S2G-RAG: Structured Sufficiency and Gap Judging for Iterative Retrieval-Augmented QA

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) grounds language models in external evidence, but multi-hop question answering remains difficult because iterative pipelines must control what to retrieve next and when the available evidence is adequate. In practice, systems may answer from incomplete evidence chains, or they may accumulate redundant or distractor-heavy text that interferes with later retrieval and reasoning. We propose **S2G-RAG** (**S**tructured **S**ufficiency and **G**ap-judging RAG), an iterative framework with an explicit controller, S2G-Judge. At each turn, S2G-Judge predicts whether the current evidence memory supports answering and, if not, outputs structured gap items that describe the missing information. We map these gap items into the next retrieval query, producing stable multi-turn retrieval trajectories. To reduce noise accumulation, we maintain a sentence-level Evidence Context by extracting a compact set of relevant sentences from retrieved documents. Experiments on TriviaQA, HotpotQA, and 2WikiMultiHopQA show that S2G-RAG improves multi-hop QA performance and robustness under multi-turn retrieval. Furthermore, S2G-RAG can be integrated into existing RAG pipelines with a lightweight component, without modifying the search engine or retraining the generator.

1 Introduction

Retrieval-augmented generation (RAG) grounds large language models (LLMs) in external evidence and has become a standard approach for knowledge-intensive question answering (Lewis et al., 2020; Shuster et al., 2021; Arslan et al., 2024). While single-hop questions can often be answered from one document, multi-hop question answering requires composing evidence across multiple documents, where later retrieval depends on intermediate findings from earlier rounds (Yang et al., 2018; Tang and Yang, 2024). This motivates iterative RAG, which interleaves retrieval and reasoning over multiple turns. However, iterative RAG faces a retrieval control bottleneck. At each turn, the system must assess whether the accumulated evidence supports answering, and if not, specify what to retrieve next (Yang et al., 2025; Ye et al., 2025). When control is unreliable, systems either answer from incomplete evidence chains or continue retrieving without a clear target, accumulating redundant or distractor-heavy text that interferes with subsequent retrieval and reasoning.

A Hotpot-style question such as "In which country was the director of *Home Alone* born?" illustrates the challenge. An initial retrieval may identify the bridge entity (Chris Columbus) but still miss the answer-bearing attribute (his country of birth). Answering at this point fails, whereas continuing with generic follow-up queries can repeatedly retrieve near-duplicate pages and grow noisy contexts. Effective iterative RAG therefore requires an auditable turn-level state that reflects whether the current evidence is adequate and, when it is not, what concrete information gap remains.

Recent work improves iterative retrieval via query rewriting, decomposition, intermediate reasoning traces, and self-reflection or critic signals (Trivedi et al., 2023; Asai et al., 2024; Yang et al., 2025; Dong et al., 2025; Wang et al., 2025; Jiang et al., 2023). Despite progress, three limitations remain. First, control is often implicit and entangled with free-form generation, which makes decisions difficult to audit and brittle under distractors. Second, the representation of the next-hop information need is under-specified, leading to drift, repetition, or over-generalization when intermediate contexts are noisy. Third, robust control benefits from turn-level supervision, but intermediate-state annotation is expensive (Yang et al., 2025), and supervision from idealized trajectories can misalign with the intermediate evidence contexts produced

by multi-turn pipelines. Meanwhile, iterative retrieval naturally accumulates long and noisy text, and naive concatenation can further destabilize control (Hwang et al., 2025; Jiang et al., 2025; Liu et al., 2024).

We propose **S2G-RAG** (**S**tructured **S**ufficiency and **G**ap-judging RAG), an iterative framework that makes retrieval control explicit and modular. Its lightweight controller, S2G-Judge, takes the question q and the accumulated evidence context C_t and outputs a binary sufficiency decision together with structured gap items that describe the missing information for the next hop. S2G-RAG maps the predicted gap items into the next retrieval query, making the next-hop target explicit and diagnosable. The controller is decoupled from answer generation, so the answer reasoner operates only on the accumulated evidence context.

To mitigate noise accumulation, S2G-RAG maintains a compact sentence-level evidence context. Given newly retrieved documents, an LLM-based Evidence Extractor selects salient sentences and explicitly prioritizes those aligned with the predicted gap items, producing an auditable evidence context for subsequent judging and final answering.

We train S2G-Judge with process supervision distilled from multi-turn pipeline execution traces. We run the retrieval and evidence accumulation procedure on training questions and log per-turn snapshots (q, C_t) by rolling out to a fixed maximum budget. A strong teacher then labels each snapshot under a context-only evidence constraint with sufficiency decisions and gap items, and we distill these signals into a compact judge model (Lightman et al., 2023; Hsieh et al., 2023).

Our main contributions are as follows.

- **Structured sufficiency and gap judging.** We formulate iterative RAG control as per-turn structured prediction of evidence sufficiency and explicit information gaps, implemented via a lightweight judge decoupled from answer generation.

- **Distillation from execution traces.** We introduce a supervision scheme built from multi-turn execution traces, enabling turn-level learning under realistic intermediate evidence contexts.

- **Gap-aware sentence-level evidence context.** We maintain a compact, auditable evidence context via pointer-based extraction with explicit gap-aware prioritization to reduce distractor interference.

- **Empirical validation.** Across TriviaQA, HotpotQA, and 2WikiMultiHopQA, S2G-RAG improves EM/F1 over strong baselines under both sparse and dense retrieval settings.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) grounds LLM outputs in external corpora by retrieving relevant documents and conditioning generation on the retrieved evidence (Lewis et al., 2020; Shuster et al., 2021; Arslan et al., 2024). Most standard RAG systems follow a single-round retrieve-then-generate pattern, which is effective when the required evidence can be covered in one retrieval pass. However, many knowledge-intensive questions require multi-step reasoning over multiple sources, including synthesizing evidence across documents, aligning entities and relations, and resolving information needs that only become clear after inspecting initial retrieval results. This motivates multi-round retrieval-and-reasoning systems that iteratively update an information state before producing the final answer.

2.2 Multi-Round RAG

Multi-hop question answering requires integrating multiple evidence pieces and performing multi-step reasoning, and is widely used to evaluate both RAG systems and LLM reasoning capabilities. In this setting, a single retrieval pass is often insufficient because crucial evidence is distributed across documents and later-hop retrieval depends on intermediate findings from earlier rounds. Recent work therefore proposes iterative RAG frameworks that alternate between retrieval and reasoning over multiple rounds.

We organize related work along four themes that recur in iterative retrieval control and evidence management. These themes include intermediate query generation, stopping criteria, supervision for turn-level control, and context management under multi-turn noise and bloat.

Intermediate query generation. A major line of work improves multi-round retrieval by generating intermediate queries, including explicit planning and question decomposition into sub-questions, as

well as follow-up query rewriting from partial evidence (Ye et al., 2025; Jiang et al., 2023; Dong et al., 2025; Wang et al., 2025). While these approaches can improve later-hop recall and evidence composition, the intermediate information need is often expressed in free-form text. As a result, query generation can drift, repeat, or become unstable under distractors and noisy intermediate contexts, and reliable inference-time behavior may require repeated calls to strong models.

Stopping criteria. Another line of work studies when it is safe to answer, which is commonly framed as deciding whether the current evidence is sufficient (Yang et al., 2025). Several methods rely on self-reflection or critic-style verification to decide whether to continue retrieval (Asai et al., 2024; Yang et al., 2025). These signals can reduce incorrect answers produced under incomplete evidence, but the control state is often not standardized or easily auditable. Moreover, when retrieval continues, these approaches typically provide limited explicit structure for specifying the remaining information need.

Supervision for turn-level control. Learning reliable turn-level control benefits from process-level supervision, yet scalable, high-quality labels for intermediate decisions and next-step information needs are difficult to obtain (Yang et al., 2025). Prior work explores distilling supervision from stronger teachers or leveraging trajectory-based learning signals for multi-step decision making (Lightman et al., 2023; Hsieh et al., 2023). A recurring challenge is distribution mismatch, where supervision derived from synthetic trajectories or heuristic signals may not reflect the intermediate evidence states produced by multi-turn pipelines in practice, which often include redundancy and distractors.

Context management. Iterative retrieval tends to accumulate long and noisy contexts, which weakens evidence focus and can destabilize both control and reasoning (Liu et al., 2024). To mitigate context bloat, prior work compresses retrieved content or performs sentence-level evidence selection (Dhole, 2025; Hwang et al., 2025). These techniques are complementary to retrieval control because they reduce the influence of redundant or distractor-heavy text, but they do not by themselves specify what to retrieve next or provide an explicit representation of the missing information that drives subsequent retrieval.

3 Methodology

This section presents **S2G-RAG**, an iterative RAG framework for multi-hop QA that makes retrieval control explicit through structured sufficiency and gap judging. We first describe the S2G-RAG inference loop (Section 3.1), then introduce S2G-Judge training via trajectory distillation from execution traces (Section 3.2), and finally detail gap-guided query construction and sentence-level evidence extraction (Sections 3.3–3.4).

3.1 S2G-RAG Inference Framework

Given a question q and an external corpus $\mathcal{D}$, S2G-RAG maintains an accumulated evidence context C_t at turn t. We initialize C_0 as empty and run the loop for at most T retrieval turns. Figure 1 illustrates the overall framework. At each turn, S2G-RAG follows a judge-first loop with four components: S2G-Judge, a retriever, a sentence-level evidence extractor, and an answer reasoner.

S2G-Judge. At turn t, the judge reads only (q, C_t) and outputs $y_t = (s_t, G_t)$. The binary variable $s_t \in \{\texttt{true}, \texttt{false}\}$ indicates whether the current evidence context is sufficient to support answering, and G_t is a set of structured *gap items* (Section 3.2.1). When $s_t = \texttt{true}$, the system proceeds to answer using (q, C_t). This sufficiency signal is primarily used to reduce answers produced under incomplete evidence.

Retriever. When $s_t = \texttt{false}$, the system constructs a query $\tilde{q}_t$ from G_t (Section 3.3) and retrieves top-k documents $D_t = \{d_{t,1}, \ldots, d_{t,k}\}$ from $\mathcal{D}$.

Evidence extractor. A sentence-level extractor h selects salient sentences from D_t conditioned on (q, G_t) and forms a verbatim evidence block E_t. We update the evidence context by concatenation

$$C_t = C_{t-1} \oplus E_t, \tag{1}$$

where $\oplus$ denotes concatenation.

Reasoner. The answer reasoner r_θ is decoupled from retrieval control. It is invoked only when the system decides to answer or when the loop reaches the maximum budget. In either case, the reasoner produces the final answer conditioned on (q, C_t).

A pseudocode description of the inference loop is provided in Appendix C.

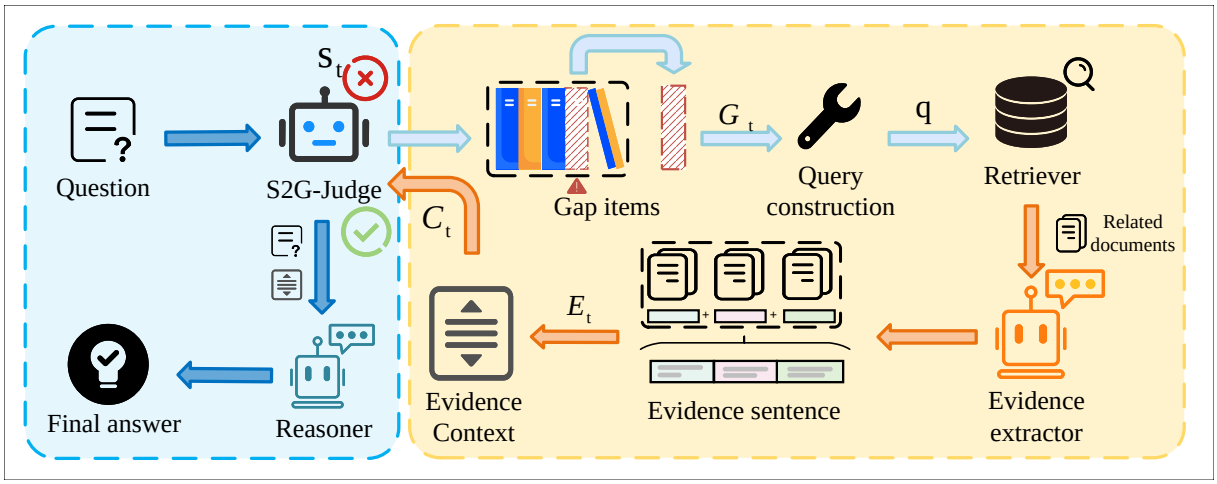

Figure 1: Overview of the S2G-RAG inference framework. Given a question q and the current evidence context C_t, S2G-Judge first predicts a sufficiency decision s_t and structured gap items G_t. If $s_t = \texttt{true}$, a reasoner generates the final answer from (q, C_t); otherwise, G_t guides query construction for retrieval, and a sentence-level evidence extractor selects salient evidence sentences from retrieved documents to form E_t, which is appended to update $C_t = C_{t-1} \oplus E_t$. The process iterates until the evidence is sufficient or the retrieval budget is reached.

3.2 S2G-Judge

3.2.1 Output Schema and Constraints

At each turn t, S2G-Judge predicts a structured output

$$y_t = (s_t, G_t), \quad s_t \in \{\texttt{true}, \texttt{false}\}. \tag{2}$$

If $s_t = \texttt{true}$, the judge outputs an empty gap set, so $G_t = \emptyset$.

Each gap item $g \in G_t$ follows a normalized schema with four fields. The `category` field takes values in {`bridge_entity`, `attribute`, `relation`, `evidence_span`, `other`}. The `target` field specifies the entity being queried. The `slot` field indicates a coarse-grained attribute or relation name. The `description` field provides a short natural-language clarification.

We impose a context-only constraint for sufficiency judgments. The judge must determine sufficiency based strictly on evidence present in C_t, which prevents relying on parametric knowledge when deciding whether the evidence is adequate.

3.2.2 Training via Trajectory Distillation from Execution Traces

We train S2G-Judge with process supervision distilled from multi-turn execution traces. We execute the same iterative retrieval and evidence accumulation procedure on training questions and log per-turn states (q, C_t). During trace collection, we instantiate the control loop with an unfine-tuned judge backbone under the same prompt/interface, and roll out the retrieval process to the maximum budget T. This produces intermediate evidence contexts that reflect multi-turn accumulation, including redundancy and distractors, which serve as inputs for judge training.

A stronger teacher model labels each (q, C_t) under the context-only constraint. We apply lightweight conflict filtering to remove low-confidence supervision. We then perform LoRA-based supervised fine-tuning so that the judge directly predicts the structured output y_t conditioned on $x_t = (q, C_t)$

$$\mathcal{L}(\phi) = -\sum_{(x_t, y_t)} \sum_{i=1}^{|y_t|} \log p_\phi\big(y_{t,i} \mid y_{t,<i}, x_t\big). \tag{3}$$

Additional training details, including supervision construction and hyperparameters, are provided in Appendix B.

3.3 Gap-Guided Query Construction

When $s_t = \texttt{false}$, S2G-RAG constructs the next retrieval query $\tilde{q}_t$ by mapping predicted gap items to query phrases. For each gap item, we prioritize concatenating `target` and `slot` when both fields are present, and otherwise fall back to the `description` field. We take up to the first K valid phrases in the order produced by the judge and append them to the original question. If no valid phrase can be constructed, we set $\tilde{q}_t = q$.

3.4 Sentence-level Evidence Extractor

LLMs are sensitive to noisy or misleading evidence (Shi et al., 2023; Zeng et al., 2025) and can degrade under long inputs (Liu et al., 2024). In iterative retrieval, directly concatenating full documents across turns can rapidly expand C_t and amplify distractors. We therefore introduce a sentence-level evidence extractor h that converts newly retrieved documents into a compact evidence block before updating the accumulated evidence context.

At turn t, the retriever returns documents $D_t = \{d_{t,1}, \ldots, d_{t,k}\}$. We split each document into sentences and form a globally indexed candidate pool $\mathcal{S}_t$, where each candidate is paired with its document title for provenance. The extractor is an LLM prompted with $(q, G_t, \mathcal{S}_t)$ and is constrained to output only indices from $\mathcal{S}_t$, without rewriting or generating any evidence text. We then deterministically map the selected indices back to the original sentences and concatenate them to form the evidence block:

$$E_t = h(q, G_t, D_t), \qquad C_t = C_{t-1} \oplus E_t. \quad (4)$$

The extraction policy is designed to preserve multi-hop progress under noisy retrieval. The extractor is instructed to include answer-bearing sentences as well as intermediate bridge facts and disambiguating context for entities in q. The predicted gap items G_t provide an explicit prioritization signal, encouraging selection of sentences that are aligned with the current missing information need rather than broadly topical content.

This sentence-level evidence context serves two purposes. First, it limits the influence of redundant or distractor-heavy text that commonly appears in top-k retrieval results, which helps keep downstream judging and reasoning focused. Second, the pointer-based design makes the evidence context auditable and reduces the risk of hallucinated evidence.

4 Experiments

4.1 Task and Datasets

We evaluate **S2G-RAG** on three open-domain QA benchmarks spanning single-hop and multi-hop reasoning. For single-hop QA, we use TriviaQA (Joshi et al., 2017), where answers are often supported by a single Wikipedia article. For multi-hop QA, we adopt HotpotQA (Yang et al., 2018), which requires composing evidence across multiple documents, and 2WikiMultiHopQA (Ho et al., 2020), which emphasizes entity disambiguation and fine-grained evidence composition. Following established protocols, we report Exact Match (EM) and F1. For comparability, all methods index the official Wikipedia dump released with each benchmark as the retrieval corpus.

4.2 Implementation Details

We instantiate the answer reasoner with Llama-3-8B-Instruct. For simplicity, the Evidence Extractor uses the same backbone as the reasoner.

For retrieval control, we fine-tune a lightweight backbone with LoRA to obtain S2G-Judge variants based on Llama-3.2-3B-Instruct. Training supervision is distilled from multi-turn execution traces labeled by a stronger teacher model (GPT-4o-mini), following Section 3.2.2. Unless otherwise noted, S2G-RAG runs for at most T=4 retrieval turns, retrieves top-k=6 documents per turn, and applies title-based de-duplication across turns.

We report results under both sparse and dense retrieval settings. For sparse retrieval, we use BM25 implemented in Pyserini. For dense retrieval, we use E5-base-v2 (Wang et al., 2022). All methods share the same retrieval corpus for each dataset and are evaluated on the official development split. Our pipeline can be replicated with 2×vGPU-48G or equivalent hardware. LoRA configurations, decoding settings and additional training details are included in the Appendix B.

4.3 Baselines

We compare S2G-RAG against representative baselines spanning single-round RAG and multi-round retrieval control. NaiveGen (Jin et al., 2025) answers without retrieval. Standard RAG (Jin et al., 2025) performs a single retrieval using the original question and generates an answer conditioned on the retrieved documents. IR-CoT (Trivedi et al., 2023) interleaves retrieval with intermediate reasoning signals that serve as retrieval cues. FLARE (Jiang et al., 2023) triggers retrieval based on generation-time uncertainty signals. ReSP (Jiang et al., 2025) iterates retrieve–summarize–plan to manage multi-turn evidence accumulation. Self-RAG (Asai et al., 2024) fine-tunes a generator to emit explicit reflection tokens that govern retrieval behavior. SIM-RAG (Yang et al., 2025) trains a lightweight critic that, at each round, evaluates the reasoner's provisional answer and accompanying rationale against the currently retrieved context and decides whether to accept the

Method	Reasoner / Retriever	TriviaQA EM	TriviaQA F1	HotpotQA EM	HotpotQA F1	2Wiki EM	2Wiki F1
Panel A: Sparse retrieval (BM25)							
NaiveGen	Llama3-8B / –	55.7	63.1	20.6	28.4	26.4	33.9
IR-CoT	Llama3-8B / BM25	56.9	68.9	28.6	41.5	23.5	32.4
SIM-RAG	Llama3-8B / BM25	70.7	75.6	32.7	43.3	34.1	40.2
S2G-RAG (ours)	Llama3-8B / BM25	**77.1**	**82.6**	**43.3**	**56.5**	**36.8**	**44.8**
Panel B: Dense retrieval (E5)							
Standard RAG	Llama3-8B / E5	58.8	68.3	25.1	35.3	10.6	21.0
Self-RAG	Llama3-8B / E5	38.2	53.4	17.1	29.6	12.1	25.1
FLARE	Llama3-8B / E5	55.8	63.2	19.7	28.0	25.8	33.9
ReSP*	Llama3-8B / E5	–	–	–	47.2	–	38.3
RAG-Critic	Llama3-8B / E5	50.0	60.1	40.0	51.2	21.0	33.0
S2G-RAG (ours)	Llama3-8B / E5	**78.0**	**83.5**	**41.1**	**53.2**	**37.7**	**45.1**

Table 1: Main results (EM/F1) on TriviaQA, HotpotQA, and 2WikiMultiHopQA. Panel A reports methods under BM25 and Panel B reports methods under E5. Bold denotes the best score within each panel under the same retriever setting. ReSP* results are reported from the original paper for reference.

answer or continue searching. RAG-Critic (Dong et al., 2025) introduces an error-aware critic that is trained with a hierarchical error taxonomy and provides fine-grained feedback about potential RAG failures, which is then used to trigger targeted correction workflows in an agentic RAG pipeline.

4.4 Main Results

Table 1 reports end-to-end EM/F1 on TriviaQA, HotpotQA, and 2WikiMultiHopQA under both sparse (BM25) and dense (E5) retrieval. Across settings, S2G-RAG improves over strong baselines, supporting the benefit of making retrieval control explicit through structured sufficiency and gap judging.

Under a matched BM25 setting with the same answer reasoner, S2G-RAG substantially outperforms prior multi-round baselines. Relative to SIM-RAG, S2G-RAG improves TriviaQA by $+6.4$ EM and $+7.0$ F1 (from 70.7/75.6 to 77.1/82.6). On HotpotQA, the gains are larger at $+10.6$ EM and $+13.2$ F1 (from 32.7/43.3 to 43.3/56.5), consistent with the stronger need for accurate multi-hop progress. On 2WikiMultiHopQA, S2G-RAG yields a $+2.7$ EM and $+4.6$ F1 improvement (from 34.1/40.2 to 36.8/44.8). These gains are achieved without modifying the answer generator, indicating that structured retrieval control can improve multi-hop evidence acquisition and reduce cases where the system answers before retrieving answer-bearing evidence.

S2G-RAG also transfers across retriever families. With E5, S2G-RAG attains the best results within the dense-retrieval panel on all three benchmarks, outperforming Standard RAG and learned-control baselines reported in this setting such as RAG-Critic. This suggests that the proposed control interface is not tied to a particular retriever.

Finally, the TriviaQA results highlight a common challenge for iterative pipelines. When only a small amount of evidence is needed, additional retrieval can introduce redundant or distracting text that interferes with answer generation. S2G-RAG mitigates this issue by maintaining a compact sentence-level evidence context, so that increased retrieval does not directly translate into uncontrolled context growth. On multi-hop benchmarks, the improvements are larger because explicit gap specification and controlled evidence accumulation better support multi-step evidence composition.

4.5 Analysis Experiments

4.5.1 Ablation Study

Table 2 summarizes ablations on HotpotQA (dev) under BM25. We remove one component at a time from the full pipeline while keeping the retriever, retrieval budget, and answer reasoner fixed.

Removing S2G-Judge causes the largest drop (-15.8 EM, -19.0 F1), highlighting the importance of an explicit control state for multi-hop QA. Without the judge, the pipeline no longer has an evidence-conditioned sufficiency check, and it also loses structured gap signals that focus the next retrieval step on the missing hop.

Replacing the trained judge with its unfine-tuned backbone also degrades performance (-4.1 EM, -5.7 F1). This suggests that trajectory distillation on intermediate evidence contexts is important for

Variant	EM	F1
Full S2G-RAG	**43.3**	**56.5**
w/o SFT (Untrained Judge)	39.2 (−4.1)	50.8 (−5.7)
w/o S2G-Judge	27.5 (−15.8)	37.6 (−19.0)
w/o Extractor	39.5 (−3.8)	52.5 (−4.0)

Table 2: Ablation on HotpotQA (dev) under BM25. Starting from the full S2G-RAG pipeline, we ablate one component at a time while keeping the retriever, maximum turns T, per-turn top-k, and the answer reasoner fixed. w/o SFT replaces the LoRA-finetuned S2G-Judge with its unfine-tuned backbone under the same prompt and output schema. w/o S2G-Judge removes the judge module, so the pipeline runs without structured sufficiency or gap predictions. w/o Extractor appends raw retrieved text instead of sentence-level evidence blocks. Parentheses indicate absolute drops from the full system.

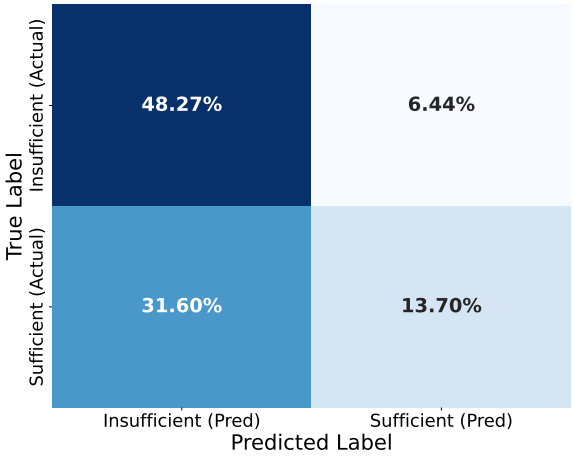

Figure 2: Confusion matrix of S2G-Judge's binary sufficiency decisions on HotpotQA (dev) using BM25 retrieval, with GPT-4o-mini as the reasoner and the evidence extractor. Here we use a retrieval-only ground-truth label TRUTH = CORRECTRETRIEVAL, where CORRECTRETRIEVAL is true iff the accumulated retrieved *titles* cover all gold supporting document titles for the instance. Values are percentages over the evaluated set.

robust sufficiency and gap prediction, especially when the accumulated context contains redundancy and distractors.

Finally, removing the sentence-level Evidence Extractor yields a consistent drop (−3.8 EM, −4.0 F1). This reflects the role of sentence-level evidence context construction in limiting the influence of redundant or distractor-heavy retrieved text, keeping both subsequent judging and final answering centered on salient evidence rather than raw document concatenation.

4.5.2 Analysis on Sufficiency Predictions

To assess the reliability of the judge's sufficiency signal, we report the confusion matrix of S2G-Judge on HotpotQA (dev) for its binary sufficient/insufficient decisions. Figure 2 indicates that S2G-Judge is unlikely to over-claim sufficiency. Conditioned on predicting sufficient, the judge is correct under Truth in most cases. In particular, the false-positive rate is 6.44%. This behavior is aligned with the role of the sufficiency signal in S2G-RAG, namely reducing answers produced from evidence contexts that are not yet adequate. Meanwhile, the judge tends to require more explicit support in the accumulated context before it declares sufficient. A portion of examples that satisfy Truth are still predicted as insufficient (31.60%). This pattern suggests that the judge is more prone to under-claim sufficiency than to prematurely accept borderline evidence contexts, leaving room for improving calibration so that clearly sufficient contexts are more often recognized as such.

4.5.3 Quantifying Evidence Memory Compression

Figure 3 reports mean context length, grouped by the final turn reached by each example. As retrieval proceeds, the Concatenated Context grows rapidly, increasing from 426 words at Turn 2 to 1,695 words at Turn 5. In contrast, the Evidence Context remains comparatively compact, increasing from 94 words at Turn 2 to 265 words at Turn 5. Overall, the extractor yields a compression ratio of roughly $4.5\times$–$6.4\times$, with larger ratios for instances that require more retrieval turns, where raw-document accumulation introduces more redundancy and distractors.

4.5.4 Teacher–Student Gap under Varying Retrieval Budgets

To examine the teacher–student gap and sensitivity to retrieval budget, we sweep the maximum number of retrieval turns T from 0 to 5 while keeping all other components fixed. All variants share the same answer reasoner and evidence extractor, so differences primarily reflect controller quality.

Figure 4 shows clear gains when increasing T from small budgets, followed by saturation as T grows. Across the full sweep, the trained S2G-Judge closely follows the teacher controller, with only small deviations. This suggests that trajectory distillation captures much of the teacher's control

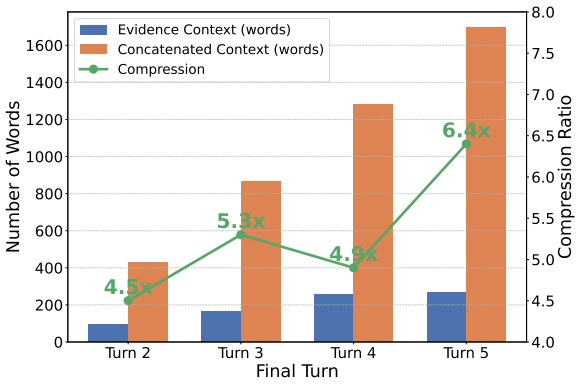

Figure 3: Context compression on HotpotQA (dev), computed on the final row per question ID. Evidence Context accumulates only extractor-selected sentence blocks, while Concatenated Context accumulates raw retrieved documents across turns. We measure context length by whitespace-separated word counts. Bars show mean context length grouped by the final turn; N denotes the number of questions whose trajectories end at that turn. The line reports the compression ratio (Concatenated/Evidence).

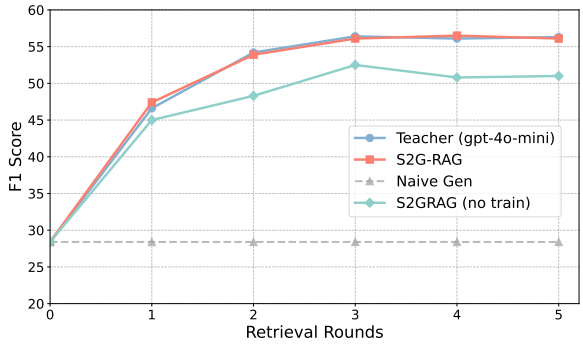

Figure 4: F1 on HotpotQA (dev) under varying maximum retrieval budget T. We compare a teacher controller (GPT-4o-mini), S2G-RAG with the trained S2G-Judge, an unfine-tuned controller (NO TRAIN), and NaiveGen without retrieval. All variants use the same answer reasoner and sentence-level evidence extractor, and the extractor uses the same maximum sentence cap.

behavior in a substantially smaller model.

The diminishing returns at larger budgets are consistent with the structure of multi-hop retrieval. After the trajectory has already accumulated the key supporting evidence, additional turns are less likely to surface genuinely new, answer-relevant facts. In addition, the sentence-level extractor constrains the evidence context to a compact set of salient sentences, so extra retrieved material is increasingly filtered as redundant or weakly related, leading to a flatter accuracy curve.

In contrast, the unfine-tuned controller is consistently worse and exhibits less reliable scaling with larger T, highlighting the importance of distillation for stable sufficiency and gap prediction under intermediate evidence contexts produced by multi-turn retrieval. NaiveGen remains unchanged across budgets, confirming that the improvements come from retrieval and controller behavior rather than additional generation alone.

5 Conclusion

In this paper, we propose S2G-RAG, which turns two key control decisions into explicit, structured outputs. At each turn, a lightweight S2G-Judge assesses whether the current evidence context is adequate and, when it is not, produces structured gap items that describe what information is still needed. These gap items provide a concrete handle for constructing the next retrieval query, reducing reliance on free-form query rewriting and making retrieval trajectories easier to inspect. To keep multi-turn contexts reliable under distractor-heavy retrieval, we complement this control interface with sentence-level evidence extraction that retains a compact, provenance-preserving Evidence Context for subsequent turns and final answering.

Empirically, S2G-RAG improves EM/F1 on TriviaQA, HotpotQA, and 2WikiMultiHopQA under both BM25 and E5 retrieval. Our analysis suggests the judge reduces answers made under insufficient evidence, while sentence-level extraction limits the influence of redundant or distracting text. We hope these findings motivate further work on structured and auditable control mechanisms for iterative RAG systems.

Limitations

We acknowledge several limitations and opportunities for future work. First, recent work has explored learning-based optimization for iterative retrieval and reasoning, including reinforcement learning and actor–critic training that directly optimize system-level objectives through interaction. In contrast, our work emphasizes an explicit and modular control interface based on structured sufficiency and gap prediction learned via trajectory distillation. These directions are complementary. The structured outputs of S2G-Judge could be incorporated into learning-based optimization as supervision, as auxiliary rewards, or as intermediate control variables when improving retrieval policies and evidence selection under practical constraints.

Second, our gap-item schema introduces a representational trade-off. It favors stability and ease of use over expressiveness, and some questions may require richer structure than can be captured by the current fields, such as multi-entity joins, temporal constraints, or compositional relations spanning several intermediate variables. Extending the schema, or inducing more structured intermediate programs while preserving auditability, is a promising direction for broader coverage on complex queries.

Finally, sentence-level evidence extraction introduces a compactness–recall trade-off. While it limits the influence of redundant or distractor-heavy text and improves auditability, it may omit surrounding context that is useful for disambiguation or miss evidence that spans multiple sentences. Moreover, our sufficiency predictions can be conservative, which in practice reduces the risk of answering from borderline evidence states, while leaving room to improve calibration so that clearly sufficient contexts are more often recognized as such. Future work could improve calibration of sufficiency decisions, and adaptive evidence selection strategies that better balance evidence coverage and context focus.

References

Muhammad Arslan, Hussam Ghanem, Saba Munawar, and Christophe Cruz. 2024. A survey on rag with llms. *Procedia computer science*, 246:3781–3790.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection.

Kaustubh D Dhole. 2025. To retrieve or not to retrieve? uncertainty detection for dynamic retrieval augmented generation. *arXiv preprint arXiv:2501.09292*.

Guanting Dong, Jiajie Jin, Xiaoxi Li, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. 2025. Rag-critic: Leveraging automated critic-guided agentic workflow for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3551–3578.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Taeho Hwang, Sukmin Cho, Soyeong Jeong, Hoyun Song, SeungYoon Han, and Jong C Park. 2025. Exit: Context-aware extractive compression for enhancing retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4895–4924.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992.

Zhouyu Jiang, Mengshu Sun, Lei Liang, and Zhiqiang Zhang. 2025. Retrieve, summarize, plan: Advancing multi-hop question answering with an iterative approach. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 1677–1686.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 737–740.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can

be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

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 (volume 1: long papers)*, pages 10014–10037.

Keheng Wang, Feiyu Duan, Peiguang Li, Sirui Wang, and Xunliang Cai. 2025. Llms know what they need: Leveraging a missing information guided framework to empower retrieval-augmented generation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2379–2400.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Diji Yang, Linda Zeng, Jinmeng Rao, and Yi Zhang. 2025. Knowing you don't know: Learning when to continue search in multi-round rag through self-practicing. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1305–1315.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William 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.

Xiaopeng Ye, Chen Xu, Chaoliang Zhang, Zhaocheng Du, Jun Xu, Gang Wang, and Zhenhua Dong. 2025. Q-prm: Adaptive query rewriting for retrieval-augmented generation via step-level process supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 15113–15128.

Linda Zeng, Rithwik Gupta, Divij Motwani, Yi Zhang, and Diji Yang. 2025. Worse than zero-shot? a fact-checking dataset for evaluating the robustness of rag against misleading retrievals. *arXiv preprint arXiv:2502.16101*.

A Additional Results

This appendix reports supplementary results that are omitted from the main paper for space. Unless otherwise noted, we keep the retrieval corpus and evaluation protocol identical to the main experiments, and change only the component stated in each subsection.

A.1 Strong Reasoner Setting

Retriever	TriviaQA		HotpotQA		2Wiki	
	EM	F1	EM	F1	EM	F1
BM25	78.3	84.4	48.3	62.6	48.1	56.8
E5	79.2	85.9	46.8	60.2	48.2	57.0

Table 3: S2G-RAG under a stronger reasoner setting. We replace the answer reasoner and the sentence-level evidence extractor with GPT-4o-mini, while keeping the same retrieval. Results are reported under BM25 and E5 on TriviaQA, HotpotQA, and 2WikiMultiHopQA.

Table 3 isolates the effect of using a stronger closed-source backbone for reasoning and sentence selection. The gains indicate that the proposed control interface remains compatible with stronger downstream components and does not require changes to the retriever or retraining the judge. We report these results in the appendix because the main comparison in Table 1 focuses on a matched open-source reasoner setting for fair baseline alignment.

A.2 Query Construction Variants

Variant	EM	F1	Correct Retrieval
Panel A: Query construction format			
Free-text gap query	41.5	54.5	50.5
Structured gap query (ours; K=1)	**43.3**	**56.5**	**52.7**
Δ (Structured – Free-text)	+1.8	+2.0	+2.2
Panel B: Number of gap items used in BuildQuery *(structured queries)*			
Structured query (K=1)	**43.3**	**56.5**	52.7
Structured query (K=2)	42.3	55.6	52.4
Structured query (K=3)	43.0	55.9	52.5

Table 4: Query construction variants on HotpotQA (dev) under BM25. Panel A compares free-text versus structured gap-based query construction. Panel B varies the number of structured gap items K used by $\mathrm{BuildQuery}$ within the structured setting. Correct Retrieval is reported as a percentage (%) on the final row per question ID. All other components are kept identical.

Table 4 analyzes the impact of query construction in the iterative retrieval loop. In Panel A, we compare two ways of forming the next-turn query when the judge indicates missing information. The *free-text* variant appends a natural-language gap descrip-

Judge backbone	TriviaQA		HotpotQA		2Wiki	
	EM	F1	EM	F1	EM	F1
Llama-3.2-3B-Instruct	77.1	82.6	43.3	56.5	36.8	44.8
Qwen-3-4B-Instruct	78.2	83.2	44.1	56.8	37.5	44.3

Table 5: Effect of the S2G-Judge backbone on end-to-end QA. We swap two LoRA-finetuned judge backbones while keeping the rest of the pipeline fixed (BM25 retriever, query construction, evidence extractor, and answer reasoner: Llama-3-8B-Instruct). Both judges are trained with the same trajectory-distillation procedure.

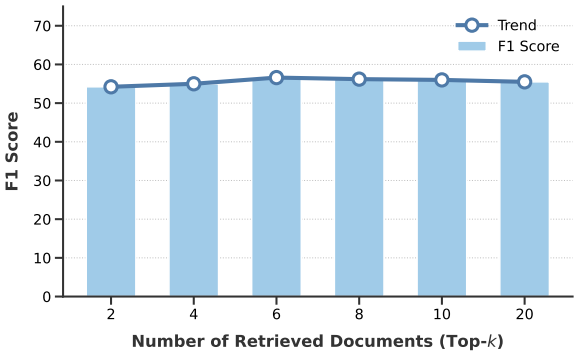

Figure 5: Robustness to per-turn retrieval breadth. We vary BM25 top-k at each retrieval turn ($k \in \{2, 4, 6, 8, 10, 20\}$) while keeping the judge, gap-guided query construction, evidence extractor, maximum turns T, and decoding settings fixed.

tion to the original question, whereas the *structured* variant constructs a query phrase from the missing-fact fields (e.g., `target` and `slot`), yielding a more consistent lexical form for BM25. The structured variant improves EM and F1 by 1.8 and 2.0 points, and increases final-turn Correct Retrieval by 2.2 points, suggesting that structured fields provide stronger retrieval cues over multiple turns.

Panel B varies the number of structured gap items used by BuildQuery. Across $K \in \{1, 2, 3\}$, performance is stable: EM/F1 varies within 1 point and Correct Retrieval remains in a narrow band (52.4–52.7%). Using a single gap item (K=1) yields the best EM/F1, but the differences are modest, indicating that the method is not sensitive to the exact value of K within this range. We therefore default to K=1 as a simple configuration that keeps queries concise while achieving the strongest average accuracy.

A.3 Effect of Judge Backbone

We study the sensitivity of S2G-RAG to the controller backbone by swapping two LoRA-finetuned S2G-Judge variants, Llama-3.2-3B-Instruct and Qwen-3-4B-Instruct, while holding all other components fixed (BM25 retriever, query construction, sentence-level evidence extraction, and Llama-3-8B-Instruct as the answer reasoner).

As shown in Table 5, both backbones yield similar end-to-end accuracy across datasets. Qwen-3-4B provides small gains on TriviaQA and HotpotQA EM with comparable F1, while results on 2WikiMultiHopQA are close. Overall, the structured sufficiency-and-gap interface is compatible with different lightweight controller backbones under the same retrieval setting.

A.4 Robustness to Retrieval Breadth

We assess robustness to retrieval breadth by varying the number of BM25 documents retrieved per turn (top-k), while holding the rest of the S2G-RAG pipeline constant. Figure 5 shows that EM/F1 remains stable across a wide range of k, with only modest fluctuations as retrieval breadth increases.

This behavior is largely attributable to sentence-level evidence management. Rather than appending full documents, S2G-RAG uses a sentence-level Evidence Extractor to form a compact evidence block from the retrieved set before the next round of judging and final answering. As k grows, newly added documents are increasingly redundant or distractor-heavy, so the marginal benefit of wider retrieval tends to diminish. By selecting a bounded set of salient sentences, the extractor limits the amount of irrelevant text that enters the accumulated evidence context, which helps keep both gap identification and answer generation stable under larger per-turn retrieval breadth.

A.5 Effect of Evidence Extraction Cap K_e

The sentence-level Evidence Extractor updates the evidence context by selecting a small set of salient sentences from the newly retrieved documents at each turn. We study how sensitive S2G-RAG is to the extractor cap by varying $K_e \in \{2, 4, 6, 8, 10\}$ on HotpotQA (dev) under BM25, while keeping the judge, query construction, retriever settings, maximum turns, and the answer reasoner fixed.

Table 6 shows that performance is stable across a wide range of caps. EM varies within 43.2–43.4 and F1 within 55.8–56.5, with no clear monotonic trend as K_e increases. This suggests that the system is not tightly tuned to a particular extraction budget. Intuitively, K_e acts as a ceiling rather than a target: the extractor can return fewer sentences

Extractor cap K_e	EM	F1
2	43.2	56.5
4	43.4	55.8
6	43.3	56.5
8	43.4	55.9
10	43.4	56.0

Table 6: Effect of the extractor cap K_e (maximum number of evidence sentences returned per turn) on HotpotQA (dev) under BM25. The extractor may return fewer than K_e sentences. All other components are kept identical.

when retrieval results offer limited relevant content. Moreover, HotpotQA instances are typically supported by a small number of key evidence sentences, so modest caps are already sufficient to preserve the essential two-hop evidence chain while limiting the inclusion of distractor text.

B Training Details

This appendix details the construction of turn-level supervision for S2G-Judge and the LoRA-based supervised fine-tuning procedure. Unless stated otherwise, all unspecified settings use library defaults.

B.1 Turn-Level Supervision Construction

Execution-trace collection. We construct supervision from multi-turn pipeline execution traces produced by the same iterative retrieval loop used at inference time (Appendix C). For each training question, we roll out the pipeline to a fixed maximum budget T and record per-turn snapshots $x_t = (q, C_t)$, where q is the original question and C_t is the accumulated evidence context at turn t.

To assign an initial sufficiency tag for filtering, we use the gold supporting documents. A snapshot is tagged as sufficient if the retrieved titles up to turn t cover all gold supporting document titles for the instance; otherwise it is tagged as insufficient.

To keep the recorded evidence contexts representative of the pipeline's intermediate states, trace collection is driven by an unfine-tuned judge backbone that uses the same prompt and output schema as the final S2G-Judge. This yields evidence contexts that naturally include redundancy and distractors from multi-round retrieval and accumulation.

Teacher labeling. A stronger teacher model, prompted with our instruction template (Appendix E.2), labels each snapshot under a strict context-only evidence constraint. The teacher outputs a binary sufficiency decision together with structured gap items, based only on the information contained in C_t rather than parametric knowledge.

B.2 Supervision Filtering

We apply lightweight filtering to improve the reliability of the distilled turn-level supervision, without introducing any gold signals into model inputs or evaluation:

- **Format checking.** We validate that the teacher output follows the required schema and discard examples with malformed or non-conformant outputs.
- **Conflict-based filtering.** For each snapshot, we compute a coarse, retrieval-derived sufficiency tag based on whether the *retrieved titles* cover the gold supporting page titles. We use this tag only as a weak sanity check to remove *obvious conflicts* between the snapshot state and the teacher label (e.g., cases where the teacher appears to violate the context-only constraint, or the snapshot/tag alignment is unreliable). Importantly, this gold-derived tag is *never* provided to the judge or used at inference time; it is used only to discard a small set of low-confidence supervision pairs.

B.3 Supervision Dataset

The final supervision file consists of per-turn examples containing the original question q, the accumulated evidence context C_t, a binary sufficiency label, and a set of structured gap items.

Dataset statistics. Table 7 summarizes the cleaned supervision set. It contains 2,804 snapshots with a relatively even turn distribution from t=1 to t=4 and approximately balanced sufficiency labels.

Train–validation split. We randomly split the cleaned supervision set into 90% training and 10% validation using a fixed random seed 42, yielding 2,523 training snapshots and 281 validation snapshots.

B.4 LoRA Configuration

We fine-tune S2G-Judge with LoRA (Hu et al., 2022) for parameter-efficient adaptation. Unless otherwise noted, we insert LoRA adapters into the attention and feed-forward projection layers in each decoder block:

Cleaned supervision statistics	
Turn distribution	
t=1	764
t=2	739
t=3	653
t=4	648
Sufficiency labels	
Sufficient	1,396
Insufficient	1,408
Total snapshots	2,804

Table 7: Statistics of the cleaned turn-level supervision set used to train S2G-Judge.

- **Rank** $r = 16$
- **Scaling** $\alpha = 32$
- **Dropout** 0.05
- **Target modules** `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`

B.5 Supervised Fine-Tuning Setup

We run supervised fine-tuning with TRL SFTTrainer on chat-formatted prompts. We use the following hyperparameters:

- **Epochs** 3
- **Learning rate** 1×10^{-4}
- **Gradient accumulation** 8
- **Max sequence length** 2048
- **Precision** FP16
- **Evaluation / checkpointing** evaluate every 200 steps; save every 200 steps; keep the latest 3 checkpoints

After training, we save the LoRA adapter weights and the tokenizer for inference-time loading.

C Pseudocode for the Evaluation Pipeline

Algorithm 1 summarizes our evaluation pipeline for S2G-RAG. Given a question q, the system iteratively maintains an accumulated evidence context C_t that stores only sentence-level evidence blocks extracted from newly retrieved documents. At each turn, the judge inspects (q, C_t) and outputs a sufficiency decision s_t and structured gap items G_t. When s_t indicates the current evidence supports answering (or when the maximum budget T is reached), the reasoner produces the final prediction using only (q, C_t). Otherwise, the predicted gaps are mapped into a retrieval query, documents are retrieved, and the evidence extractor selects a compact set of sentences to append to the evidence context.

Algorithm 1 Evaluation framework of S2G-RAG.

Require: Dev set $\mathcal{S} = \{q_i\}_{i=1}^{N}$;
Retriever $\mathrm{Retrieve}(\cdot)$ over corpus $\mathcal{D}$;
Judge $\mathrm{Judge}(q, C_t) \rightarrow (s_t, G_t)$;
Query builder $\mathrm{BuildQuery}(q, G_t) \rightarrow \tilde{q}_t$;
Evidence extractor $\mathrm{Extract}(q, G_t, D_t) \rightarrow E_t$;
Reasoner $\mathrm{Reason}(q, C_t) \rightarrow \hat{a}$;
Max turns T.
Ensure: Per-turn execution traces and final predictions.

```
procedure EVALUATE(S)
    for each question q ∈ S do
        C_0 ← ∅                    ▷ accumulated evidence memory
        for t = 0 to T do
            (s_t, G_t) ← Judge(q, C_t)
            if s_t = true or t = T then
                â ← Reason(q, C_t)
                Record the final trace at turn t+1
                break
            else
                q̃_t ← BuildQuery(q, G_t)
                D_t ← Retrieve(q̃_t)
                E_t ← Extract(q, G_t, D_t)
                C_{t+1} ← C_t ⊕ E_t
                Record the trace at turn t+1
            end if
        end for
    end for
end procedure
```

D Case Studies

To provide qualitative insight, we report representative multi-turn traces from HotpotQA. Table 8 is a success case: the judge first predicts a relation gap about who Fredric Rieders testified against, which surfaces the correct bridge entity (Michael Swango); it then requests a missing evidence span to verify the remaining constraint (the "three consecutive life terms" and "Florence, Colorado" clue), after which it declares sufficiency and answers correctly.

Table 9 is a failure case dominated by retrieval coverage and ambiguity. Although the trajectory identifies plausible villages in Philipstown, it never retrieves the gold supporting title (Valhalla Highlands Historic District) and instead accumulates evidence for a competing but locally supported district (Cold Spring Historic District). The judge remains insufficient throughout, but the final prediction is still incorrect due to exhausting the retrieval budget without reaching the correct supporting document.

Question: Who did Fredric Rieders tesify agains who had killed as many as 60 patients and recieved three consecutive life terms to be served in Florence Colorado? **Gold answer:** *Michael Swango*. **Prediction:** *Michael Swango* (correct).	
Turn	Trace (Gap → Query intent → Evidence → Decision/Output)
1	**Predicted gap item G_t.** relation; target: *Fredric Rieders*; slot: `testified_against`; desc: The person Fredric Rieders testified against. **Query intent.** Identify who Rieders testified against; match the clue "killed as many as 60 patients". **Evidence. [Fredric Rieders]** "Some of his other well-known cases include … another case is that of Michael Swango, or "Dr. Death"—a serial killer who killed as many as 60 of his patients …" **Decision/Output.** `insufficient`; bridge candidate: *Michael Swango*.
2	**Predicted gap item G_t.** evidence_span; target: *Michael Swango*; slot: `sentence_length`; desc: Michael Swango's sentence length. **Query intent.** Verify "three consecutive life terms" and the Florence, Colorado clue. **Evidence. [Michael Swango]** "He was sentenced in 2000 to three consecutive life terms without the possibility of parole, and is serving that sentence at the ADX Florence supermax prison near Florence, Colorado." **Decision/Output.** `sufficient`; answer: *Michael Swango*.

Table 8: A successful two-hop case. The judge first surfaces a bridge entity via a relation gap (Turn 1), then verifies the remaining constraint via an evidence-span gap (Turn 2).

E Prompts (Judge / Teacher / Extractor)

E.1 S2G-Judge Prompts

E.2 Teacher Prompt

E.3 Sentence-level Evidence Extractor Prompts

Question: What national historic district is located near a village in the town of Philipstown, New York?	
Gold answer: *Valhalla Highlands Historic District*.	**Prediction:** *Cold Spring Historic District* (incorrect).
Turn	Trace (Gap → Query intent → Evidence → Decision/Output)
1	**Predicted gap items** G_t**.** (i) relation; target: *Philipstown, New York*; slot: `village_name`; desc: The name of the village in Philipstown, New York. (ii) relation; target: *village in Philipstown, New York*; slot: `national_historic_district`; desc: The national historic district located near the village in the town of Philipstown, New York. **Query intent.** Identify candidate villages in Philipstown; then look for a nearby national historic district. **Evidence. [Cold Spring, New York]** "Cold Spring is a village in the town of Philipstown . . . "; **[Nelsonville, New York]** "Nelsonville is a . . . village located in the town of Philipstown . . . "; plus many off-topic district pages. **Decision/Output.** `insufficient`.
2	**Predicted gap item** G_t**.** relation; target: *Nelsonville, New York*; slot: `national_historic_district`; desc: The national historic district located near Nelsonville, New York. **Query intent.** Search for a national historic district near Nelsonville (as the inferred village). **Evidence.** Only village-location support repeats; no evidence links Nelsonville to the asked district; the gold title does not appear. **Decision/Output.** `insufficient`.
3	**Predicted gap items** G_t**.** (i) relation; target: *Fish and Fur Club*; slot: `location`; desc: The location of the Fish and Fur Club in relation to the village in the town of Philipstown, New York. (ii) evidence_span; target: *Nelsonville, New York*; slot: `national_historic_district`; desc: A sentence explicitly stating that Nelsonville, New York is a national historic district. **Query intent.** Use nearby landmarks to triangulate the missing district signal. **Evidence. [Fish and Fur Club]** "The original Fish and Fur Club building is at . . . in Nelsonville . . . "; **[J. Y. Dykman Store]** ". . . located . . . in the village of Nelsonville . . . across from the Fish and Fur Club . . . " (useful but off-target). **Decision/Output.** `insufficient`.
4	**Predicted gap items** G_t**.** (i) relation; target: *Fish and Fur Club*; slot: `location`; desc: The location of the Fish and Fur Club in relation to the village in the town of Philipstown, New York. (ii) relation; target: *Fish and Fur Club*; slot: `national_historic_district`; desc: The national historic district associated with the Fish and Fur Club. **Query intent.** Resolve the missing district by following auxiliary entities; fall back to the strongest surfaced village branch. **Final-turn evidence. [Cold Spring, New York]** "Cold Spring is a village in the town of Philipstown in Putnam County, New York, United States." "The central area of the village is on the National Register of Historic Places as the Cold Spring Historic District . . . " **Decision/Output.** `insufficient`; predicted answer *Cold Spring Historic District* (incorrect).

Table 9: Failure case. The trajectory identifies villages in Philipstown but never retrieves the gold supporting title *Valhalla Highlands Historic District*. It instead converges on a locally supported but incorrect district (*Cold Spring Historic District*) surfaced in the final turn.

System prompt for S2G-Judge

```
You are a QA/RAG sufficiency judge.
Given a QUESTION and a CONTEXT (documents retrieved so far), decide whether the CONTEXT alone contains enough
information to reliably answer the QUESTION. If not, list the gap items that describe what information is still missing.
Output format (strict):
- Output exactly one JSON object and nothing else.
- The output must have exactly two keys: "sufficient" and "gap items".
Schema:
"sufficient": true/false,
"gap items": [
"category": "bridge entity | attribute | relation | evidence span | other",
"target": "string",
"slot": "string",
"description": "string"
]
Constraint:
- If "sufficient" is true, then "gap items" must be an empty list [].
```

Table 10: System prompt for S2G-Judge. The judge outputs a binary sufficiency decision and, when insufficient, structured gap items.

Teacher prompt (Part I: task definition and labeling rules)

You are a QA/RAG sufficiency judge.
You will be given:
- a QUESTION
- the current-round CONTEXT (what the QA system has retrieved so far)
Your tasks:
1. Decide whether the given CONTEXT alone contains enough information to reliably answer the QUESTION.
- Answer using the boolean field "sufficient".
IMPORTANT:
- You MUST base your decision ONLY on the CONTEXT.
- Even if you personally know the correct answer from world knowledge or training data, if the CONTEXT does not clearly support a correct answer, you MUST set "sufficient": false.
- Only when the CONTEXT itself already provides enough explicit evidence to justify a correct answer, you may set "sufficient": true.
2. If the information is NOT sufficient (i.e., "sufficient": false), list the gap items needed to answer the QUESTION.
Use the field `gap_items`: a list of 1–3 objects, each with:
- "category": one of ["bridge entity","attribute","relation","evidence span","other"]
- "target": a short string naming the entity or concept this gap is about
- "slot": a coarse type of the missing fact (e.g., alias, name, nationality, birth place, location)
- "description": a short English phrase describing the missing information
Guidelines:
- Only mark gap items as missing if they are NOT explicitly stated in the CONTEXT.
- Think about the reasoning chain and identify which links are still missing.
- Avoid vague statements like "need more information" without specifying what is missing.
Surface forms and aliases:
- If QUESTION uses one name (A) and CONTEXT introduces an alias (B), create ONE gap item with target A / B and descriptions that mention both forms.
- Do NOT create two separate gap items for the same underlying relation just due to surface forms.
3. If the information IS sufficient (i.e., "sufficient": true), set `gap_items`: [].

Table 11: Teacher prompt used to label per-turn snapshots for judge training (Part I).

Teacher prompt (Part II: output format and example)

You MUST respond with a single JSON object and NOTHING else.
The JSON MUST have exactly the following shape:
```
"sufficient": true/false,
"gap items": [
    { "category": "...", "target": "...", "slot": "...", "description": "..." },
    ...
]
```
Do NOT add any extra keys.
Do NOT add explanations outside the JSON.
EXAMPLE (for style ONLY; alias is already present in CONTEXT)
QUESTION: "What nationality was A's wife?"
CONTEXT: (... includes "A is better known as B", but does NOT mention wife's name or nationality)
Output:
```
"sufficient": false,
"gap items": [
    { "category": "relation", "target": "A / B", "slot": "spouse name",
        "description": "The name of A (B)'s wife." },
    { "category": "attribute", "target": "A (B)'s wife", "slot": "nationality",
        "description": "The nationality of A (B)'s wife." }
]
```
QUESTION
<q>
CONTEXT (this is the ONLY evidence you may use)
<ctx>
Now output ONLY the JSON object:

Table 12: Teacher prompt used to label per-turn snapshots for judge training (Part II).

System prompt (Evidence Selector)
You are a sentence-level evidence selector in a multi-hop RAG pipeline. You will be given an ORIGINAL QUESTION, MISSING FACTS describing what information is still needed, and a numbered list of SENTENCES from retrieved documents. Treat the MISSING FACTS as reliable guidance. Selection priorities: (1) Select sentences that directly help answer the ORIGINAL QUESTION. (2) If budget remains, select sentences that are highly relevant to the MISSING FACTS, including intermediate bridge facts that may enable the next hop. Output format (strict): - Output exactly one JSON object and nothing else. - The JSON must have exactly one key: "evidence global ids". Example: "evidence global ids": [1, 5, 7] Constraints: - "evidence global ids" is an array of integers. - Select at most K sentences (K is provided in the user message). - Only use ids that appear in the numbered sentence list. - Do not rewrite, paraphrase, or generate any evidence text.

Table 13: System prompt for the sentence-level Evidence Extractor. The extractor outputs sentence indices (pointers) rather than rewriting evidence text.