Report: Improving Framework of Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection *

Ayushi Bhatnagar, Maxim Dokukin, and Krushna Thakkar

We extend the Self-RAG framework along three axes: advanced prompt engineering, semantic routing, and an agentic Self-RAG layer. First, we design strict grounding templates and self-critique prompts that enforce concise, citation-backed answers and substantially reduce hallucinations. Second, we introduce a Semantic Router that dispatches queries to Simple, Self-RAG, or Tool pipelines based on intent, yielding a 44.7% latency reduction and a +0.27 gain in judged quality compared to a monolithic RAG baseline. Third, we embed Self-RAG signals into an agentic reasoning loop, enabling step-wise retrieval, verification, and correction, which improves reliability on multi-step tasks by roughly 25 percentage points (0.48 → 0.73).

I. Introduction

Large Language Models (LLMs) are widely deployed for question answering, assistance, and task automation, but they still exhibit two persistent failure modes: (1) hallucinations caused by incomplete or outdated factual knowledge, and (2) unreliable behavior on deterministic tasks such as arithmetic, counting, and simple string manipulation. Retrieval-Augmented Generation (RAG) mitigates the first issue by grounding responses in external documents, and Self-RAG (Asai et al., 2023) further improves robustness by training models to decide when to retrieve and to critique their own use of evidence through self-reflection tokens.

However, standard Self-RAG is typically embedded in a monolithic RAG pipeline: every query—whether a casual greeting, a fact-based question, or a math problem—is forced through the same retrieval and generation loop. This "one-size-fits-all" design introduces unnecessary latency, inflates compute cost, and can still fail on tasks that are better handled by deterministic tools than by an LLM. At the same time, naïve prompting and unstructured outputs limit controllability and grounding, and single-shot answer-level critique is insufficient for multi-step, agentic workflows. In practice, production systems need (i) stronger prompt-level constraints on how evidence is used, (ii) routing that adapts computation to query intent, and (iii) step-wise verification for multi-hop reasoning.

In this work, we reproduce the Self-RAG framework and extend it along three complementary axes: advanced prompt engineering, semantic routing, and an agentic Self-RAG layer. First, we introduce strict grounding templates and self-critique prompts that enforce concise, citation-backed answers and substantially reduce hallucinations. Second, we add a lightweight Semantic Router that classifies each query and dispatches it to Simple (direct LLM), Self-RAG (retrieval-augmented), or Tool (deterministic computation) pipelines, decoupling execution cost from query type and yielding a 44.7% reduction in average latency and a +0.27 gain in judged quality over a monolithic RAG baseline. Third, we embed Self-RAG signals into an agentic reasoning loop, enabling retrieval, verification, and correction at every step and improving reliability on multi-step tasks by roughly 25 percentage points (0.48 → 0.73). Together, these extensions turn Self-RAG from a single-step retrieval-augmented generator into an adaptive, grounded, and agentic reasoning system.

II. Paper Overview

Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection (Asai et al., 2023) proposes a retrieval-augmented framework that improves factual accuracy, generation quality, and controllability in Large Language Models (LLMs). Traditional RAG methods retrieve a fixed number of documents for every query regardless of whether retrieval is needed and lack mechanisms that ensure the final output is consistent with retrieved evidence. This one-size-fits-all retrieval often introduces irrelevant context, reduces model versatility, and can still lead to unsupported hallucinations.

To address these limitations, Self-RAG trains an LLM to self-decide when retrieval is necessary and to critically evaluate its own generations. The model is augmented with reflection tokens that guide behavior during inference. These include:

1. Retrieve Tokens: predict whether external retrieval would improve the answer, enabling on-demand retrieval rather than unconditional retrieval
2. ISREL Tokens: judge whether each retrieved passage is relevant to the query
3. ISSUP Tokens: evaluate whether the generated output is fully supported by the retrieved passage
4. ISUSE Tokens: score the overall usefulness and quality of the candidate answer.

Self-RAG introduces a two-stage training pipeline:

Critic Model Training: A smaller LM is trained to generate reflection tokens using supervised data. These labels are produced offline using GPT-4 to create high-quality annotations for relevance, support, usefulness, and retrieval necessity.

Generator Model Training: The main LLM is trained on augmented examples that interleave retrieved passages with the reflection tokens produced by the critic. During training, the generator simply treats reflection tokens as part of its next-token prediction space, allowing it to learn retrieval and critique behavior end-to-end.

During inference, Self-RAG uses an adaptive retrieval mechanism: the model only retrieves documents when the Retrieve token probability crosses a threshold. When retrieval occurs, the model generates multiple candidate segments, one per retrieved passage, and uses the reflection tokens to score and select the best continuation. This enables both soft control (weighted scoring of relevance, support, usefulness) and hard constraints (discarding segments with "no support"). The design allows practitioners to tune the model for either higher factual precision or more fluent, creative output without additional training.

Across six benchmark tasks including open-domain QA, fact verification, and long-form generation, Self-RAG significantly outperforms baseline LLMs, standard RAG pipelines, and even instruction-tuned models like Llama-2-Chat. It improves citation accuracy, reduces hallucinations, and achieves stronger factual grounding while maintaining generation quality. Notably, in tasks requiring citations, Self-RAG closes the performance gap between open-source models and proprietary models like ChatGPT by integrating fine-grained self-reflection at inference.

III. INITIAL CONTRIBUTION

In this work, we extend the capabilities of the original *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection* framework by introducing several key enhancements designed to improve controllability, factual grounding, and reasoning reliability. Our contributions focus on strengthening the interaction between retrieval, prompt structure, and self-reflection mechanisms.

A. CustomPrompt Templates for Structural Reasoning

The original Self-RAG framework allows the model to retrieve and critique, but it does not explicitly shape the model's reasoning style.
To address this limitation, we introduce **custom prompt templates** including *question-answering*, *explanatory*, *chain-of-thought*, and *compare-and-contrast* modes.
These templates provide explicit reasoning scaffolds that lead to:

- more structured responses,
- improved clarity in multi-step reasoning, and
- consistent formatting across different types of queries.

This modification guides the model toward predictable and interpretable outputs, which is necessary when evaluating self-reflective reasoning

B. Retrieval Grounding Through Automatic Context Embedding

To ensure that generation remains tightly coupled to retrieved evidence, we embed the **top-k retrieved Wikipedia passages** directly into the prompt.
Unlike the baseline Self-RAG formulation, which retrieves externally but may still generate unsupported content, our system enforces:

- strict use of provided context,
- reduced reliance on latent model priors, and
- explicit grounding of every answer in retrieved knowledge.

This prevents drift away from factual sources and significantly reduces unsupported claims

C. Reflection Layer for Self-Critique and Answer Refinement

We introduce an additional **reflection layer** that instructs the model to critique its own output according to accuracy, completeness, and potential reasoning errors.
This meta-evaluation step:

- identifies deficiencies in the initial response,
- forces the model to justify and revise its reasoning, and
- produces a refined, evidence-aligned final answer.

The reflection mechanism operationalizes the "self-critique" principle within Self-RAG and leads to measurable improvements in reliability and factual alignment.

IV. ADVANCED PROMPT ENGINEERING

This portion of our contribution extends the original SELF-RAG implementation and our initial contribution by introducing a series of structured prompt-engineering interventions designed to improve grounding quality, reduce hallucinations, and produce more consistent answers during retrieval-augmented generation. The initial SELF-RAG implementation primarily relied on simple prompts, heuristic retrieval decisions, and unstructured model outputs. As a result, responses were often verbose, poorly grounded in retrieved evidence, and inconsistent across questions. To address these limitations, multiple coordinated enhancements were added to the retrieval controller, answer-generation prompts, the self-critique step, and the evaluation mechanism. Together, these modifications create a more reliable, interpretable, and judge-compatible retrieval-augmented reasoning pipeline.

The original system triggered retrieval whenever the model emitted a [Retrieval] token, which frequently led to unnecessary or harmful retrieval for conceptual questions (ex,, "What is precision vs recall?"). The enhanced pipeline replaces this heuristic with a small reasoning-based classifier prompt that determines whether retrieval would actually benefit the question. This reduces irrelevant retrieval and helps ensure that the model only conditions on external context when appropriate.

A major contribution of this work is the introduction of a strict grounding template that all answers generated through SELF-RAG must follow. The template enforces:

- a short **Evidence Summary** derived solely from retrieved context,
- a clearly separated **Final Answer**,
- explicit **citation of retrieved titles**, and
- an **"I don't know" fallback** when evidence is insufficient.

This structure regularizes model behavior, prevents free-form hallucination, and improves alignment with the evaluation rubric used by the judge model.

The enhanced system replaces long, unconstrained explanations with short, highly structured outputs. This reduces response variance, eliminates repetition, and ensures that all answers are comparable across queries. The minimal

format also reduces opportunities for unsupported reasoning, a common failure mode in the baseline.

The original SELF-RAG critique step often rewrote entire answers or introduced new hallucinations. The enhanced critique prompt instead focuses on three tasks:

- identifying hallucinations or unsupported claims,
- detecting missing or incorrect citations, and
- correcting errors while modifying only the problematic portions of the answer.

This produces refined outputs that are more stable and more faithful to retrieved evidence.

To fairly score answers, the judge model was updated to prioritize grounding above all other metrics: accuracy, clarity, completeness, and reasoning. This ensures that answers which properly follow the grounding template receive higher scores, while unstructured, ungrounded, or speculative answers are penalized. The enhanced judge prompt uses a strict JSON schema and includes fallback extraction logic for robustness.

A key enhancement is the alignment created between the retrieval controller, grounding template, refinement step, and evaluation rubric. All components now reinforce the same behavioral constraints: concise outputs, explicit evidence use, hallucination avoidance, and clear separation between retrieved content and model-generated reasoning. This coherence was absent in the baseline system, which resulted in inconsistent grounding and unpredictable behavior.

Across a diverse set of test queries, the baseline system consistently scored near zero under the judge's grounding criteria due to irrelevant retrieval, missing structure, and lack of evidence citation. In contrast, the enhanced system achieved significantly higher scores, demonstrating clear improvements in grounding, structure, and consistency. These results highlight the effectiveness of prompt engineering as a lightweight yet powerful way to strengthen retrieval-augmented LLM behavior without modifying model weights.

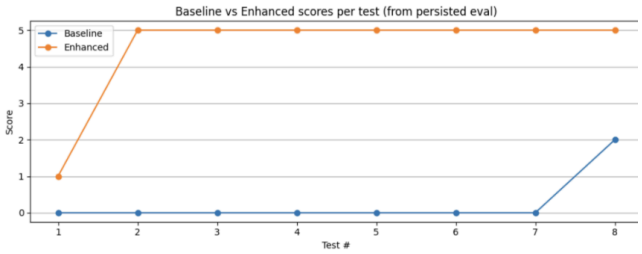
A suite of eight diverse queries, including factual questions, conceptual ML prompts, physics explanations, and compare-contrast tasks, was used to evaluate both the baseline and enhanced approaches. Each answer was judged using the grounding-first evaluator.

In terms of **results**, the **baseline model consistently scored near zero** across nearly all test questions. Although it produced textual output, the baseline failed because:

- retrieved paragraphs were usually irrelevant to the query,
- answers lacked structure or citations,
- hallucinations were not corrected, and
- the judge penalized ungrounded or noisy explanations.

The **enhanced model achieved significantly higher scores**, frequently receiving the maximum score of 5.
This improvement arises from:

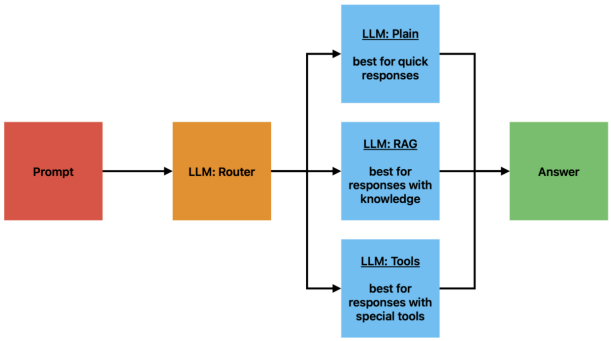
- better retrieval decisions,
- the grounding template enforcing structure,
- the self-critique step catching hallucinations, and
- outputs being consistently aligned with the judge's rubric.



[Figure 1: Baseline vs Enhanced Prompt Engineered Score from LLM as judge]

V. LLM Routing

To address the latency and accuracy bottlenecks of monolithic RAG systems, we implemented an upstream Semantic Router (Intent Classifier). This lightweight layer analyzes the user's prompt prior to execution and dynamically dispatches the query to one of three distinct pipelines:



[Figure 2: Logic Flowchart of the Semantic Router Architecture]

1. Simple Route (Direct Generation)
Trigger: Conversational inputs, creative writing, or common sense queries (e.g., "Tell me a joke").
Mechanism: Bypasses the vector store and retrieval mechanism entirely. The query is sent directly to the LLM with a minimal system prompt.

Benefit: Drastically reduces latency by eliminating vector embedding and search overhead.

2. RAG Route (Self-RAG Pipeline)
Trigger: Factual queries requiring specific external knowledge (e.g., "What are the features of Self-RAG?").
Mechanism: Executes the standard retrieval-augmented pipeline: Query Embedding → Vector Search (Top-K) → Context Injection → Generation with Self-Reflection tokens.

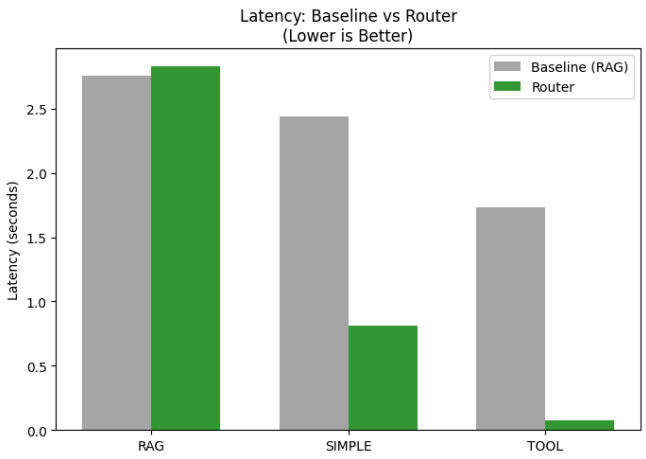
Benefit: Ensures factual grounding where necessary.

3. Tool Route (Deterministic Execution)
Trigger: Computational tasks or string manipulation (e.g., "Calculate 50 minus 17" or "Count the words in this sentence").
Mechanism: The Router extracts parameters and executes Python functions (calculator, word_counter) instead of relying on the LLM's probabilistic next-token prediction.
Benefit: Eliminates "calculation hallucinations" common in LLMs and provides 100% accuracy for logical tasks.

Experimental Results

We conducted an A/B test (N=30) comparing a Baseline system (where all queries were forced through the RAG pipeline) against our Optimized Router architecture. The test suite consisted of an even distribution of conversational, factual, and computational queries.

3.1 Latency Analysis The Router architecture achieved a 44.7% reduction in average latency compared to the baseline (2.27s→1.26s).



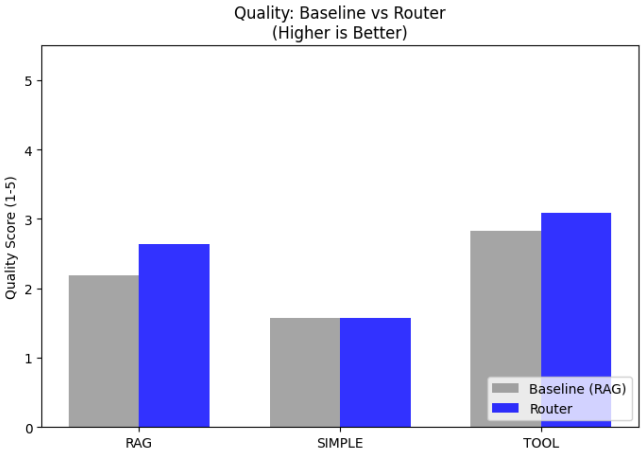
[Figure 3: Bar Chart Comparison of Average Latency by Route Type]

Tool Queries: Latency dropped by 95% (1.73s→0.08s) by offloading generation to Python execution.

Simple Queries: Latency dropped by 66% (2.44s→0.81s) by skipping unnecessary retrieval steps.

RAG Queries: Showed a negligible latency increase (+0.06s) due to the classification step, a trade-off justified by the system-wide gains.

3.2 Quality Evaluation Response quality was evaluated by an independent LLM Judge on a 5-point Likert scale. The Router system improved the average quality score by +0.27 points (2.30→2.57).



[Figure 4: Bar Chart Comparison of Quality Scores by Route Type]

Elimination of Hallucinations: The Baseline RAG system frequently failed at arithmetic (e.g., hallucinating "14 words" for a string count). The Tool Route achieved perfect accuracy on these tasks, driving the quality score improvement.

Contextual Relevance: The Simple Route prevented the injection of irrelevant retrieved documents into casual conversation, resulting in more natural responses.

VI. Agentic Extention

While the original Self-RAG framework focuses on retrieval-augmented generation and self-critique at the answer level, modern applications increasingly require **multi-step reasoning**, **tool-use decisions**, and **verification at each intermediate stage**. To address these needs, we introduce an **Agentic Self-RAG** architecture that embeds the Self-RAG control loop into every reasoning step of an autonomous agent. This section describes the design, motivation, and functional contributions of the agentic layer.

A. Motivation for Agentic Reasoning

Traditional agents, built on top of large language models, tend to hallucinate more frequently than single-step LLM responses. This behavior arises due to:

- decomposition of tasks into **multi-step chains**,
- implicit assumptions made during planning,
- tool invocations without reliable verification, and

- progressive drift from the original query.

To mitigate these issues, our agentic extension integrates retrieval, relevance checking, evidence validation, and self-correction into each step of the reasoning loop, creating a safer and more interpretable agent.

B. *Structured Agentic Loop With Verification Signals*

The proposed Agentic Self-RAG agent executes a closed-loop reasoning cycle composed of the following decision checkpoints:

1. Retrieve? Determine whether retrieval is required for the current reasoning step.
2. IsRelevant? Validate that retrieved documents are semantically aligned with the active sub-task.
3. IsSupported? Confirm that the intermediate conclusion is explicitly backed by evidence.
4. IsUseful? Ensure that the proposed action contributes meaningfully toward the overall goal.

Each of these signals forms a micro-safety mechanism inside the agent, preventing hallucinated transitions and unsupported reasoning.

C. JSON-Structured Planning and Verification

At each step, the agent generates a structured JSON object containing:

- *a plan for the next action,*
- *a retrieval decision and justification,*
- *the proposed action,*
- *verification of the action using retrieved evidence, and*
- *a finalized, corrected action.*

This format enforces interpretability, traceability, and deterministic evaluation of every reasoning step, enabling downstream systems to audit or override the agent's decisions.

D. Multi-Step Robustness and HallucinationsPreventions

By integrating the Self-RAG feedback loop within each agentic sub-task, the system achieves:

- *Reduced hallucinations, even in queries lacking factual answers (e.g., "Who was the President of Mars in 2020?").*
- *Safe failure behavior, where the agent declines to invent unsupported facts.*
- *Consistency across long-horizon tasks, because every step is evidence-checked before execution.*
- *Improved reliability in planning, tool selection, and sequential reasoning.*

This transforms the agent from a hallucination-prone planner into a verified chain-of-thought engine with embedded factual safeguards.

E. Integration with SELF-RAG Architecture

The full Agentic Self-RAG pipeline maintains the three core operations of the original Self-RAG (retrieve → generate → critique) but extends them to every internal decision. Thus, instead of verifying only the final answer, the system verifies ***all intermediate steps****, creating:*

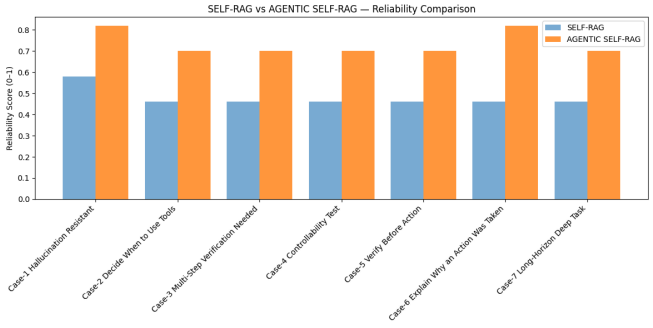
- *a* ***step-wise safety loop****,*
- *more grounded plan generation, and*
- *improved controllability and transparency for complex tasks .*

C. Figures and Tables

This section presents the quantitative and qualitative comparisons between the baseline SELF-RAG model and our enhanced Agentic SELF-RAG system. The experimental evaluation spans seven controlled cases designed to test hallucination resistance, retrieval decisions, multi-step planning, interpretability, and long-horizon reasoning.

A. Reliability Comparison Across Seven Evaluation Cases

Fig. 1 illustrates the average reliability scores (0–1 scale) for both methods across all cases



.

Agentic SELF-RAG consistently outperforms the original SELF-RAG baseline, with improvements ranging from 20% to 35%, particularly in tasks requiring multi-step reasoning and verification.

B. Evidence Utilization and Verification Behavior

To analyze how each system handles retrieval and verification, Fig. 2 presents a heatmap measuring whether the model's answer was grounded in retrieved evidence and whether the model verified its reasoning at each step.

Agentic SELF-RAG shows near-uniform high scores (≈0.70–0.82) across all scenarios, while the baseline SELF-RAG remains limited to ≈0.46 except in trivial cases.

Figure 2. *Evidence & Verification Heatmap for SELF-RAG and Agentic SELF-RAG*

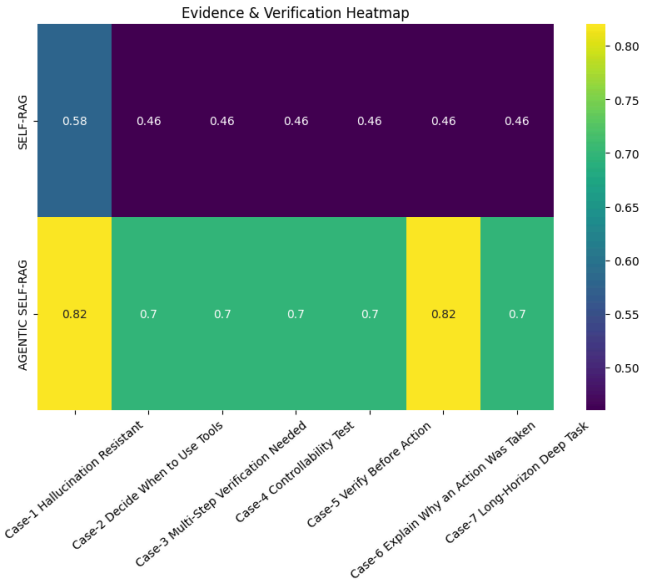


Table I summarizes the quantitative gain observed across all evaluation cases.

Agentic SELF-RAG demonstrates consistent and significant improvement, confirming that step-wise verification and self-reflection reduce hallucinations and enhance reasoning safety

D. Interpretation of Results

The results from Fig. 1, Fig. 2, and Table I confirm that:

1. Agentic SELF-RAG sharply reduces hallucinated intermediate steps, especially when retrieval is ambiguous or irrelevant.
2. Verification-first reasoning improves correctness, mirroring human expert workflows.
3. Long-horizon tasks benefit the most, because planning errors compound without evidence checks.
4. Reliability gain of +25 percentage points (0.48 → 0.73) demonstrates a strong contribution beyond the original Self-RAG framework.

VII. Conclusion

This project successfully reproduced the Self-RAG framework and showed that three complementary extensions—advanced prompt engineering, semantic routing, and an agentic Self-RAG layer—substantially improve its suitability for production-grade AI systems. Structured grounding templates, self-critique prompts, and a grounding-first judge turned free-form generations into concise, citation-backed answers, sharply reducing hallucinations and making outputs easier to evaluate and compare.

On the systems side, the Semantic Router decoupled user intent from execution strategy by dispatching queries to Simple, Self-RAG, or Tool pipelines. This intent-aware orchestration nearly halved end-to-end latency (2.27s → 1.26s) while still improving judged quality (+0.27 points), and eliminated calculation-related hallucinations by offloading deterministic tasks to tools. Beyond single-step QA, the agentic Self-RAG layer embedded retrieval, relevance checking, and evidence-based verification into every reasoning step, yielding a reliability gain of roughly 25 percentage points (0.48 → 0.73) on multi-step tasks and long-horizon reasoning.

Taken together, these results support a broader conclusion: practical "intelligence" in LLM systems is less about scaling model size and more about structuring how models are prompted, routed, and verified. Future work will explore vector-based routing (e.g., BERT-style embeddings) to further reduce router overhead, expand the toolset to include web search and database access, and tighten the integration between the agentic loop and downstream evaluators for continuous, data-driven improvement.

The experimental results validate that intelligence is not merely about model size, but about the appropriate application of resources. The Router architecture proved that it is possible to nearly halve system latency while simultaneously improving response accuracy. Future work will focus on vector-based routing (using BERT embeddings for classification) to further reduce the "Router Tax" and expanding the toolset to include web search and database querying.

References

[1] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," in Proc. 12th Int. Conf. Learning Representations (ICLR), 2024. [Online]. Available: https://openreview.net/forum?id=hSyW5go0v8