \emptyset$
for $t = 1$ **to** T **do**
  if Stop(S_{t-1}) **then**
    break
  end if
  $z_t \leftarrow$ SelectTargetSlot(S_{t-1})
  $a_t \leftarrow$ ConstructQuery(q, z_t, S_{t-1}, τ)
  $D_t \leftarrow G(R(a_t))$
  $\mathcal{D} \leftarrow \mathcal{D} \cup D_t$
  $S_t \leftarrow$ UpdateState(q, S_{t-1}, D_t)
end for
$y \leftarrow$ StateGroundedAnswer$(q, \mathcal{D}, S_t)$
return y

trieves evidence for unresolved slots, updates the state with retrieved evidence, and finally generates an answer from the accumulated evidence and final state. As illustrated in Figure 2, the state makes the intermediate reasoning process inspectable: the system knows what has been resolved, what remains missing, and which evidence supports each intermediate value.

Conventional agentic RAG often selects the next action from an implicit LLM-generated reasoning history:

$$a_t = \mathcal{M}(h_t),$$

where h_t is a textual history. In contrast, our method selects the next action from the current state:

$$a_t = \pi(q, S_{t-1}, \tau),$$

where τ is the typed prior, S_{t-1} is the current state, and π is the state-grounded action policy. Thus, the agent retrieves because a specific state condition requires evidence, rather than because the LLM implicitly decides to continue.

3.2 Typed State and Evidence-Grounded State

Typed State Induction converts the input question into a typed prior:

$$\tau = (c, \mathcal{Z}, \mathcal{D}, \mathcal{H}),$$

where c is the reasoning type, $\mathcal{Z}$ is the slot schema, $\mathcal{D}$ is the dependency schema, and $\mathcal{H}$ contains retrieval hints. This module is implemented as a structured LLM parser with a fixed JSON schema. It identifies the answer target, intermediate information needs, and dependency relations among them. The reasoning type covers common patterns such as bridge, comparison, attribute, temporal, and location questions.

The state at step t is represented as a set of slots:

$$S_t = \{z_1, z_2, \ldots, z_n\}.$$

Each slot is defined as

$$z_i = (d_i, r_i, \mathcal{P}_i, s_i, v_i, \mathcal{E}_i),$$

where d_i is the slot description, r_i is the role, $\mathcal{P}_i$ denotes dependency links, s_i is the status, v_i is the current value, and $\mathcal{E}_i$ stores supporting evidence IDs. The status is one of:

$$s_i \in \{\textit{missing}, \textit{candidate}, \textit{confirmed}, \textit{conflict}\}.$$

A *missing* slot has not been resolved; a *candidate* slot has a plausible but unverified value; a *confirmed* slot is supported by retrieved evidence or derived from confirmed dependency slots; and a *conflict* slot indicates that retrieved evidence supports incompatible values.

In our implementation, the slot schema is fixed after induction. Subsequent inference updates slot values, statuses, and evidence links, but does not dynamically append new slots. This keeps state evolution inspectable and prevents uncontrolled state expansion.

3.3 State Update and Validation

At each step, the retrieved passages update the current state:

$$S_t = \mathcal{U}(q, S_{t-1}, D_t).$$

The update module is LLM-based but constrained by a structured output schema. For each existing slot, it may apply one of four operations: KeepMissing, SetCandidate, Confirm, or MarkConflict. A slot is set as *candidate* when retrieved evidence suggests a plausible value, but the support is incomplete or ambiguous. It is marked as *confirmed* when the value is consistent with the slot description and directly supported by evidence. For derived answer-target slots, such as comparison decisions, confirmation can also be based on confirmed dependency slots. If retrieved passages support incompatible values, the slot is marked as *conflict*.

We validate each update before applying it. Updates referring to nonexistent slot IDs are ignored, malformed JSON outputs trigger a conservative fallback, and evidence IDs are retained only when

they correspond to retrieved passages. Therefore, intermediate values are not treated as confirmed unless they are explicitly linked to evidence or derived from confirmed dependencies.

3.4 State-Grounded Action Policy

The state-grounded action policy selects the next action according to slot status, dependency structure, and evidence support. It follows the priority order:

stop $\rightarrow$ *conflict resolution* $\rightarrow$ *dependency retrieval* $\rightarrow$ *candidate verification* $\rightarrow$ *bridge retrieval*.

The system stops when the answer-target slot and its required dependencies are confirmed. Conflict slots trigger additional retrieval for disambiguation. If a missing slot depends on confirmed slots, the policy constructs a dependency-aware query by inserting the confirmed values into the next retrieval query. Candidate slots trigger verification retrieval, while unresolved bridge slots are retrieved when no higher-priority action applies.

When multiple slots share the same priority, we select the one with the smallest dependency depth and then follow the role order: answer-target, dependent, bridge, and other. For comparison questions, symmetric candidate slots can be retrieved in the same step when possible. This deterministic rule prevents repeated low-priority retrieval and makes the action sequence inspectable.

This policy realizes the main operations of our framework: second-hop retrieval from confirmed bridge values, dependency-aware query construction for unresolved dependent slots, typed retry from unresolved slots and typed priors, and conflict handling through additional evidence or stronger support. All operations are triggered by explicit state conditions rather than implicit reasoning history.

3.5 Stopping and State-Grounded Answer Generation

The required slots are the answer-target slot and all its ancestors in the dependency graph. The system stops when the answer-target slot and its required dependencies are confirmed, when remaining unresolved slots are irrelevant to the answer target, or when the maximum step budget is reached. If unresolved conflict slots remain on the answer path, the system continues until the budget is exhausted.

After k state updates, all retrieved evidence is aggregated:

$$D = D_1 \cup D_2 \cup \cdots \cup D_k.$$

The final answer is generated from the question, accumulated evidence, and final state:

$$y = \mathcal{M}(q, D, S_k).$$

The answer prompt prioritizes confirmed answer-target slots and their supporting evidence. If the generated answer conflicts with confirmed slot values, it is revised according to the confirmed state. If the answer relies only on an unverified candidate, the system selects the best-supported value or outputs the most concise answer justified by the available evidence. The answer rewriting step only converts the output into a concise answer string and does not access gold answers.

4 Experiment

4.1 Setup

Datasets. We evaluate on HotpotQA[2] (Yang et al., 2018), 2Wiki[3] (Ho et al., 2020), and MuSiQue[4] (Trivedi et al., 2022), covering bridge, comparison, temporal, geographic, and multi-document reasoning. Main comparison and ablation experiments use 1,500 development examples per dataset, while efficiency and sensitivity analyses use 500 examples per dataset. Results across these settings should therefore not be directly compared by absolute values.

Baselines. For brevity, we refer to the proposed State-Guided Agentic RAG framework as State-Guided RAG in the following sections. We compare with Direct QA (Yang et al., 2025), NaïveRAG (Lewis et al., 2020), RAPTOR (Sarthi et al., 2024), GraphAnchor (Liu et al., 2026), FLARE (Jiang et al., 2023), IRCoT (Trivedi et al., 2023), Q-DREAM (Ye et al., 2025), and A-RAG (Du et al., 2026), covering no-retrieval, single-shot retrieval, and multi-step retrieval settings. For reproduced baselines, we follow released implementations or public protocols, preserving method-specific retrieval backends while standardizing the Qwen3 backbone family, dataset splits, answer normalization, and EM/F1 evaluation script. More details of the implementation are provided in Appendix A.

[2] https://hotpotqa.github.io/
[3] https://github.com/Alab-NII/2wikimultihop
[4] https://github.com/stonybrooknlp/musique

LLMs	Type	Methods	MuSiQue		2Wiki		HotpotQA		Average	
			F1	EM	F1	EM	F1	EM	F1	EM
Qwen3-14B	*No Retrieval*	Direct	9.82	2.73	31.02	26.92	30.32	21.93	23.72	17.19
	Single-shot Retrieval	NaïveRAG	15.34	8.92	34.22	29.66	53.49	40.65	34.35	26.41
		RAPTOR	39.03	23.07	54.18	39.00	**68.82**	50.47	54.01	37.51
		Q-DREAM	28.14	18.73	60.08	50.47	48.90	36.60	45.71	35.27
	Multi-step Retrieval	GraphAnchor	27.52	17.27	44.73	39.07	59.92	46.20	44.06	34.18
		FLARE	35.12	23.00	49.68	42.13	58.71	45.07	47.84	36.73
		IRCoT	30.20	19.30	41.30	33.30	51.60	39.50	41.03	30.70
		A-RAG	35.74	26.10	50.42	46.90	62.31	52.80	49.49	41.93
	Ours	State-Guided RAG	**43.95**	**33.20**	**64.42**	**55.60**	67.66	**54.00**	**58.68**	**47.60**
Qwen3-8B	*No Retrieval*	Direct	10.75	2.36	28.15	23.47	26.07	17.43	21.66	14.42
	Single-shot Retrieval	NaïveRAG	15.08	7.86	33.51	28.10	49.98	37.76	32.86	24.57
		RAPTOR	36.03	22.00	48.51	36.73	59.40	43.20	47.98	33.98
		Q-DREAM	25.86	16.73	48.26	36.87	37.29	25.00	37.14	26.20
	Multi-step Retrieval	GraphAnchor	26.86	15.27	45.95	40.27	60.63	47.07	44.48	34.20
		FLARE	32.26	20.13	46.29	39.20	53.08	40.93	43.88	33.42
		IRCoT	29.50	17.70	43.90	35.40	50.70	39.30	41.37	30.80
		A-RAG	25.92	17.80	44.32	41.40	54.53	44.50	41.59	34.57
	Ours	State-Guided RAG	**40.42**	**29.20**	**51.00**	**42.40**	**61.07**	**47.20**	**50.83**	**39.60**
Qwen3-4B	*No Retrieval*	Direct	8.88	1.57	24.09	19.97	20.65	13.57	17.87	11.70
	Single-shot Retrieval	NaïveRAG	17.21	8.85	31.10	17.73	48.98	34.78	32.43	20.45
		RAPTOR	29.10	19.80	**45.43**	30.60	55.96	41.30	43.50	30.57
		Q-DREAM	16.96	6.40	29.49	13.93	22.80	10.60	23.08	10.31
	Multi-step Retrieval	GraphAnchor	25.71	15.47	41.11	30.47	57.59	43.40	41.47	29.78
		FLARE	28.40	18.80	41.12	34.80	52.59	40.60	40.70	31.40
		IRCoT	24.00	14.30	43.60	**35.90**	46.30	34.70	37.97	28.30
		A-RAG	19.54	11.80	37.14	34.40	44.43	36.10	33.70	27.43
	Ours	State-Guided RAG	**32.63**	**22.07**	41.65	33.33	**59.85**	**46.67**	**44.71**	**34.02**

Table 1: Comparison results on three multi-hop QA datasets with Qwen3-14B, Qwen3-8B, and Qwen3-4B. The best score in each column is shown in bold, and the second-best score is underlined.

4.2 Main Results

Table 1 compares State-Guided RAG with baselines on three datasets using Qwen3-14B, Qwen3-8B, and Qwen3-4B. Our method achieves the best average performance under all three backbone scales. With Qwen3-14B, it obtains 58.68 average F1 and 47.60 average EM, outperforming the strongest baseline averages by 4.67 F1 and 5.67 EM. The same trend under Qwen3-8B and Qwen3-4B shows that the improvement is not merely due to reader scale.

The gains are most evident on MuSiQue and 2Wiki, where questions often require resolving intermediate entities or relations before retrieving final evidence. With Qwen3-14B, State-Guided RAG improves the best baseline by 4.92 F1 and 7.10 EM on MuSiQue, and by 4.34 F1 and 5.13 EM on 2Wiki. On HotpotQA, RAPTOR remains competitive in F1, but our method still achieves the best EM and the best overall average. These results suggest that graph or tree structures can help on specific datasets, while explicit evidence-grounded state tracking provides more stable gains across datasets and model scales. The weaker results of closed-book QA and NaïveRAG further indicate that parametric knowledge or one-shot retrieval is insufficient for multi-step evidence acquisition; effective agentic RAG requires an explicit state to record what is confirmed, what is missing, and which action should be taken next.

4.3 Effect of Retrieved Evidence TopK

Table 2 studies how many retrieved passages should be provided to each state update step. TopK-K limits the final merged evidence to at most K passages per step, while the default setting uses adaptive state-guided retrieval with a first-hop evidence cap of 3 and a merged evidence limit of 8.

Using only one passage substantially reduces average F1 to 49.94, showing that a single passage is often insufficient for state verification and dependency resolution. Increasing K improves performance, with TopK-5 reaching 59.97 average F1. However, larger values such as 7 or 9 do not bring consistent gains and process more evidence. The default setting achieves the best average F1 of

TOP-K	MuSiQue		2Wiki		HotpotQA		Average			
	EM	F1	EM	F1	EM	F1	EM	F1	Steps	Time/Q
K=1	26.40	34.91	43.00	51.43	50.20	63.48	39.87	49.94	3.75	13.10s
K=2	35.80	45.32	51.00	59.35	54.00	67.42	46.93	57.36	3.52	14.32s
K=3	35.80	45.64	56.20	65.00	**54.60**	67.73	48.87	59.46	3.51	16.43s
K=4	36.40	47.30	54.40	63.18	54.00	66.42	48.27	58.97	3.50	16.46s
K=5	36.80	47.54	56.00	65.24	53.20	67.14	48.67	59.97	3.44	16.39s
K=7	37.60	48.27	52.40	61.96	53.00	65.90	47.67	58.71	3.36	16.67s
K=9	**37.80**	48.75	54.00	63.45	53.80	66.91	48.53	59.70	3.31	16.66s
Default	**37.80**	**48.79**	**57.00**	**66.41**	54.40	**68.46**	**49.73**	**61.22**	3.49	15.85s

Table 2: Retrieval TopK sensitivity with Qwen3-14B on 500 examples per dataset. Default denotes adaptive state-guided retrieval. Time/Q is average inference time per question; best and second-best scores are bolded and underlined.

State-Step Budget	MuSiQue		2Wiki		HotpotQA		Average			
	EM	F1	EM	F1	EM	F1	EM	F1	Steps	Time/Q
Budget=1	10.80	19.89	22.20	27.46	36.40	45.17	23.13	30.84	1.00	8.00s
Budget=2	31.20	41.96	43.20	50.23	53.40	65.46	42.60	52.55	1.94	10.91s
Budget=3	38.00	48.70	53.20	62.54	**55.00**	67.64	48.73	59.63	2.76	13.50s
Budget=4	**39.80**	49.77	**58.00**	**67.36**	53.00	66.43	50.27	61.19	3.49	15.78s
Budget=5	39.60	**50.32**	57.20	66.86	54.80	67.59	**50.53**	**61.59**	4.20	17.90s
Budget=7	38.80	49.92	57.80	66.50	54.60	68.18	50.40	61.53	5.68	22.85s
Budget=9	37.60	48.93	56.60	65.79	**55.00**	67.85	49.73	60.86	7.16	29.38s
Dynamic	37.80	48.79	57.00	66.41	54.40	**68.46**	49.73	61.22	3.49	15.85s

Table 3: State-step budget sensitivity with Qwen3-14B on 500 examples per dataset. Budget=k uses at most k state-update steps, while Dynamic adaptively controls retrieval and reasoning. Time/Q is average inference time per question; best and second-best EM/F1 scores are bolded and underlined.

61.22 with moderate latency, indicating that flexible evidence merging is better than enforcing a fixed evidence budget at every step.

4.4 Effect of State-Step Budget

Table 3 compares fixed state-step budgets with the dynamic setting. A single state update performs poorly, showing that one transition is usually insufficient for multi-hop QA. Performance improves as the budget increases, and Budget=5 achieves the best fixed-budget average performance with 61.59 F1 and 50.53 EM.

The dynamic setting obtains comparable results, with 61.22 average F1 and 49.73 average EM, while using fewer steps than Budget=5 and much lower latency than Budget=7 or Budget=9. It also achieves the best HotpotQA F1, suggesting that adaptive stopping can retain useful reasoning depth without forcing unnecessary updates. Overall, moderate state evolution is most effective, while excessive fixed updates mainly increase computation and may introduce retrieval noise.

4.5 Effect of Conflict Handling

Table 4 evaluates conflict handling by merging conflict slots into candidate slots and disabling conflict-specific retrieval. Removing conflict handling reduces average F1 from 61.22 to 57.04 and EM from 49.73 to 46.53, with larger drops on MuSiQue and 2Wiki. It also increases state steps and latency, suggesting that separating conflict slots helps resolve evidence disagreement more efficiently. Since conflicts occur only in some cases, we view conflict handling mainly as a robustness mechanism rather than the primary source of improvement.

4.6 Effect of Slot Budget

Additional slot-budget analysis is provided in Appendix C.4. The results show that one slot is not enough to represent intermediate entities, dependencies, and answer targets. A moderate slot budget gives the best trade-off between reasoning effectiveness and cost, supporting the design choice that the state should be expressive but not overly fragmented.

Method	MuSiQue		2Wiki		HotpotQA		Average			
	EM	F1	EM	F1	EM	F1	EM	F1	Steps	Time/Q
w/o Conflict Handling	33.60	43.39	52.00	61.28	54.00	66.46	46.53	57.04	3.77	17.91s
State-Guided RAG	37.80	48.79	57.00	66.41	54.40	68.46	49.73	61.22	3.49	15.85s

Table 4: Ablation results of conflict handling with Qwen3-14B on 500 examples per dataset. Time/Q denotes average inference time per question in seconds. The w/o Conflict Handling variant merges conflict slots into candidate slots and disables conflict-specific retrieval.

Methods	MuSiQue		2Wiki		HotpotQA		Average	
	F1	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**
w/o State	18.88	11.47	31.53	27.27	54.29	41.67	34.90	26.80
w/o Typed State Induction	42.66	32.13	63.69	54.87	66.99	53.33	57.95	46.78
w/o State-Grounded Action Policy	39.13	29.67	57.54	48.40	63.76	50.40	53.48	42.80
State only	40.00	30.13	57.17	48.40	63.47	50.67	53.55	43.07
State-Guided RAG	**43.95**	**33.20**	**64.42**	**55.60**	**67.66**	**54.00**	**58.68**	**47.60**

Table 5: Ablation results of State-Guided RAG with Qwen3-14B. We remove the structured state, typed state induction, or state-grounded action policy to assess their contributions; State only retains state tracking without the full retrieval policy.

4.7 Ablation Study

Table 5 reports ablations with Qwen3-14B on 1,500 examples per dataset. *w/o State* removes the structured state, while *State only* keeps the state as auxiliary memory without using it for action selection. *w/o Typed State Induction* removes the typed prior, and *w/o State-Grounded Action Policy* replaces state-conditioned action selection with generic iterative retrieval. All variants use the same backbone, split, index, step budget, and evaluation script.

The results show that explicit state modeling is essential. Compared with *w/o State*, *State only* raises average F1 from 34.90 to 53.55 and EM from 26.80 to 43.07. However, removing the State-Grounded Action Policy reduces F1 from 58.68 to 53.48 and EM from 47.60 to 42.80, showing that the state must actively guide retrieval, retry, and stopping rather than only serve as memory. The drop is especially clear on 2Wiki, where dependency-aware query construction is important.

Removing Typed State Induction causes a smaller drop, from 58.68 to 57.95 average F1, suggesting that typed priors help initialization but are not the main source of gains. Overall, the improvements mainly come from evidence-grounded state tracking and state-conditioned action control. The answer rewriting ablation in Appendix C.1 further shows that the gains are not due to answer-format post-processing.

4.8 Case Study

A detailed case study is provided in Appendix B. The examples show that some baselines retrieve the necessary evidence but still fail due to temporal comparison reversal, noisy collapsed-tree context, or premature stopping. In contrast, State-Guided RAG confirms bridge entities and dependent birthdate slots before performing the final comparison, making the reasoning process more explicit and evidence-grounded.

5 Conclusion

We studied agentic RAG for multi-hop question answering and proposed State-Guided Agentic RAG, which formulates retrieval as an explicit state resolution process. The framework maintains a structured slot state to track missing information, evidence, dependencies, and conflicts, and uses it to guide retrieval, retry, stopping, and answer grounding. Experiments on HotpotQA, 2Wiki, and MuSiQue with Qwen3-14B/8B/4B show strong average improvements over standard and agentic RAG baselines across model scales. Future work will extend the framework to broader retrieval scenarios and strengthen state verification under uncertainty.

6 Limitations

Our framework relies on LLM-based state induction and state update, so incomplete slot schemas,

over-confirmed candidates, or missed conflicts may still affect later retrieval and answer grounding. It also introduces additional retrieval and LLM calls compared with one-shot RAG, leading to higher latency and token cost. Moreover, our experiments focus on English multi-hop QA benchmarks, and the generality of the framework to long-form generation, multilingual QA, domain-specific QA, real-time web retrieval, and multimodal retrieval remains to be further studied.

7 Ethical Considerations and Risks

Ethical Considerations. This work uses public QA benchmarks and local retrieval indexes for research purposes, and does not collect personal data, conduct user profiling, or use private user information. However, retrieved corpora may contain biased, outdated, incorrect, or offensive content, which may affect state updates and final answers. Applications beyond benchmark evaluation should therefore ensure license-compliant data use, corpus auditing, content filtering, and human verification when necessary.

Risks. An explicit state may appear reliable even when intermediate state transitions are incorrect. Wrongly confirmed entities, relations, or evidence dependencies may guide later retrieval toward misleading answers, and incomplete retrieval may lead to answers based on partial evidence. The system should therefore not be used as an autonomous decision-making tool in high-stakes settings without provenance display, abstention mechanisms, stronger evidence verification, and human oversight.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Shaohan Wang, Pengyu Wang, Xiaorui Wang, and Zhendong Mao. 2026. A-rag: Scaling agentic retrieval-augmented generation via hierarchical retrieval interfaces. *Preprint*, arXiv:2602.03442.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*.

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

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. 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*, pages 7029–7043.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *Preprint*, arXiv:2305.06983.

Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. 2026. Evaluating large language models for accuracy incentivizes hallucinations. *Nature*.

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

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*.

Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. 2023. Making large language models a better foundation for dense retrieval. *Preprint*, arXiv:2312.15503.

Zhenghao Liu, Mingyan Wu, Xinze Li, Yukun Yan, Shuo Wang, Cheng Yang, Minghe Yu, Zheni Zeng, and Maosong Sun. 2026. Graph-anchored knowledge indexing for retrieval-augmented generation. *Preprint*, arXiv:2601.16462.

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

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 10014–10037.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Chunpu Xu, Hongyin Luo, Chi-Min Chan, Jie Fu, Yike Guo, Wei Xue, and Ruibin Yuan. 2024. RQ-RAG: Learning to refine queries for retrieval augmented generation. In *Proceedings of the Conference on Language Modeling*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Linhao Ye, Lang Yu, Zhikai Lei, Qin Chen, Jie Zhou, and Liang He. 2025. Optimizing question semantic space for dynamic retrieval-augmented multi-hop question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17814–17824. Association for Computational Linguistics.

Zijie Zhong, Hanwen Liu, Xiaoya Cui, Xiaofan Zhang, and Zengchang Qin. 2025. Mix-of-granularity: Optimize the chunking granularity for retrieval-augmented generation. In *Proceedings of the 31st*

International Conference on Computational Linguistics, pages 5756–5774. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Rongzhi Zhu, Xiangyu Liu, Zequn Sun, Yiwei Wang, and Wei Hu. 2025. Mitigating lost-in-retrieval problems in retrieval augmented multi-hop question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 22362–22375. Association for Computational Linguistics.

A Implementation Details

A.1 Inference Procedure

Our method maintains a structured knowledge state throughout inference. Given a question, the system first induces evidence slots, then repeatedly retrieves evidence, updates slot values, and selects the next action according to the current state. The formal state used for action selection consists of slot descriptions, dependency links, target roles, slot values, evidence IDs, and slot statuses. The implementation additionally logs confidence scores for analysis, but confidence is not used as a formal state variable by the action policy. The major slot statuses are *missing*, *candidate*, *confirmed*, and *conflict*.

Retrieval Setup. For our method, each dataset is indexed with a chunk-level dense semantic FAISS (Douze et al., 2024) index using BGE-M3 embeddings (Chen et al., 2024). Retrieved passages are reranked before being passed to the state update module (Li et al., 2023). In the default setting, the first-hop reranked evidence cap is 3, while the final merged evidence pool may additionally include title-hop or second-hop evidence with a merge limit of 8. In the TopK sensitivity experiment, we explicitly constrain the final merged evidence provided to each state update step to at most K passages. Additional index-storage and retrieval-backend details are provided in Appendix C.5 and Appendix D.

Implementation and Hardware. All experiments are conducted on an offline Linux server with NVIDIA H100 80GB GPUs. We use batch size 1 for inference. Unless otherwise specified, model loading, index loading, and retriever initialization are excluded from per-question inference time. Reported latency includes retrieval, reranking, LLM calls, parsing, and state update for each attempted question.

Metrics. We report Exact Match (EM) and token-level F1. EM measures whether the normalized prediction exactly matches the gold answer, while F1 measures token-level overlap. For efficiency analysis, we report average inference time per question, P90 latency, average state steps, average LLM calls, and tokenizer-estimated token cost. Token cost is computed over all LLM calls: for each question, every prompt input and every model-generated output are tokenized with the Qwen3 tokenizer and summed over the full inference process.

Success and Failure. Success indicates that the inference pipeline completes without parsing, retrieval, or generation errors. Failures include invalid structured outputs that cannot be recovered, runtime errors, or empty final predictions. Failed cases are assigned empty predictions and included in EM/F1 computation. Some baseline logs do not expose LLM-call counts; we mark these values as unavailable and do not use them for strict cross-method comparison.

A.2 Baseline Descriptions

Direct QA. Direct QA directly prompts the Qwen3 model to answer the question without retrieving external evidence (Yang et al., 2025). This baseline measures how much multi-hop QA performance can be obtained from the model's parametric knowledge alone.

NaïveRAG. NaïveRAG follows the standard retrieve-then-read paradigm (Lewis et al., 2020). It retrieves relevant passages once from a dense semantic index and then prompts the reader LLM to generate the final answer from the retrieved context.

RAPTOR. RAPTOR constructs a hierarchical tree of text chunks through recursive clustering and summarization (Sarthi et al., 2024). During inference, it retrieves from the tree-structured representation, which allows both fine-grained chunks and higher-level summaries to be used as evidence.

GraphAnchor. GraphAnchor organizes retrieved evidence with graph-anchored knowledge indexing (Liu et al., 2026). It uses graph-structured

evidence organization to support retrieval and reasoning over related entities and passages.

FLARE. FLARE performs active retrieval during generation by detecting uncertain future content and using it to trigger retrieval (Jiang et al., 2023). Unlike one-shot RAG, it can retrieve additional evidence while the answer is being generated.

IRCoT. IRCoT interleaves chain-of-thought reasoning with retrieval for knowledge-intensive multi-step questions (Trivedi et al., 2023). It repeatedly generates intermediate reasoning steps and uses them to retrieve new evidence for subsequent reasoning.

Q-DREAM. Q-DREAM optimizes the question semantic space for dynamic retrieval-augmented multi-hop QA (Ye et al., 2025). It decomposes complex questions into dependent subquestions and retrieves evidence according to the evolving subquestion structure.

A-RAG. A-RAG scales agentic retrieval-augmented generation through hierarchical retrieval interfaces (Du et al., 2026). It uses an agentic retrieval process to plan and access evidence across different retrieval levels before producing the final answer.

B Case Study

This section presents a controlled case study on the same multi-hop comparison question across State-Guided RAG and several representative baselines. The goal is not only to compare final predictions, but also to analyze how different methods retrieve evidence, resolve intermediate dependencies, decide whether additional retrieval is needed, and generate the final answer. State-Guided RAG explicitly induces typed evidence slots, confirms bridge entities, retrieves dependent birth-date evidence, and grounds the final comparison in the final evidence-grounded state. In contrast, the baseline traces reveal different failure modes: Q-DREAM retrieves all necessary evidence but reverses the final temporal comparison; RAPTOR retrieves sufficient facts but suffers from noisy collapsed-tree context during answer selection; and GraphAnchor obtains the correct passage-grounded answer while its graph-only branch and stopping judgement expose weaknesses in state sufficiency estimation. Together, these examples illustrate why explicit state tracking and state-grounded action control are important for reliable multi-hop retrieval and answering.

Case Study: State-Guided Reasoning Trace

Question: Which film has the director who was born later, *El Extraño Viaje* or *Love In Pawn*?

Ground Truth: El Extraño Viaje

Final Answer: El Extraño Viaje

State-Guided Reasoning Trace

1. Typed State Induction. The system identifies this as a comparison question over two candidate films and constructs the following state slots.

s0: director of *El Extraño Viaje*
s1: director of *Love In Pawn*
s2: birth date of s0
s3: birth date of s1
s4: film whose director was born later

2. Bridge Retrieval. The system retrieves evidence for the two bridge questions:

Q1: Who directed *El Extraño Viaje*?
Q2: Who directed *Love In Pawn*?

State update:

s0 → **confirmed = Fernando Fernán Gómez**
s1 → **confirmed = Charles Saunders**

3. Dependency-Aware Retrieval. After the directors are confirmed, the system retrieves evidence for the dependent birth-date slots:

Q3: When was Fernando Fernán Gómez born?
Q4: When was Charles Saunders born?

State update:

s2 → **confirmed = 28 August 1921**
s3 → **confirmed = 8 April 1904**

4. State-Grounded Answering. The system compares the confirmed dates:

28 August 1921 > 8 April 1904.

State-Guided RAG Output: El Extraño Viaje

Analysis. This example shows that State-Guided RAG explicitly decomposes the comparison question into dependent evidence slots and resolves them in order. The system first confirms the two directors, then uses these confirmed bridge entities to retrieve their birth dates. Since the final comparison is performed over confirmed state values, the answer is grounded in explicit evidence rather than inferred from a noisy retrieved context. This illustrates how state-guided retrieval helps reduce missing-hop errors and supports reliable final answer grounding.

Case Study: Q-DREAM Trace and Analysis

Question: Which film has the director who was born later, *El Extraño Viaje* or *Love In Pawn*?

Ground Truth: El Extraño Viaje

Final Answer: Love In Pawn

Trace

1. Query Decomposition. Q-DREAM decomposes the comparison question into four subquestions: the director of *El extraño viaje*, the director of *Love in Pawn*, the birth date of the first director, and the birth date of the second director. This decomposition is appropriate because the question requires resolving both bridge entities before comparing their birth dates.

2. Bridge Retrieval. Q-DREAM retrieves the two directors correctly. The evidence confirms that *El extraño viaje* is directed by **Fernando Fernán Gómez**, while *Love in Pawn* is directed by **Charles Saunders**.

3. Dependency Rewriting. After obtaining the bridge entities, Q-DREAM rewrites the dependent subquestions into standalone queries: *Fernando Fernán Gómez → date of birth* and *Charles Saunders → date of birth*. Both rewrites correctly substitute the identified directors into the dependent retrieval targets.

4. Dependent Evidence Retrieval. Q-DREAM then retrieves the birth-date evidence for both directors. The retrieved evidence confirms that **Fernando Fernán Gómez was born on 28 August 1921** and **Charles Saunders was born on 8 April 1904**.

5. Final Answering Error. The retrieved evidence implies that **28 August 1921 > 8 April 1904**, so the director born later is **Fernando Fernán Gómez**, and the correct film should be **El Extraño Viaje**. However, Q-DREAM outputs **Love In Pawn**.

Analysis. This case is not a decomposition, dependency rewriting, or retrieval failure. Q-DREAM successfully retrieves all key evidence needed to answer the question, including both directors and both birth dates. The error occurs in the final answering stage, where the model reverses the temporal comparison for *born later*. Thus, the failure is best characterized

as a **final comparative reasoning error after successful evidence retrieval**.

Case Study: RAPTOR Trace and Analysis

Question: Which film has the director who was born later, *El Extraño Viaje* or *Love In Pawn*?

Ground Truth: **El Extraño Viaje**

Final Answer: **Love in Pawn**

Trace

1. Initial Retrieval. RAPTOR builds a two-layer tree with 16 nodes and retrieves the final collapsed-tree node set **{15, 11, 2, 0, 14, 3, 4, 7}**. These nodes already contain the key comparison evidence: *El extraño viaje* is directed by **Fernando Fernán Gómez**, while *Love in Pawn* is directed by **Charles Saunders**.

2. Summary-Node Evidence. Among the retrieved nodes, RAPTOR also selects higher-level summary nodes. Node **14** contains the biography evidence **Fernando Fernán Gómez (28 August 1921)**, while node **11** contains **Charles Saunders (8 April 1904)**. Thus, the retrieved context already provides the two dates needed for comparison.

3. Noisy Collapsed-Tree Context. Although the necessary evidence is present, the retrieved context is mixed with distracting entries such as *Luck in Pawn*, *A Lover in Pawn*, *W. Augustus Barratt*, and *Rafaela Aparicio*. These semantically nearby but unnecessary nodes make the final context less clean.

4. Output. The retrieved evidence implies: **28 August 1921 > 8 April 1904**. Therefore, the correct answer should be **El Extraño Viaje**. However, RAPTOR finally outputs **Love in Pawn**, which is incorrect.

Analysis. This case shows that RAPTOR can retrieve **sufficient supporting evidence** for the question, including both directors and both birth dates. The failure arises not from missing evidence, but from *noisy collapsed-tree retrieval*: the final reader must synthesize relevant facts together with distracting film and entity nodes. As a result, RAPTOR makes a wrong **final comparison / answer selection** despite having already retrieved enough evidence to answer correctly.

Case Study: GraphAnchor Trace and Analysis

Question: Which film has the director who was born later, *El Extraño Viaje* or *Love In Pawn*?

Ground Truth: ***El Extraño Viaje***

Final Answer: ***El extraño viaje***

Trace

1. Initial Retrieval. GraphAnchor retrieves evidence for both candidate films. *Love in Pawn* is directed by **Charles Saunders**, while *El extraño viaje* is directed by **Fernando Fernán Gómez**.

2. State Judgement. The model marks the current state as **sufficient**. Therefore, the before-version system stops after the initial stage and does not trigger another retrieval round.

3. Missed Follow-up. The model also generates a useful follow-up query: ***What are the birth dates of Charles Saunders and Fernando Fernán Gómez?*** However, this query is **not executed**, because the state has already been judged sufficient.

4. Output. The passage-grounded final answer is ***El extraño viaje***, which is correct. However, the graph-only branch predicts ***Love In Pawn***, which is incorrect.

Analysis. This case shows that GraphAnchor can still obtain the correct answer through **passage-grounded generation**, even when the generated graph is unreliable. The retrieved passages identify the two directors but do not provide their birth dates. Although the model proposes the right follow-up question, the system does not execute it due to the early **sufficient** judgement. Therefore, this example exposes a limitation of the stopping mechanism: a useful next query may be ignored, and the graph-only branch may produce an incorrect answer.

C Additional Experiments and Analyses

This section collects supplementary experimental results and diagnostic analyses that support the main findings, including answer rewriting control, efficiency analysis, structural prompt rule ablation, slot-budget sensitivity, index storage comparison, and rule-based error analysis.

C.1 Answer Rewriting Ablation

To examine the effect of the final answer rewriting step, we evaluate the raw answer generated before rewriting using the same evaluation script. In the w/o Answer Rewriting setting, the pipeline is unchanged except that the raw answer is directly scored. Table 6 shows that w/o Answer Rewriting achieves comparable performance to the full pipeline, indicating that the reported improvements primarily come from evidence-grounded state tracking and state-conditioned actions rather than post-processing. The main table retains the full State-Guided RAG results for consistency with the proposed answer-grounding procedure. In fact, removing answer rewriting slightly improves the averaged score, suggesting that the final gains are not driven by formatting-oriented post-processing.

C.2 Efficiency Analysis

Table 7 compares latency, reasoning steps, success/-failure counts, and token usage on 500 examples per dataset with Qwen3-14B. NaïveRAG is the fastest and uses the fewest tokens because it performs only one-shot retrieval. Compared with A-RAG, our method requires less average inference time and fewer tokens on all three datasets. Combined with the performance results in Table 1, these efficiency results suggest that State-Guided RAG achieves a better effectiveness-efficiency trade-off than heavier agentic baselines such as A-RAG. GraphAnchor is faster than our method, but its average performance is lower across the evaluated datasets and model scales.

C.3 Structural Prompt Rules Ablation

Table 8 studies whether the manually written structural rules in the decomposition prompt are the main source of performance gains. The ablated variant removes these rules, including comparison, geographic, kinship, and work/person chain rules, while keeping the same JSON schema and the same state-guided retrieval pipeline. The average F1 decreases only slightly from 61.22 to 60.57, and the differences are small and mixed across datasets. This suggests that structural rules provide useful but not uniformly beneficial priors for typed state induction. The result is consistent with the main ablation study, where removing Typed State Induction causes a much smaller drop than removing the state-grounded action policy. Overall, the primary gains are better attributed to evidence-grounded

Method	MuSiQue		2Wiki		HotpotQA		Average	
	EM	F1	EM	F1	EM	F1	EM	F1
w/o Answer Rewriting	32.67	43.87	57.87	65.92	53.93	67.89	48.16	59.23
State-Guided RAG	33.20	43.95	55.60	64.42	54.00	67.66	47.60	58.68

Table 6: Effect of answer rewriting with Qwen3-14B on 1,500 examples per dataset. w/o Answer Rewriting directly evaluates the raw answer before the final rewriting step, while State-Guided RAG denotes the full method with answer rewriting.

Dataset	Method	N	Success / Fail	Avg Time/Q	P90 Time	Avg Steps	Avg Total Tok.
HotpotQA	NaïveRAG	500	500 / 0	**0.38s**	**0.48s**	**1.00**	**1198.06**
	GraphAnchor	500	500 / 0	1.42s	2.24s	6.01	4371.13
	A-RAG	500	485 / 15	24.10s	33.75s	2.35	11254.30
	State-Guided RAG	500	480 / 20	14.53s	22.68s	3.24	7465.28
2Wiki	NaïveRAG	500	500 / 0	**0.41s**	**0.51s**	**1.00**	**1818.85**
	GraphAnchor	500	500 / 0	1.41s	1.78s	6.01	5997.81
	A-RAG	500	465 / 35	26.14s	36.15s	3.10	13259.32
	State-Guided RAG	500	479 / 21	16.51s	25.30s	3.54	8878.27
MuSiQue	NaïveRAG	500	500 / 0	**0.29**	**0.37**	**1.00**	**1358.80**
	GraphAnchor	500	500 / 0	1.53s	2.44s	6.02	4973.01
	A-RAG	500	475 / 25	30.95s	45.16s	2.80	12214.92
	State-Guided RAG	500	494 / 6	12.38s	16.94s	3.61	8222.37

Table 7: Efficiency and token usage comparison with Qwen3-14B on 500 examples per dataset. Avg Time/Q and P90 Time are reported in seconds and exclude initialization; Avg Total Tok. denotes the average total tokens per question across all LLM calls.

state tracking and state-conditioned retrieval control rather than prompt engineering alone.

C.4 Slot Budget Analysis

Table 9 analyzes the effect of state capacity by varying the maximum number of induced slots and comparing fixed budgets with the dynamic setting. Using only one slot yields 34.22 average F1, showing that a single state variable is insufficient for representing intermediate entities, dependencies, and answer targets. Increasing the budget greatly improves performance, and Budget=4 achieves the best average result with 61.37 F1 and 50.40 EM.

Further increasing the budget to six or eight does not bring consistent gains and increases inference time. The dynamic setting achieves comparable performance, with 61.22 average F1 and 49.73 average EM, while maintaining moderate latency. Overall, these results suggest that the state should be compact but expressive: too few slots under-represent multi-hop structure, whereas too many slots may introduce unnecessary fragmentation and computational overhead.

C.5 Index Storage Analysis

Table 10 reports the disk storage of retrieval indexes. Our method builds a standard chunk-level dense semantic FAISS index for each dataset. It does not introduce additional offline graph, tree, or hierarchical knowledge indexes beyond the dense retrieval index. Therefore, the storage cost mainly comes from the standard semantic retrieval index and the associated metadata, rather than an extra structured knowledge index.

C.6 HotpotQA Error Analysis

We conduct a rule-based diagnostic error analysis on HotpotQA with Qwen3-14B using 1,500 examples. For our method, we first verify the prediction file with the same EM/F1 script used in the main experiments, obtaining 54.00 EM and 67.66 F1. The taxonomy is computed over the 690 EM-incorrect examples. For NaïveRAG and Q-DREAM, we use the per-example EM/F1 values stored in their original prediction logs to ensure consistency with their reported results.

The error categories are assigned by deterministic rules over the saved prediction logs, retrieved evidence records, intermediate state traces, and normalized answer outputs. Each EM-incorrect ex-

Method	MuSiQue		2Wiki		HotpotQA		Average			
	EM	F1	EM	F1	EM	F1	EM	F1	Steps	Time/Q
w/o Structural Prompt Rules	39.60	49.93	56.60	65.95	53.60	65.83	49.93	60.57	3.49	14.99s
State-Guided RAG	37.80	48.79	57.00	66.41	54.40	68.46	49.73	61.22	3.49	15.85s

Table 8: Ablation results of structural prompt rules with Qwen3-14B on 500 examples per dataset. The w/o Structural Prompt Rules variant removes the manually specified comparison, geographic, kinship, and related structural rules from the decomposition prompt while keeping the same JSON schema and state-guided inference pipeline.

Slot Budget	MuSiQue		2Wiki		HotpotQA		Average			
	EM	F1	EM	F1	EM	F1	EM	F1	Steps	Time/Q
Budget=1	10.20	19.35	25.80	31.64	41.20	51.67	25.73	34.22	1.29	5.24s
Budget=2	37.00	47.42	50.80	60.08	55.80	68.88	47.87	58.79	3.09	11.36s
Budget=4	37.60	48.03	**57.40**	**66.76**	**56.20**	**69.32**	**50.40**	**61.37**	3.45	14.40s
Budget=6	38.20	48.76	55.00	64.47	52.00	65.71	48.40	59.65	3.49	15.94s
Budget=8	**38.80**	**49.54**	56.00	64.83	53.40	67.23	49.40	60.53	3.49	16.42s
Dynamic	37.80	48.79	57.00	66.41	54.40	68.46	49.73	61.22	3.49	15.85s

Table 9: Slot budget sensitivity with Qwen3-14B on 500 examples per dataset. Dynamic denotes the adaptive state-guided setting. Time/Q is average inference time per question; best and second-best EM/F1 scores are bolded and underlined.

ample is assigned one dominant error type. When multiple rules are triggered, we apply a fixed priority order that follows the pipeline: runtime or empty-output errors are assigned first, followed by retrieval or evidence gaps, state-update or decomposition-planning errors, answer-target or comparison errors, and finally answer-format or alias errors. Therefore, the taxonomy is intended as a diagnostic analysis of pipeline-level failure patterns rather than a human-annotated error benchmark.

Ours. Table 11 shows that the remaining errors of our method are mainly state update or verification errors, accounting for 34.93% of wrong cases. Under the rule-based taxonomy, this category corresponds to cases where the logs indicate that the system may verify an incorrect candidate value, assign a value to the wrong slot, or fail to distinguish weak candidates from confirmed facts after retrieval. The second largest category is answer format or alias error, accounting for 27.25%. These errors often involve entity aliases, abbreviations, or over-specific answer strings. Retrieval or evidence gaps account for 13.91%, indicating that state-guided retrieval reduces but does not eliminate evidence acquisition failures.

NaïveRAG. Table 12 shows that NaïveRAG suffers strongly from evidence-coverage limitations. Retrieval or evidence gaps account for 17.94% of wrong cases, while insufficient answers caused by retrieval gaps account for another 16.11%. Together, these retrieval-related failures account for 34.05% of NaïveRAG errors. This pattern is consistent with our motivation that single-shot retrieve-then-read pipelines often fail to collect the evidence required for multi-hop reasoning. Since NaïveRAG performs only one retrieval step, it cannot explicitly track intermediate entities or issue follow-up queries when the second-hop evidence is missing.

Q-DREAM. Table 13 shows a different failure pattern. Q-DREAM introduces multi-step decomposition and retrieval, but its largest error category is decomposition or retrieval-planning gap. Under the rule-based taxonomy, this means that incorrect or incomplete subquestions may redirect the subsequent retrieval chain. The observed Q-DREAM errors therefore suggest that multi-step retrieval alone is not sufficient when the generated subquestions or rewritten queries drift away from the original entity-relation dependency.

Comparison. The three error distributions reveal different bottlenecks. NaïveRAG mainly fails because one-shot retrieval cannot cover multi-hop ev-

Method	Index Type	Embedding Model	HotpotQA	2Wiki	MuSiQue	Approx. Total
NaïveRAG	paragraph-level semantic index	bge-m3	2.22 GB	1.84 GB	521.73 MB	4.58 GB
FLARE	Elasticsearch BM25 inverted index	None	254.81 MB	176.84 MB	61.10 MB	0.49 GB
IRCoT	BM25-based inverted index	None	5.2 GB	223.1 MB	83.5 MB	5.51 GB
RAPTOR	hierarchical tree index, built on the fly	bge-m3	N.A.	N.A.	N.A.	N.A.
GraphAnchor	chunk-level semantic index	bge-m3	2.0 GB	1.7 GB	460 MB	4.16 GB
Q-DREAM	DPRM cluster-specific semantic index	bge-m3 + DPRM-LoRA	4.4 GB	3.7 GB	1.4 GB	9.50 GB
A-RAG	sentence-level semantic index	bge-m3	206.67 MB	105.43 MB	1.75 GB	2.06 GB
Ours	chunk-level semantic FAISS index	bge-m3	2.27 GB	1.97 GB	535.57 MB	4.78 GB

Table 10: Comparison of index storage across different RAG methods and datasets. N.A. indicates that the index is not persistently stored on disk. Approximate totals are computed from the displayed storage values, with MB values converted to GB for readability.

Major Error Type	Count	Ratio among Wrong	Ratio among All
State update / verification error	241	34.93	16.07
Answer format / alias error	188	27.25	12.53
Retrieval or evidence gap	96	13.91	6.40
Answer-target missing	80	11.59	5.33
Comparison reasoning / selection error	52	7.54	3.47
Empty or runtime output	24	3.48	1.60
Answer generation / extraction error	5	0.72	0.33
Unresolved conflict error	3	0.43	0.20
Answer rewriting / extraction error	1	0.14	0.07

Table 11: Rule-based major error taxonomy of our State-Guided RAG model on 1,500 HotpotQA examples with Qwen3-14B. The taxonomy is computed on the 690 examples that are incorrect under the same EM/F1 evaluation script used in the main experiments. Each example is assigned one dominant error type using deterministic rules over prediction logs, retrieved evidence records, intermediate state traces, and normalized answer outputs.

idence. Q-DREAM performs multi-step retrieval, but often fails when decomposition or query rewriting redirects retrieval to an incorrect subquestion chain. In contrast, our method reduces the number of HotpotQA EM errors from 875 for NaïveRAG and 951 for Q-DREAM to 690. The remaining errors shift toward state update or verification and answer alias/format mismatch. This supports our central claim: effective agentic RAG requires not only iterative retrieval, but also an explicit evidence-grounded state that controls which information is missing, which dependency has been confirmed, and which retrieval action should be taken next.

D Retrieval and Indexing Details

For our method, each dataset is indexed with a chunk-level dense FAISS index using BGE-M3 embeddings. Retrieved passages are reranked before being passed to the state update module. The default first-hop reranked evidence cap is 3, and the final merged evidence limit is 8. The merged evidence pool may include first-hop retrieval results, title-based hits, value-title hits, and second-hop retrieval results triggered by confirmed dependency values.

The TopK analysis explicitly constrains the final merged evidence provided to each state update step to at most K passages, while leaving the remaining inference pipeline unchanged. Therefore, the TopK setting measures the effect of evidence budget per state update step rather than the raw dense-retrieval candidate pool size.

E Additional Notes on Reproducibility

All experiments are conducted with local LLMs and local retrieval indexes. The released repository does not include raw datasets, model weights, or prebuilt indexes. Users should prepare HotpotQA, 2Wiki, and MuSiQue, build the dataset-specific retrieval indexes, and update the corresponding paths in the configuration files. The reported token costs are tokenizer-estimated with the Qwen3 tokenizer over all LLM calls, including both prompt inputs and generated outputs.

F Prompt Templates

This section lists the main prompt templates used in our implementation. We lightly format line breaks for readability while preserving the core instructions and output schemas. The decomposition prompt is split into two parts to avoid oversized prompt boxes in the appendix.

Major Error Type	Count	Ratio among Wrong	Ratio among All
Reader reasoning error with retrieved evidence	224	25.60	14.93
Answer format / alias error	160	18.29	10.67
Retrieval or evidence gap	157	17.94	10.47
Insufficient answer due to retrieval gap	141	16.11	9.40
Insufficient answer despite retrieved evidence	116	13.26	7.73
Comparison reasoning / selection error	77	8.80	5.13

Table 12: Rule-based major error taxonomy of NaïveRAG on 1,500 HotpotQA examples with Qwen3-14B. NaïveRAG performs single-shot retrieval and generation, and many errors are categorized as evidence gaps or insufficient answers after one retrieval step. Each EM-incorrect example is assigned one dominant error type using deterministic rules over prediction logs, retrieved evidence records, and normalized answer outputs.

Major Error Type	Count	Ratio among Wrong	Ratio among All
Decomposition / retrieval planning gap	489	51.42	32.60
Final reasoning error with retrieved evidence	214	22.50	14.27
Answer format / alias error	145	15.25	9.67
Comparison reasoning / selection error	77	8.10	5.13
Insufficient answer due to retrieval gap	16	1.68	1.07
SDOM query rewrite error	8	0.84	0.53
Insufficient answer despite retrieved evidence	1	0.11	0.07
Empty or runtime output	1	0.11	0.07

Table 13: Rule-based major error taxonomy of Q-DREAM on 1,500 HotpotQA examples with Qwen3-14B. Most errors are categorized as decomposition or retrieval-planning gaps under the rule-based diagnostic taxonomy, indicating that incorrect subquestions can redirect the subsequent retrieval chain. Each EM-incorrect example is assigned one dominant error type using deterministic rules over prediction logs, retrieved evidence records, rewritten subquestions, and normalized answer outputs.

F.1 Decomposition Prompt

This prompt is used to induce evidence slots from the input question. It does not answer the question. Instead, it creates a structured slot schema for subsequent state tracking.

F.2 State Update Prompt

This prompt updates the current state using newly retrieved evidence. Slot values are revised only when supported by evidence.

F.3 Answer Generation Prompt

The final answer is generated from the current knowledge state. The prompt prioritizes confirmed answer slots and requires the answer string to be concise.

F.4 Answer Rewriting Prompt

This prompt rewrites a raw answer into a shorter canonical form for exact-match style evaluation. It does not access gold answers or evaluation labels.

Prompt F.1a: Evidence Slot Decomposition

```
You are a retrieval planner for multi-hop question answering.

Given a question, decompose it into a small set of evidence slots.
Each slot should describe a fact that must be retrieved or verified.
Do NOT answer the question.
Do NOT invent facts.

Requirements:
- Use 2 to 6 slots.
- Mark each slot with:
  - slot_id
  - description
  - slot_type
  - depends_on
  - target_role
  - expected_answer_type
  - cardinality
- target_role:
  - "bridge" for intermediate entity/bridge facts
  - "dependent" for facts that depend on another slot
  - "answer_target" for the slot whose value most directly determines the final answer
- There must be EXACTLY ONE answer_target slot.
- Use depends_on whenever a slot relies on another slot.
- Preserve the original relation semantics exactly.
```

Prompt F.1b: Structural Rules and Output Schema

```
Important structural rules:
1. For kinship composition, expand the chain explicitly:
   - paternal grandmother = father's mother
   - paternal grandfather = father's father
   - maternal grandmother = mother's mother
   - maternal grandfather = mother's father
   - spouse of X = first resolve X, then spouse(X)
   - child of X = first resolve X, then child(X)

2. For administrative/geographic chains, preserve every hop:
   - place -> country -> population
   - birthplace -> county
   - city -> state/province -> bordering entity
   - employer -> headquarters

3. For comparison questions:
   - preserve candidate A and candidate B explicitly
   - preserve the compared attribute for each candidate explicitly
   - answer_target should be the selected candidate, not the compared attribute value

4. For common-super-location questions:
   - resolve each entity's location first
   - then derive the shared / higher-level answer in answer_target

5. For work/person chains:
   - keep work attributes separate from person attributes
   - performer label is not the same as release label
   - director birthplace is about the director, not the film

6. WH questions should not end with a boolean answer_target unless the question is truly yes/no.

Return strict JSON:
{
  "slots": [
    {
      "slot_id": "s0",
      "description": "...",
      "slot_type": "entity|relation|attribute|event|date|location|boolean|other",
      "depends_on": [],
      "target_role": "bridge|dependent|answer_target|other",
      "expected_answer_type": "person|date|location|county|province|record_label|award|team|instrument|era|boolean|other",
      "cardinality": "single|multi"
    }
  ]
}

Question: {{question}}
```

Prompt F.2: Evidence-Grounded State Update

```
You are updating a structured knowledge state for a multi-hop QA agent.

Given:
1. the original question
2. current knowledge state
3. newly retrieved evidence snippets

Your job:
- fill or revise slot values only when justified by evidence
- keep unsupported slots unresolved
- mark conflicts when evidence disagrees
- estimate confidence in [0,1]

Return strict JSON:
{
  "slot_updates": [
    {
      "slot_id": "s1",
      "value": "...",
      "status": "missing|candidate|confirmed|conflict",
      "confidence": 0.0,
      "supporting_evidence_ids": ["e1", "e2"],
      "notes": "..."
    }
  ],
  "conflicts": [
    {
      "slot_id": "...",
      "reason": "..."
    }
  ],
  "next_missing_slots": ["..."]
}

Question: {{question}}
Current state:
{{state_json}}
Evidence:
{{evidence_json}}
```

Prompt F.3: State-Grounded Answer Generation

```
You are answering a question using only the current knowledge state.

Rules:
- Prioritize slots with target_role = "answer_target".
- Use confirmed facts first.
- Do not use unsupported world knowledge.
- The field "final_answer" must contain ONLY the shortest possible answer string.
- Do NOT put explanation, reasoning, or extra words in "final_answer".
- If the answer is yes/no, output exactly "yes" or "no".
- If the answer is an entity, output only the entity name.
- If the answer is a date, output only the date string.
- Put any brief explanation only in "rationale".

Return strict JSON:
{
  "final_answer": "...",
  "rationale": "..."
}

Question: {{question}}
State:
{{state_json}}
```

Prompt F.4: Answer Rewriting

```
You are rewriting an answer into the shortest canonical form for exact-match style QA evaluation.

Rules:
- Keep the meaning unchanged.
- "final_answer" must contain ONLY the shortest answer string.
- Remove explanation, prefix, suffix, and extra detail.
- If the answer is a location, prefer the standard place name.
- If the answer is a person/entity, output only the entity name.
- If the answer is a date, output only the date string.
- If the answer is yes/no, output exactly "yes" or "no".
- Do not use external knowledge.
- Do not use gold answers.

Return strict JSON:
{
  "final_answer": "..."
}

Question: {{question}}
Raw answer:
{{raw_answer}}
State:
{{state_json}}
```