INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 8, August 2026 | ISSN: 2456-4184 | IJNRD.ORG

A Survey of Retrieval-Augmented Generation for Small Language Models :

Barriers, Solutions, and Open Challenges.

[1]Kinal Patadiya, [2]Dr. Priti Patel

[1]Student, [2]Assistant Professor

Department of Computer Science

Shree Ramkrishna Institute of Computer Education and Applied Sciences, Surat, India

➢ ABSTRACT

Large Language Models (LLMs) have revolutionized NLP applications by performing tasks such as question answering, summarization, and reasoning, all of which rely on amassing a large volume of knowledge within their parameters. They carry an inherent caveat, however: they learn information at training time and then "freeze" it afterward, so that when something changes, they must be retrained. Retrieval-augmented generation (RAG) solves this by linking the model to an external source of documents and letting it retrieve the relevant ones when answering a query.

Many current RAG studies assume an LLM-scale reader. This survey instead examines what happens when the reader is a Small Language Model (SLM), in the range of 0.5 to 7 billion parameters, which can fit into on-device assistants, private enterprise pipelines, or latency-focused systems where large models are too slow to be useful. SLM-RAG is well motivated because SLMs have fewer weights and are therefore more easily helped by retrieval. On knowledge-heavy tasks, retrieval-augmented smaller models have been shown to outperform much larger closed-book models.

However, pipelines designed for LLM output fail in characteristic ways at SLM scale: contexts fill rapidly under tight context windows, retrieval noise erodes performance far more severely, and grounded hallucination persists even when correct evidence is present.

This survey classifies the emerging solutions into five groups: retrieval-side filtering, generator-side training, architectural redesigns, efficiency techniques, and collaborative SLM–LLM systems. For each, we consider the particular barriers it addresses, whether it carries hidden dependencies on large models that undercut its claim to be SLM-native, and where meaningful gaps remain.

Index Terms—retrieval-augmented generation, small language models, on-device inference, knowledge-intensive NLP, edge AI, RAG pipeline, SLM-RAG.

I. INTRODUCTION

1.1 Background and Motivation

Large language models have moved from research laboratories into everyday infrastructure. GPT-4, Claude, and Gemini now sit behind customer service pipelines, code assistants, and document analysis tools. This capability depends on parametric memory acquired during pre-training, which is fixed once training ends. Any knowledge that postdates the training cutoff, or that was not well represented in the training corpus, is simply not available to the model. Retraining to incorporate updates is expensive and, in practice, rarely done more than a few times a year.

Three problems follow from this architecture. Knowledge becomes stale: a model trained on data with a 2024 cutoff cannot answer questions about events in 2025 without intervention. Models hallucinate, generating fluent but incorrect text when queried about information they do not actually possess. And generations are not attributable, because the model does not cite sources and users cannot verify where a claim came from.

Retrieval-augmented generation, introduced by Lewis et al. in 2020[1], addresses all three problems through a structural change. Instead of relying on the generator to remember everything, a RAG system retrieves relevant documents from an external corpus at inference time and prepends them to the prompt. The generator then conditions on both its parametric knowledge and the retrieved context. Knowledge can be updated by modifying the external corpus rather than retraining the model. Hallucinations can be reduced by grounding generation in retrieved evidence. And generations can be attributed to specific retrieved passages, providing a transparency property that pure parametric generation lacks [5].

RAG has since become a standard component of production LLM deployments. The Huang and Huang survey[2] documents this shift: retrieval is now the default approach to knowledge-intensive tasks in commercial systems, and the space of RAG techniques has grown rapidly since 2020. Most of this work assumes an LLM-scale reader. The question this survey addresses is what happens when the reader is an order of magnitude smaller.

1.2 The Small Language Model Regime

The spotlight has lately been on the "LLM journey," but another evolution is quietly shaping how language models reach end users. Since Nguyen et al.[3], small language models (SLMs)—typically ranging from 0.5B to 7B parameters—have emerged as the go-to solution for on-device usage, edge deployment, and privacy-sensitive enterprise applications.

This trend is driven by three factors. **Cost:** a 3B model running on a local GPU costs essentially nothing per query, whereas a single GPT-4 query costs on the order of a cent; across millions of queries per day, that becomes a significant line item. **Latency:** a cloud API round-trip can take from 500ms to several seconds, while an on-device SLM can respond within tens of milliseconds—a difference that is clearly noticeable in interactive applications. **Data privacy and sovereignty:** third-party APIs often lack the legal or contractual authority to process sensitive data, and in verticals such as healthcare, finance, and government, the need for local processing cannot be ignored.

Phi-3[6], Gemma 2[11], Qwen2.5[12], and TinyLlama[13] are representative examples of the current generation. Apple Intelligence[14] ships roughly 3B-parameter models on consumer devices, swapping task-specific LoRA adapters at runtime. More recent additions include Phi-4-mini, Gemma 3, and Qwen3. A broader survey of the SLM design space is given by Subramanian et al.[4] Table 1 summarizes the models most relevant to RAG.

SLMs carry one well-documented structural weakness: they hold less parametric knowledge than larger models. The Phi-3 authors state this directly, noting that phi-3-mini cannot store much factual knowledge and recommending retrieval augmentation as the remedy[6]. This is not a training deficiency but a property of scale. A 3.8B model has roughly 2% of GPT-4's parameter count, and a substantial fraction of the factual knowledge GPT-4 holds simply cannot fit in a model that small.

The Huang and Huang survey[2] quantifies this: a 7B model with retrieval outperforms a 70B model without it on knowledge-intensive benchmarks. Ten times fewer parameters, better results—the right information supplied at the right time. When the teams building small models point to retrieval as the fix and the benchmarks agree, SLM-plus-RAG is not a fallback but a practical strategy for compensating for limited parametric capacity.

1.3 Why LLM-Era RAG Does Not Transfer

Most published RAG pipelines were designed and tested with GPT-3.5 and GPT-4 class models as the reader, and they are built around assumptions that do not always hold at SLM scale. A well-designed LLM-RAG pipeline expects the generator to handle long contexts without getting confused, filter out irrelevant passages on its own, reason across multiple retrieved documents, and produce properly structured output. These are reasonable expectations for a large model. At 1 to 3 billion parameters, however, these assumptions start to weaken — and in some cases they break down entirely.

MiniRAG[7] provides direct evidence. LightRAG[34], a graph-based RAG system, scores 56.90% accuracy with gpt-4o-mini as the reader; swapping in Phi-3.5-mini on the same benchmark drops accuracy to 39.81%. GraphRAG[36] fails outright with SLMs as readers because its indexing pipeline produces broken knowledge graphs at SLM scale. Section 3 examines these failures in detail.

The failure modes are characteristic, not incidental. Three patterns recur: retrieved documents overflow the short context windows SLMs work with; small models absorb noise from retrieved documents rather than filtering it out; and

techniques that work reliably at 7B scale (chain-of-thought prompting, self-refinement, iterative query decomposition) degrade at 1 to 3 billion parameters.

1.4 Scope and Contributions

This survey addresses retrieval-augmented generation for models in the 0.5B–7B parameter range. Its scope is deliberately narrower than that of recent general RAG surveys[2], which encompass the full LLM-RAG landscape, and than that of SLM surveys[3], which treat small models without a specific focus on retrieval. The present work is positioned at their intersection, namely SLM+RAG, where it aims to contribute.

Out of Scope. Several topics are excluded, for three reasons. First, LLM-era RAG is considered only insofar as it motivates SLM-era developments, as comprehensive treatments already exist[2]. Second, multimodal RAG—retrieval over images, video, or audio—entails technical challenges that fall outside the scope of a text-centric SLM-RAG survey. Third, code RAG constitutes a distinct body of research with dedicated benchmarks and is therefore not examined here. Finally, agentic RAG systems introduced after early 2025 are referenced where pertinent but not treated in depth, owing to the rapid evolution of that subfield.

Contributions. This survey offers three contributions. First, it consolidates reported failure cases arising from the application of LLM-based RAG pipelines to SLMs, classified into six categories: (1) context-window effects, (2) noise-sensitivity effects, (3) retrieval-indexing mismatch, (4) reasoning-depth limitations, (5) agentic-method failures, and (6) evaluation gaps. Second, it proposes a taxonomy of solutions organized by point of intervention: retrieval-side filtering, generator-side training, architectural redesign, efficiency methods, and SLM–LLM collaboration. Third, it critically examines the persistent reliance on LLM teachers for training data among methods described as SLM-native—a dependency that undermines their claims to full efficiency.

Literature Selection. The reviewed papers were drawn from arXiv and the ACL Anthology, together with additional venues including NeurIPS, ICLR, and NAACL, covering the period from the original formulation of RAG in 2020 through mid-2025. A paper was deemed eligible if it proposed, analyzed, or evaluated a RAG technique relevant to small-model deployment—spanning retrieval, generation, training, efficiency, or evaluation—under parameter, memory, or latency constraints. Studies confined to multimodal RAG, code RAG, or LLM-scale RAG without regard for SLM-specific constraints were acknowledged but not examined in detail. To ensure accuracy, this survey draws all quantitative results directly from the source papers rather than from secondary summaries.

Paper Organization. The remainder of this survey is structured as follows. Section 2 establishes the technical foundations, encompassing SLM characteristics, RAG architecture, retrieval components, and evaluation frameworks. Section 3 examines the technical challenges that distinguish SLM-RAG from LLM-RAG. Section 4 reviews emerging solutions across the five taxonomy categories. Section 5 discusses application domains. Section 6 outlines open challenges for future research. Section 7 concludes.

II. Background and Preliminaries

In this section, the authors discuss small language models (§2.1), the classical RAG architecture (§2.2), retrieval components (§2.3), and the evaluation frameworks (§2.4).

2.1 Small Language Models

Small language models (SLMs) are compact transformer models comprising roughly 0.5 to 7 billion parameters. The upper bound of this range is not universally fixed: Nguyen et al.[3] adopt 7B, whereas others extend it to 13B. This survey adopts 7B as the primary threshold, treats the 7B–13B range as a boundary zone, and classifies models of 13B parameters and above as large language models (LLMs).

Model Families. Several SLM families are in common use today. Microsoft's Phi-3 series includes the 3.8B-parameter phi-3-mini alongside larger variants of up to 14B parameters; its technical report emphasizes data quality, demonstrating that a model trained on a carefully filtered and synthetically augmented corpus can rival substantially larger models in

performance. Google's Gemma family, available in 2B, 9B, and 27B configurations with an 8,192-token context window, is widely used for on-device deployment. Alibaba's Qwen2.5 series spans 0.5B to 72B parameters and offers a broad range of context lengths—32K tokens for sub-7B models and 128K tokens for models of 7B and above. TinyLlama[13], a 1.1B-parameter model, provides only 2,048 tokens of context, severely constraining the amount of retrieved content that can be included in the prompt. Finally, Apple Intelligence[14] runs approximately 3B-parameter models directly on consumer devices through aggressive quantization combined with task-specific LoRA adapters that are swapped at runtime.

These models are summarized in Table 1. Across this set, context length varies by nearly two orders of magnitude, a difference that bears directly on RAG. A retrieved document that fits comfortably within Qwen2.5-7B's 128K-token window can exceed TinyLlama's entire context budget, necessitating filtering or summarization before the document can be supplied to the generator as evidence.

Table 1: Representative SLMs — model, developer, parameters, context length, release year, training emphasis

Model	Developer	Parameters	Context Length	Release Date	Training Emphasis
TinyLlama	StatNLP Research	1.1B	2,048	Sep 2023	Open pretraining on 3T tokens; Llama-2 architecture
Phi-3-mini	Microsoft	3.8B	4K / 128K	Apr 2024	Curated and synthetic data quality
Phi-3-small	Microsoft	7B	8K / 128K	Apr 2024	Curated and synthetic data quality
Phi-3-medium	Microsoft	14B	4K / 128K	Apr 2024	Curated and synthetic data quality
Gemma 2 (2B)	Google	2B	8,192	Jun 2024	Knowledge distillation from larger teacher
Gemma 2 (9B)	Google	9B	8,192	Jun 2024	Knowledge distillation from larger teacher
Apple Intelligence (on-device)	Apple	~3B	32K	Jun 2024	On-device with per-task LoRA adapters
Qwen2.5 (0.5B)	Alibaba	0.5B	32K	Sep 2024	Multilingual, 18T tokens
Qwen2.5 (1.5B)	Alibaba	1.5B	32K	Sep 2024	Multilingual, 18T tokens
Qwen2.5 (3B)	Alibaba	3B	32K	Sep 2024	Multilingual, 18T tokens
Qwen2.5 (7B)	Alibaba	7B	128K	Sep 2024	Multilingual, 18T tokens
Llama-3.2 (1B)	Meta	1.2B	128K	Sep 2024	Distilled from Llama-3.1
Llama-3.2 (3B)	Meta	3.2B	128K	Sep 2024	Distilled from Llama-3.1
Phi-4-mini	Microsoft	3.8B	128K	Feb 2025	Synthetic reasoning data; GQA, 200K vocab
Gemma 3 (1B)	Google	1B	32K	Mar 2025	Distillation; text-only
Gemma 3 (4B)	Google	4B	128K	Mar 2025	Distillation; multimodal; 140+ languages
Qwen3 (0.6B)	Alibaba	0.6B	32K	Apr 2025	Hybrid thinking/non-thinking modes; 36T tokens
Qwen3 (1.7B)	Alibaba	1.7B	32K	Apr 2025	Hybrid thinking/non-thinking modes
Qwen3 (4B)	Alibaba	4B	32K	Apr 2025	Hybrid thinking/non-thinking modes

The models in Table 1 represent the major SLM families rather than an exhaustive catalog; newer entries such as SmolLM and the DeepSeek distilled models follow the same patterns

Deployment Motivation. SLMs occupy a deployment regime that LLMs cannot reach. On-device inference eliminates cloud round-trips and reduces latency to a few milliseconds. Private enterprise pipelines can process queries locally without transmitting sensitive data to third-party APIs. Many edge applications—including IoT devices, mobile assistants, and offline tools—must operate within strict memory and compute budgets, and here the roughly 3–4 GB footprint of a quantized 3B model is feasible where a 70B model is not.

Three limitations recur across SLMs and bear directly on RAG. First, parametric capacity is bounded: as the Phi-3 authors state plainly, phi-3-mini cannot store much factual knowledge[6]—a consequence of scale rather than a training deficiency. Second, reasoning depth diminishes at smaller scales, particularly for multi-step tasks involving decomposition, intermediate computation, and self-correction—capabilities that appear far more reliably at 7B and above than at 1–3B[5]. Third, context handling is not uniform across the advertised window: Liu et al.[10] document the "lost in the middle" phenomenon, in which accuracy on retrieval-intensive tasks is markedly higher when relevant information appears at the beginning or end of the context than in the middle—an effect that is measurable in LLMs and becomes more severe at smaller scales.

2.2 Retrieval-Augmented Generation

RAG systems pair a retriever, which returns a ranked set of documents from an external corpus for a query, with a generator that produces output conditioned on both the query and those documents. Figure 1 shows the full pipeline: an

offline indexing stage that stores documents as searchable vectors, and an online query stage that retrieves from that store and generates an answer.

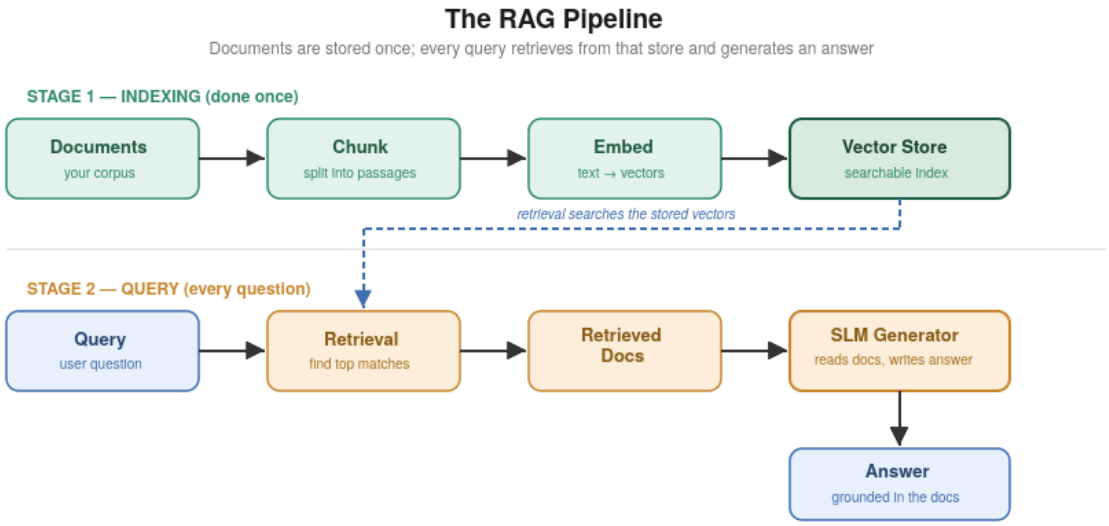

Figure 1: The RAG pipeline. Documents are indexed once into a vector store; each query then retrieves from that store and the generator produces a grounded answer.

Foundational Formulation. Lewis et al.[1] introduced RAG in 2020 using Dense Passage Retrieval as the retriever and BART as the generator. They defined two variants that still structure much of the design space. RAG-Sequence conditions an entire output sequence on a single retrieved document, which is simpler and more efficient but limits the model to one source per answer. RAG-Token allows different tokens in the output to draw on different documents, giving more flexibility at higher computational cost. Both variants use maximum inner product search to identify the top-k documents, typically with k between 5 and 20.

Dense Retrieval Pipeline. The retriever typically operates on dense embeddings. A query encoder maps the query to a fixed-dimensional vector. Document encoders have already mapped corpus documents to vectors and indexed them for approximate nearest-neighbor search using tools such as FAISS. Relevance is measured by cosine similarity or dot product between query and document vectors. The top-k documents are then concatenated with the query and passed to the generator, which produces output autoregressively.

Design Choices for the SLM Setting. Two choices matter more when the generator is a small model. First, whether to train the retriever jointly with the generator or use a pre-built one. Lewis et al. trained jointly but found that updating the document encoder did not help; most later systems skip retriever training entirely, which is easier to set up but weaker on specialized topics. Second, how to feed retrieved documents into the prompt. Options range from concatenating the top-k results unchanged, to reordering them, to filtering out weak ones before the generator sees them. These choices matter more for SLMs because their context budgets are smaller and their tolerance for noise is lower. Sections 3 and 4 cover both.

2.3 Retrieval Components

The quality of retrieval imposes a ceiling on answer quality. This holds for any RAG system but is especially pronounced when the generator is small: a large model can temporarily mask poor retrieval by drawing on its parametric knowledge, whereas a small model cannot compensate in this way and is therefore far more dependent on the retriever.

Dense Encoders. In 2019, SBERT[17] introduced the use of two BERT encoders, trained jointly on query and document texts, to map each text into a vector that permits efficient comparison at retrieval time. This paradigm has been adopted by more recent models such as Contriever, E5, and BGE, yet SBERT remains the reference standard.

ColBERTv2[18] employs a distinct approach known as late interaction. Rather than storing a single vector per document, it retains one vector per token and, during scoring, compares each query token to its nearest counterpart in the document. This preserves fine-grained detail that single-vector methods discard, at the cost of an index roughly ten times larger. The paper mitigates—though does not fully eliminate—this overhead through compression.

Retrieval Benchmarks. Two dominant paradigms exist for evaluating retrievers. BEIR[19] comprises 18 zero-shot tasks spanning fact checking, biomedical search, open-domain QA, and citation prediction; notably, it establishes that dense retrievers remain competitive with—and at times surpass—classical BM25 keyword matching in the zero-shot setting. MTEB[20] is broader, covering 58 datasets across 8 task types, and its leaderboard has become the standard reference for evaluating dense embedding models.

End-to-End Task Benchmarks. KILT[21] serves a different purpose. Rather than evaluating retrievers in isolation, it unifies 11 knowledge-intensive NLP tasks—including fact checking, open-domain QA, slot filling, entity linking, and dialogue—over a single 2019 Wikipedia snapshot, enabling end-to-end evaluation of a full RAG system against one shared knowledge source. This addresses a genuine shortcoming of component-level benchmarking, in which a system may score well on BEIR yet perform poorly on the downstream task of interest. KILT is an in-knowledge-base benchmark: each test question has a corresponding answer within Wikipedia. A consequence, however, is that questions unanswerable from the snapshot are excluded. While this simplifies evaluation, it introduces a practical limitation, since knowing when to abstain—to respond "I don't know"—is often as important as answering correctly, and KILT does not assess this capability.

2.4 Evaluation Frameworks

End-to-end RAG evaluation is inherently more difficult than component-level evaluation. A strong retriever may still yield incorrect answers if the generator disregards the supplied evidence, while a weak retriever may still produce correct ones if the generator's parametric knowledge compensates for the gap. Because component scores do not combine cleanly, RAG-specific evaluation frameworks are required.

Quality Evaluation. RAGAS[22] assesses RAG systems along four dimensions. *Faithfulness* measures whether the generated output is grounded in the retrieved context—that is, whether its claims can be verified against the provided documents. *Answer relevance* measures whether the output genuinely addresses the query. *Context precision* measures whether the retrieved documents contain information pertinent to the answer. *Context recall* measures whether all information necessary to produce the answer was successfully retrieved. RAGAS employs GPT-3.5 or a stronger model as the judge for these assessments, thereby importing LLM-level reasoning into the evaluation loop. This design choice is problematic when the system under evaluation is an SLM operating without cloud LLM access—precisely the deployment context that motivates SLM-RAG. Section 6 revisits this limitation in detail.

Hallucination Evaluation. Hallucination benchmarks target a specific failure mode: the generation of content unsupported by the available evidence. HaluEval[23] comprises 35,000 annotated examples spanning QA, dialogue, and summarization tasks. Its principal empirical finding is that ChatGPT hallucinates on roughly 19.5% of these tasks, even without adversarial conditioning. RAGTruth[24] focuses exclusively on grounded generation, providing 17,790 responses labeled for hallucination at the span level. The benchmark evaluates the behavior of both LLMs and SLMs with respect to an explicitly presented set of retrieved evidence. Its key finding, for the purposes of this survey, is that GPT-4 produces 5–10 times fewer hallucinations when evidence is re-supplied, whereas Llama-2-7B and Mistral-7B do not exhibit a comparable improvement. This indicates that the SLM grounding problem is a distinct empirical phenomenon rather than merely a scaled-down version of the LLM grounding problem.

Noise Robustness. RGB systematically varies the proportion of noise—irrelevant or false documents—in the retrieved set, from 30% up to 90%. As this proportion increases, generator accuracy degrades correspondingly. The quantitative results from RGB are drawn upon extensively in Section 3.3 to characterize the noise sensitivity of SLMs in detail.

III. Technical Barriers

Six barriers recur when RAG pipelines designed for LLM-scale generators are applied to SLMs. Each is documented below with direct experimental evidence.

3.1 Context Window Pressure

Context window capacity is the most immediate barrier to SLM-RAG. A standard RAG pipeline retrieves five to ten documents and concatenates them with the query. At typical chunk sizes of 256 to 512 tokens per document, prompts reach 1,500 to 6,000 tokens before the generator produces any output. LLMs running at 32K or 128K context handle this comfortably; SLMs often cannot.

Context Budgets Are Tighter Than They Appear. Even models that advertise long context windows do not use them uniformly well. Liu et al.[10] documented a systematic pattern in their "Lost in the Middle" work: accuracy on document question-answering tasks is highest when relevant information appears near the beginning or end of the context, and drops substantially for middle positions. The curve is U-shaped, and a model with a 32K reported context may have an effective context of considerably less if its attention underweights the middle.

A second finding from the same work matters specifically for SLM-RAG. Reader accuracy saturates well before retriever recall. Liu et al. show that increasing the number of retrieved documents from 20 to 50 yields only a 1 to 1.5 percentage point gain in end-task accuracy, despite retriever recall continuing to climb. More retrieval is not automatically more useful. For SLMs with limited context budgets, feeding additional documents into a small reader wastes context space and provides diminishing returns.

The Practical Constraint. TinyLlama's 2,048-token context is the extreme case in Table 1. A 50-token query plus five retrieved passages of 400 tokens each consumes 2,050 tokens — the entire context budget — before the model has generated anything. Even more permissive SLMs like Gemma 2 at 8,192 tokens leave limited room for multi-document retrieval if the query itself is long or if the task requires chain-of-thought reasoning in the output.

SLM-RAG pipelines therefore cannot rely on the "retrieve more and let the model sort it out" strategy that works at LLM scale. Selective retrieval, aggressive compression, and explicit filtering become necessary rather than optional. Section 4.1 examines the approaches that have been proposed.

3.2 Retrieval-Indexing Mismatch

RAG assumes that the generator can process retrieved documents well enough to distill an answer. For LLMs this assumption holds. For SLMs below 3B parameters, it often does not.

MiniRAG's Diagnosis. The clearest empirical evidence comes from the MiniRAG paper[7]. Its authors ran LightRAG[34], GraphRAG[36], and several other graph-based RAG systems against both LLM and SLM readers on the same benchmarks. LightRAG drops from 56.90% accuracy with gpt-4o-mini as reader to 39.81% with Phi-3.5-mini, a roughly 30% relative drop. GraphRAG fails outright when its indexing pipeline runs on any of the SLMs tested, producing blank rows in the comparison table because the SLM-generated knowledge graph is too incomplete or malformed to support downstream retrieval.

The failure is architectural, not incidental. Graph-based RAG pipelines rely on the indexing step for coherent entity extraction, relationship mapping, and semantic descriptions — tasks that work at LLM scale and break down at SLM scale. When the indexing LLM is replaced by an SLM, retrieval quality collapses independent of whether the reader itself is an LLM or an SLM. An ablation in the MiniRAG paper isolates the effect: replacing heterogeneous graph indexing with only semantic indexing approximately halves the end-to-end accuracy. Weak entity extraction and weak semantic understanding compound rather than averaging.

RoseRAG's Pilot Data. RoseRAG[9] provides a complementary measurement at the retrieval end. Using Qwen2.5-1.5B-Instruct as the reader, the authors report that exact match accuracy drops from roughly 0.6 to 0.4 as the number of noisy retrieved documents rises from zero to five. More notably, F1 scores drop from 0.4 to 0.2 as the number of retrieved documents increases from one to ten. More retrieval makes the small reader worse, inverting the standard LLM-RAG assumption that additional retrieval provides additional evidence.

Retrieval systems optimized for recall dump large volumes of candidate evidence into the prompt on the assumption that the reader can filter. SLMs cannot filter well enough. Section 4.3 develops the implication that retrieval-indexing architecture needs to be rebuilt for the SLM setting rather than ported from LLM pipelines.

3.3 Noise Sensitivity and Hallucination

When retrieval produces noisy or partially irrelevant documents, the generator is responsible for ignoring the noise and grounding in the relevant evidence. This is where SLMs show their clearest weakness relative to LLMs.

RGB's Systematic Measurement. The RGB benchmark evaluates noise robustness by explicitly injecting irrelevant documents into the retrieved set at controlled ratios. ChatGLM2-6B drops from 91.33% exact match at zero noise to

57.33% at 80% noise — a 34-point absolute drop. ChatGPT degrades less dramatically over the same range, from 96.33% to 76.00%, and the absolute performance floor remains higher across the board.

Negative rejection — the ability to correctly refuse to answer when no relevant evidence is retrieved — is also weaker at SLM scale. RGB's best negative-rejection score belongs to ChatGPT at 24.67% exact match, already a low number for a supposed baseline; SLMs score substantially lower. When presented with only noise, most SLMs confidently generate plausible but unsupported answers rather than refusing.

RGB's error taxonomy identifies three recurring failure modes. Long-distance information errors occur when the relevant evidence is present in the retrieved set but sits far from the query in the prompt; the model fails to bridge the distance. Evidence uncertainty errors occur when multiple pieces of evidence partially support different answers; the model picks one without adequate justification. Concept confusion errors occur when the retrieved documents mention entities similar to but not identical to the query target; the model conflates them. All three become more frequent at smaller model scales.

RAGTruth's Grounded Hallucination Data. RAGTruth[24] approaches the same problem from the hallucination-detection angle. The benchmark annotates 17,790 LLM and SLM responses at the span level in grounded RAG settings where retrieved evidence has been explicitly provided. Its central finding is that Llama-2-7B and Mistral-7B produce 5 to 10 times the hallucination density of GPT-4 across QA, data-to-text, and summarization tasks, even with grounding.

The qualifier matters. SLM hallucination is not merely more frequent—it occurs at spans where the retrieved evidence does contain the correct answer. The generator fails to anchor in the evidence even when the evidence is present. This distinguishes SLM hallucination from straightforward knowledge gaps and suggests that improving retrieval quality alone will not close the gap. The generator itself requires training or architectural modifications that enforce evidence-anchored generation, which Section 4.2 examines.

A Caveat on Scale. Most RGB and RAGTruth data uses 6-7B models. The qualitative patterns should hold at sub-2B scales (noise sensitivity and hallucination worse, not better), but the magnitudes there are not directly measured. Section 6 returns to this gap.

3.4 Reasoning Depth Limitations

Modern RAG systems frequently expect the generator to reason over retrieved evidence: chain-of-thought across multiple documents, multi-hop question answering, self-verification of generated claims. These capabilities scale with model size and often emerge or stabilize only at 7B and above.

Capacity-Bounded Knowledge. The Phi-3 authors[6] state the constraint for their 3.8B model directly: it cannot store large volumes of factual information. This is precisely the argument for retrieval, but it also limits reasoning. Some tasks presuppose background information that the generator is assumed to hold parametrically, and when retrieval does not supply enough of it, the model produces answers that are locally plausible but globally incoherent.

The Chunking-Reasoning Interaction. There's a similar issue raised by Qu et al. with chunking paradigm work ²6. The documents retrieved are divided at indexing time into fixed or variable size chunks. For multi-hop reasoning, if the chunks in the different documents need to be linked together across the chunks, then the chunk boundaries lead to errors. In some instances, chunks are connected and synthesized into a cohesive text by LLM readers. SLM readers are less good at this bridging and have chunk boundaries as hard walls in their reasoning traces.

The Impossible Triangle. Zhu and Zeng[29] framed the underlying tension in 2022 as the "Impossible Triangle": a pre-trained model cannot maximize small size, few-shot capability, and fine-tuned capability at once. RAG and targeted fine-tuning have softened this (Phi-3 and Gemma 2 approach larger-model performance on specific tasks), but reasoning depth, knowledge capacity, and deployment footprint still cannot all be maximized together.

3.5 Multi-Step and Agentic RAG Failures

A growing number of RAG systems use iterative or agentic patterns: the generator decomposes a query into sub-questions, issues multiple retrieval calls, and synthesizes the results. These patterns improve LLM-RAG performance on complex multi-hop queries and fail consistently on SLMs.

Concrete Method Failures. RoseRAG evaluated several established agentic-RAG methods on sub-2B SLMs and reports specific failure modes. ReAct and SelfAsk, which rely on the generator to decompose queries into coherent sub-questions, produce decomposition chains that are too fragmented or off-topic for the retrieval stage to recover from. BlendFilter and ASTUTE, which rely on the generator to self-filter irrelevant retrieved passages, produce filtering decisions noisier than no filtering at all — the small model cannot reliably distinguish relevant from irrelevant evidence. The RoseRAG authors frame this as a scale-dependent limitation rather than a flaw in the methods themselves.

DRAG as a Partial Counterexample. DRAG[8] offers a useful contrast. Rather than asking an SLM to perform multi-step reasoning at inference time, DRAG distills multi-step capabilities from a GPT-4o-mini teacher into the SLM during training. The resulting Phi-3.5-mini reader improves from 35.26% to 40.13% exact match on the E split of the benchmark used, Qwen2.5-3B improves from 32.59% to 37.50%, and Llama-3.1-8B improves from 45.07% to 51.73%. These are meaningful gains, but they carry a dependency: the SLM is only as good as the teacher LLM's annotations. Section 6 examines this teacher-dependency pattern as a recurring concern.

The broader pattern is that agentic RAG methods implicitly assume a reasoning budget the generator can afford to spend iteratively. SLMs have less of this budget, so each reasoning step is lower quality, and errors compound across iterations faster than in LLM-RAG. Linear or single-step RAG pipelines are often more reliable for SLMs than sophisticated multi-step pipelines, even when the latter demonstrably improve LLM-RAG performance.

3.6 Evaluation and Measurement Gaps

The barriers identified thus far are technical properties of SLM-RAG systems. A further barrier is methodological: current evaluation practices for RAG were developed primarily with LLM readers in mind and do not always transfer cleanly to the SLM setting.

LLM-as-Judge in SLM Evaluation. Several leading evaluation standards rely on LLM judges. RAGAS[22] explicitly requires GPT-3.5 or a stronger model for its faithfulness and answer-relevance assessments, and many recent RAG benchmarks likewise depend on GPT-4 to score system outputs or to generate reference answers. This introduces a methodological inconsistency. The very deployment scenarios that motivate SLM usage—offline, private, edge-based, and cost-sensitive settings—are precisely those in which access to cloud LLMs for evaluation is unavailable or undesirable. As a result, the system is not evaluated within the regime for which it is designed.

Benchmark Coverage. Most knowledge-intensive NLP benchmarks are built around in-knowledge-base queries: each test case is paired with a gold passage in the knowledge source. This is the design that KILT[21] explicitly adopts. An important class of queries is thereby excluded—namely, those for which the knowledge base contains no answer and the system must recognize this and abstain. Standard benchmarks measure such negative rejection poorly, a capability that, as noted in Section 3.3, is particularly weak in SLMs. This real-world failure mode remains largely unaddressed by current evaluation methodology.

Missing Sub-2B Data. Production deployments increasingly target sub-2B models, such as Qwen3-0.6B and Qwen3-1.7B, yet most benchmarks are based on 6–7B models. Consequently, part of the argument developed in Sections 3.2–3.4 rests on extrapolation that is qualitatively motivated but not quantitatively substantiated at this smaller scale.

Taken together, these observations indicate that SLM-RAG failures stem not from a single bottleneck but from interacting constraints: context capacity, retrieval noise, indexing quality, reasoning depth, and evaluation methodology compound one another. A remedy directed at any one of them rarely resolves the rest, which is why the solutions discussed in the next section intervene at different points in the pipeline rather than at a single stage. Table 2 maps the failure modes identified in this section to the papers that document them, indicating which benchmarks provide direct evidence and which rely on extrapolation.

Table 2: Failure modes of LLM-era RAG on SLMs — mapping barriers to evidence

Barrier	Section	Primary Evidence	Models Tested	Quantitative Finding	Evidence Scope
Context window pressure	3.1	Lost in the Middle[10]	GPT-3.5, Claude-1.3, LongChat-13B	Accuracy highest at prompt edges; 20→50 docs yields only 1–1.5% gain	Partial extrapolation (sub-3B)
Context budget exhaustion	3.1	Table 1 (derived)	TinyLlama, Gemma 2, Phi-3	TinyLlama 2,048-token context; a query plus 5 × 400-token passages fills entire budget	Direct

Retrieval-indexing mismatch (indexing)	3.2	MiniRAG[7]	gpt-4o-mini, Phi-3.5-mini, MiniCPM, Llama-3.2-3B	LightRAG accuracy: 56.90% → 39.81% when SLM replaces LLM reader; GraphRAG fails outright on all SLMs tested	Direct
Retrieval-indexing mismatch (reader)	3.2	RoseRAG[9] pilot	Qwen2.5-1.5B-Instruct	EM drops from ~0.6 to ~0.4 with five noisy documents; F1 drops from 0.4 to 0.2 as retrieval grows from 1 to 10 docs	Direct
Noise sensitivity	3.3	RGB[25]	ChatGLM2-6B, ChatGPT, LLaMA-2 variants	ChatGLM2-6B: 91.33% → 57.33% EM as noise rises from 0% to 80%	Partial extrapolation (sub-2B)
Negative rejection weakness	3.3	RGB[25]	ChatGPT, ChatGLM2-6B, Vicuna-7B	Best negative-rejection EM is 24.67% (ChatGPT); SLMs score substantially lower	Partial extrapolation (sub-2B)
Error taxonomy: long-distance info, evidence uncertainty, concept confusion	3.3	RGB[25]	6–7B SLM class	Three recurring error types identified; frequency increases at smaller scales	Conceptual (sub-2B not measured)
Grounded hallucination	3.3	RAGTruth[24]	Llama-2-7B, Mistral-7B, GPT-4	SLM hallucination density 5–10× that of GPT-4 across QA, D2T, summarization even with grounding	Partial extrapolation (sub-2B)
Reasoning depth limitations (parametric knowledge)	3.4	Phi-3[6] (authors' own statement)	phi-3-mini (3.8B)	Authors state phi-3-mini "does not have the capacity to store too much factual knowledge"	Direct
Reasoning-chunking interaction	3.4	Chunking Paradigm[26]	Mixed	Multi-hop reasoning breaks at chunk boundaries; effect stronger on smaller readers	Conceptual
Size/capability trade-off	3.4	Impossible Triangle[29]	Conceptual (pre-LLM era analysis)	Moderate model size, SoTA few-shot, and SoTA fine-tuning cannot be simultaneously achieved	Conceptual (2022 framing)
Agentic method failure (decomposition)	3.5	RoseRAG[9]	Qwen2.5-1.5B, gemma-2-2b-it, Llama-3.2-1B	ReAct and SelfAsk produce fragmented decompositions unusable for retrieval	Direct
Agentic method failure (self-filtering)	3.5	RoseRAG[9]	Qwen2.5-1.5B, gemma-2-2b-it, Llama-3.2-1B	BlendFilter and ASTUTE produce filtering noisier than no filtering	Direct
Distillation as partial remedy	3.5	DRAG[8]	Phi-3.5-mini, Qwen2.5-3B, Llama-3.1-8B	Teacher-distilled SLMs gain 4–7 EM points; but method requires GPT-4o-mini teacher	Direct
LLM-as-judge evaluation dependency	3.6	RAGAS[22]	GPT-3.5 / GPT-4 judges	Standard SLM-RAG evaluation requires cloud LLM access — same deployment context SLMs avoid	No — design feature
In-KB benchmark assumption	3.6	KILT[21]	All evaluated systems	All evaluation instances assume gold passage in knowledge source; unanswerable queries excluded	No — design feature
Sub-2B measurement gap	3.6	Derived from Sections 3.2–3.4	Phi-3.5-mini (3.8B), Qwen3-0.6B, Qwen3-1.7B	Production deployments target sub-2B; most benchmarks measure 6–7B SLMs	Inherent gap

IV. EMERGING SOLUTIONS

Solutions to the barriers identified in §3 cluster by where they intervene in the RAG pipeline: retrieval-side filtering (§4.1), generator-side training (§4.2), architectural redesigns (§4.3), efficiency techniques (§4.4), and SLM–LLM collaboration (§4.5). Figure 2 maps these five categories onto the pipeline, and Table 3 summarizes the taxonomy with Table 4 detailing training-time teacher-LLM dependencies.

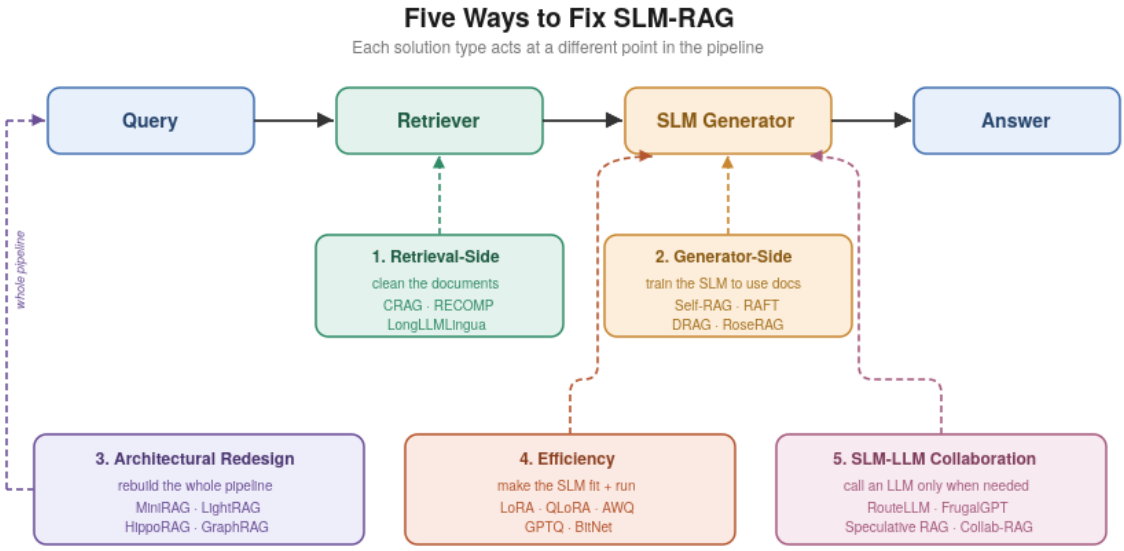

Figure 2: Five ways to fix SLM-RAG. Each solution category acts at a different point in the pipeline.

4.1 Retrieval-Side Filtering and Correction

Retrieval-side methods improve what the generator sees. They fix poor retrieval output before it reaches the small model, which matters because SLMs cannot recover from noisy input as well as LLMs can.

CRAG. Corrective RAG[16] adds a lightweight retrieval evaluator to the pipeline. The evaluator is a T5-large model with 770M parameters, much smaller than the generator. For each query, it scores the retrieved documents and assigns one of three labels: Correct, Incorrect, or Ambiguous. When the label is Correct, the documents go through a refinement step that breaks them into smaller passages and keeps the most relevant ones. When the label is Incorrect, the retrieved documents are discarded and the system falls back to web search. When the label is Ambiguous, both approaches are used together.

CRAG is designed to plug into any existing RAG pipeline. The T5-large evaluator can be run locally, so it fits SLM deployment constraints. The web-search fallback handles cases where the local corpus does not contain the answer — a failure mode that SLMs handle particularly badly.

REPLUG. REPLUG[32] trains the retriever to produce documents that help the generator. It treats the generator as a black box and uses the generator's language modeling score as the training signal for the retriever. The retriever learns to find documents that reduce the generator's perplexity on the target output. REPLUG was designed for very large LMs (100B+), but the technique applies to SLMs too: if the retriever can be trained with SLM supervision, the pipeline does not need LLM access at inference time.

RECOMP. RECOMP[28] compresses retrieved documents before passing them to the generator. It uses two compressors. The extractive compressor is a dual-encoder that selects the most relevant sentences from retrieved documents by ranking them against the query. The abstractive compressor is a small encoder-decoder distilled from GPT-3 that generates query-focused summaries. Both compressors are much smaller than the target generator. On question-answering tasks, RECOMP compresses retrieved documents to 5–10% of their original length with less than a 10% relative drop in performance. For SLMs with tight context budgets, this compression is valuable.

LongLLMLingua. LongLLMLingua[27] takes prompt compression further. It uses a small language model (GPT-2-small in its experiments) to compute token-level perplexity and removes tokens that contribute little information. A key

technique is question-aware compression: the system scores document importance by computing the perplexity of the question conditioned on each document, not the perplexity of the document itself. This produces more targeted compression. On LongBench, LongLLMLingua compressed prompts from ~10,000 tokens to ~2,000 tokens while actually improving task performance (48.3 vs 44.0 average score) — the compression removes noise that was hurting the generator.

Adaptive-RAG. The methods above act on documents after retrieval. Adaptive-RAG[53] acts earlier, on the decision of whether to retrieve at all. Not every question needs retrieval, and forcing it on simple questions wastes the SLM's limited context budget on documents it does not need. Adaptive-RAG trains a small classifier to read an incoming question and predict its complexity, then routes the question to one of three paths: answer directly with no retrieval, retrieve once and answer, or retrieve iteratively for genuinely multi-step questions. Because the classifier is itself a small model, the routing adds little overhead, and matching effort to question difficulty improves both accuracy and efficiency over always retrieving or never retrieving.

Summary. Retrieval-side methods are attractive for SLM deployment because they keep the filter or compressor small and independent of the generator. The generator continues to work on shorter, cleaner input than it would see from a naive retrieval pipeline. These methods also do not require training the SLM itself, which avoids the computational cost and teacher-LLM dependencies that appear in the next section.

4.2 Generator-Side Training

Retrieval-side methods improve the input to the generator. Generator-side methods improve the generator itself. These approaches train the SLM to handle retrieved evidence better through supervised fine-tuning, reinforcement learning, or distillation from a larger teacher.

Self-RAG. Self-RAG[15] trains the generator to emit reflection tokens that decide whether to retrieve, judge whether retrieved documents are relevant, and check whether the output is supported by them. The model retrieves adaptively, only when it lacks the knowledge to answer directly. Self-RAG-7B outperforms ChatGPT-with-retrieval on four of six benchmarks (PopQA, PubHealth, biography generation, ASQA).

Its annotations are produced by GPT-4, so training a new Self-RAG model requires GPT-4 access, a recurring pattern in generator-side methods.

RAFT. RAFT[30] (Retrieval-Augmented Fine-Tuning) trains the generator on a mix of clean and distractor-containing retrieval examples. For each training question, 80% of the examples include the gold document and several distractor documents; 20% include only distractor documents and force the model to rely on parametric knowledge. The training answers include chain-of-thought reasoning, so the model learns to cite specific retrieved passages when generating answers. RAFT achieves gains of up to 35% on HotpotQA and 76% on Torch Hub benchmarks over the base Llama-2-7B-chat. The CoT reasoning adds a further 9–15% improvement. Like Self-RAG, RAFT requires a GPT-4-class teacher to generate the chain-of-thought annotations.

InstructRAG. InstructRAG[31] removes the GPT-4 teacher dependency: the generator produces its own rationales from (question, retrieved documents, ground-truth answer) triples and learns from them. On the same Llama-3-Instruct-8B backbone it reaches 66.2% on PopQA versus 55.8% for Self-RAG, showing teacher-free training can beat teacher-supervised training.

DRAG. DRAG[8] uses a GPT-4o-mini teacher but is designed specifically for small-model deployment. The teacher generates augmented training data including reasoning traces, and the small model is trained on this data. Phi-3.5-mini improves from 35.26% to 40.13% exact match on the benchmark's E split; Qwen2.5-3B improves from 32.59% to 37.50%; Llama-3.1-8B improves from 45.07% to 51.73%. DRAG also includes a PII reduction step that removes personally identifying information from training data, reducing leakage by 95.7%.

RoseRAG. RoseRAG[9] focuses on sub-2B models (Qwen2.5-1.5B, Llama-3.2-1B, gemma-2-2b-it). It uses rejection sampling plus margin-aware preference optimization: the model generates multiple candidate answers for each training query, a preference selection strategy identifies the best and worst candidates, and the model is optimized with DPO on the preference pairs. RoseRAG outperforms established baselines on HotpotQA, 2WikiMultiHopQA, and StrategyQA. The paper also explicitly documents why established methods (ReAct, SelfAsk, BlendFilter, ASTUTE) fail on sub-2B SLMs, providing the evidence cited in Section 3.5.

Summary. Generator-side training is effective but carries a recurring cost: most methods require a larger teacher LLM to generate training data. Self-RAG, RAFT, and DRAG all need GPT-4 or GPT-4o-mini access. InstructRAG and RoseRAG demonstrate that teacher-free training is possible, which is an important direction for pure SLM pipelines. Section 6 returns to this teacher-dependency pattern.

4.3 Architectural Redesigns for SLMs

Retrieval-side filtering and generator-side training work around the existing RAG architecture. Architectural redesigns change the architecture itself. The five methods in Table 3 rethink how the corpus is indexed or how retrieval is performed, and the central question for each is whether every step in the pipeline is feasible at SLM scale.

MiniRAG. MiniRAG[7] is the only graph-RAG system in this group designed from the start for SLM indexing. It builds a heterogeneous graph containing both entity nodes and text-chunk nodes, then performs topology-enhanced retrieval that traverses the graph rather than asking the reader to reason across chunks at inference. The MiniRAG paper provides the central evidence used throughout this survey: LightRAG accuracy drops from 56.90% with gpt-4o-mini to 39.81% with Phi-3.5-mini, while GraphRAG produces blank rows on every SLM tested because its indexing step fails. MiniRAG itself maintains accuracy across the same SLM substitution.

LightRAG. LightRAG[34] uses graph-based text indexing with a dual-level retrieval framework: low-level retrieval targets specific entities and their relationships, high-level retrieval covers broader topics and themes. An incremental update algorithm avoids rebuilding the full index when new documents arrive. The graph indexing step relies on an LLM for entity and relationship extraction, which is the dependency that MiniRAG's ablations show breaking at SLM scale.

HippoRAG. HippoRAG[35] takes a different graph approach motivated by the hippocampal indexing theory of human memory. Open information extraction builds a knowledge graph over the corpus, and Personalized PageRank is run on this graph at query time to score document relevance. Single-step retrieval with HippoRAG matches or beats iterative methods like IRCoT while being 10–20× cheaper and 6–13× faster, and accuracy improves by up to 20 points on MuSiQue and 2WikiMultiHopQA. The OpenIE step has historically required an LLM, but the paper reports Llama-3.1-8B working competitively in this role, which brings HippoRAG within reach of an SLM-feasible pipeline.

GraphRAG. GraphRAG[36] from Microsoft Research builds an entity knowledge graph from the corpus, then uses Leiden community detection to partition it into hierarchical groups of related entities. Pre-generated community summaries support map-reduce query answering: each community produces a partial response, and the partial responses are combined into a final answer. GraphRAG is designed for global queries about an entire corpus ("what are the main themes?") rather than fact-lookup. It is the clearest illustration of the SLM-indexing problem: every step depends on a capable LLM, and the MiniRAG comparison shows the pipeline fails completely when an SLM is substituted at indexing time.

HyperRAG. HyperRAG[37] is a systems-level redesign rather than an algorithmic one. It addresses the inference-time cost of LLM-based rerankers by reusing document-side KV-cache across queries. Once a document's KV representation has been computed, subsequent queries that score the same document reuse the cached state instead of recomputing it. The paper reports 2–3× throughput improvement with decoder-only rerankers, with downstream task performance going up rather than down. HyperRAG uses Gemma-2B as the reranker in its experiments, which fits naturally into an SLM deployment regime.

Summary. These redesigns target retrieval-indexing mismatch (§3.2). LightRAG and GraphRAG inherit the LLM-indexing failure they aim to fix; MiniRAG and HyperRAG instead build pipelines feasible at SLM scale throughout. HippoRAG sits between, with a documented 8B-class indexing dependency but a genuinely lightweight inference path.

Table 3: Taxonomy of emerging solutions for SLM-RAG

Method	Pipeline Stage	Approach	Key Technique	Barriers Addressed (§3)	Training Req.
CRAG[16]	Retrieval-side	Plug-and-play filtering	T5-large (770M) evaluator; web-search fallback	Noise sensitivity (3.3); context budget (3.1)	None (evaluator pre-trained)
REPLUG[32]	Retrieval-side	Retriever training via LM feedback	KL-divergence supervision from generator	Retrieval-indexing mismatch (3.2)	Generator access for supervision

RECOMP[28]	Retrieval-side	Document compression	Extractive + abstractive compressors	Context budget (3.1); noise sensitivity (3.3)	GPT-3 distillation for abstractive variant
LongLLMLingua[27]	Retrieval-side	Token-level compression	Question-aware perplexity scoring via GPT-2-small	Context budget (3.1); noise sensitivity (3.3)	None (uses pre-trained small LM)
Self-RAG[15]	Generator-side	Reflection tokens	Adaptive retrieval + per-segment self-critique	Noise sensitivity (3.3); reasoning depth (3.4)	GPT-4 teacher for reflection tokens
RAFT[30]	Generator-side	Distractor-aware fine-tuning	80% gold + distractors, 20% distractors only; CoT	Noise sensitivity (3.3); reasoning depth (3.4)	GPT-4-1106 for CoT annotations
InstructRAG[31]	Generator-side	Self-synthesized rationales	Generator produces its own training rationales	Noise sensitivity (3.3); grounded hallucination (3.3)	Teacher-free (key property)
DRAG[8]	Generator-side	Multi-step distillation for SLMs	Teacher-generated reasoning traces; PII reduction	Multi-step failures (3.5); reasoning depth (3.4)	GPT-4o-mini teacher
RoseRAG[9]	Generator-side	Preference optimization	Rejection sampling + margin-aware DPO	Noise sensitivity (3.3); sub-2B operation (3.6)	Teacher-free
MiniRAG[7]	Architectural	SLM-native graph RAG	Heterogeneous graph (entities + chunks); topology search	Retrieval-indexing mismatch (3.2); all SLM scales	None (SLM-compatible indexing)
LightRAG[34]	Architectural	Dual-level graph retrieval	Low-level + high-level graph retrieval	Context budget (3.1); reasoning-chunking (3.4)	LLM required for graph indexing
HippoRAG[35]	Architectural	Graph + PageRank	Personalized PageRank over knowledge graph	Multi-step failures (3.5); reasoning depth (3.4)	LLM or Llama-3.1-8B for OpenIE
GraphRAG[36]	Architectural	Hierarchical community summaries	Leiden community detection; multi-level summaries	Context budget (3.1); reasoning depth (3.4)	LLM required; fails on SLMs per MiniRAG
HyperRAG[37]	Architectural	KV-cache reuse for rerankers	Precomputed document-side KV pairs; Gemma-2B reranker	Context budget (3.1); efficiency	None (uses pre-trained reranker)
LoRA[38]	Efficiency	Parameter-efficient fine-tuning	Low-rank weight decomposition	Enables generator-side methods at low cost	N/A (enables other methods)
QLoRA[39]	Efficiency	Parameter-efficient FT + 4-bit quant	NF4 quantization + double quant + paged optimizers	Deployment footprint; enables fine-tuning	N/A (enables other methods)
AWQ[40]	Efficiency	Post-training quantization	Activation-aware per-channel scaling	Deployment footprint	None (post-training, activation-calibrated)
GPTQ[41]	Efficiency	Post-training quantization	OBQ-based layerwise reconstruction	Deployment footprint	None (post-training, calibration data only)
BitNet b1.58[42]	Efficiency	Quantization-aware training	Ternary weights trained from scratch	Deployment footprint at extreme compression	Full pre-training at 1.58 bits
RouteLLM[43]	Collaborative	Binary routing	Win-prediction model trained on preference data	Cost/capability trade-off; sub-2B capability gaps	Preference data; no LLM teacher for router
FrugalGPT[44]	Collaborative	Cascading + caching	Query cheaper models first; escalate if low confidence	Cost/capability trade-off	None for routing logic
Speculative RAG[33]	Collaborative	Drafter-verifier	Small drafter (Mistral-7B) + frozen verifier (Mixtral-8x7B)	Reasoning depth (3.4); context budget (3.1)	Instruction tuning for drafter
Collab-RAG[46]	Collaborative	SLM decomposer + LLM reader	Iterative preference optimization using LLM feedback	Multi-step failures (3.5); reasoning depth (3.4)	Teacher-free via preference optimization

Table 4: Training-time teacher-LLM dependencies

Method	**Teacher Used at Training Time**	**What the Teacher Provides**	**Teacher-Free Feasible?**
Self-RAG[15]	GPT-4	Reflection token annotations	No
RAFT[30]	GPT-4-1106	Chain-of-thought reasoning traces	No
DRAG[8]	GPT-4o-mini	Multi-step reasoning traces; training data augmentation	No
InstructRAG[31]	None	Self-synthesized rationales from (Q, docs, A)	Yes — by design
RoseRAG[9]	None	Self-generated candidates; preference selection via rejection sampling	Yes — by design
Collab-RAG[46]	GPT-4o-mini (as environment)	Answer quality feedback for iterative preference optimization	Partial — LLM needed at inference but no distillation

CRAG[16]	None	Pre-trained T5-large evaluator	Yes
REPLUG[32]	Generator itself (any scale)	LM likelihood as retrieval training signal	Partial — depends on generator scale chosen
RECOMP[28]	GPT-3 for abstractive variant only	Query-focused summaries for distillation	Partial — extractive variant is teacher-free
LongLLMLingua[27]	None	Uses pre-trained GPT-2-small for perplexity scoring	Yes
MiniRAG[7]	None	Designed for SLM indexing from the start	Yes
LightRAG[34]	LLM for graph indexing	Entity and relationship extraction during indexing	No (per MiniRAG ablations)
HippoRAG[35]	LLM or Llama-3.1-8B	OpenIE for knowledge graph construction	Partial — Llama-3.1-8B competitive with GPT-3.5
GraphRAG[36]	LLM for graph indexing	Entity/relationship extraction; community summaries	No — fails on SLMs per MiniRAG
HyperRAG[37]	None	Uses pre-trained Gemma-2B reranker	Yes
Speculative RAG[33]	None for drafter instruction tuning	—	Partial — requires instruction tuning data

4.4 Efficiency Techniques

Efficiency techniques do not address RAG barriers directly. Rather, they enable SLMs to satisfy the deployment constraints that make RAG-equipped SLMs viable: limited GPU memory for fine-tuning, limited RAM for inference, and constrained energy budgets on edge devices.

LoRA. LoRA[38] freezes the pre-trained weights and injects trainable rank-decomposition matrices into each Transformer layer. For GPT-3 175B, this reduces the number of trainable parameters by a factor of 10,000 and GPU memory requirements by a factor of three, incurring no additional inference latency once the LoRA weights are merged into the base model. For SLM-RAG, LoRA serves as the enabling primitive: most generator-side training methods discussed in §4.2 rely on LoRA rather than full fine-tuning, since the latter is prohibitive on consumer GPUs. Apple Intelligence's per-task adapter design[14] represents the production-scale realization of this idea.

QLoRA. QLoRA[39] combines LoRA with aggressive base-model quantization. The base model is quantized to 4-bit NormalFloat (NF4), a data type designed to match the typical distribution of neural-network weights. Double quantization further compresses the quantization constants themselves, while paged optimizers manage memory spikes during training. QLoRA reportedly fine-tunes a 65B model on a single 48GB GPU while matching the performance of full 16-bit fine-tuning, thereby making SLM fine-tuning feasible on consumer hardware—a 7B model fits comfortably on a 16GB GPU under QLoRA.

AWQ. AWQ[40] is a post-training quantization method that protects the salient weight channels identified by activation magnitude rather than by weight magnitude. This protection is implemented through equivalent transformations that scale up the salient channels prior to quantization, thereby avoiding mixed-precision storage. Paired with the TinyChat inference runtime, AWQ delivers more than a 3× speedup over FP16 on both desktop and mobile GPUs. Notably, the paper demonstrates a 70B Llama-2 model running on a Jetson Orin Nano (8GB, 15W)—hardware incapable of accommodating any 70B-scale model without aggressive compression.

GPTQ. GPTQ[41] is the standard one-shot post-training quantization method, based on layerwise reconstruction with approximate second-order information. It quantizes GPT-style models of 175B parameters in approximately four GPU hours, reducing precision to three or four bits per weight with negligible accuracy degradation. The resulting end-to-end inference speedup over FP16 is 3.25× on the A100 and 4.5× on the more cost-effective A6000.

BitNet b1.58. BitNet b1.58[42] follows a different approach: rather than quantizing post hoc, it constrains every weight to the ternary set {−1, 0, 1} during training from scratch. At 3B parameters and above, BitNet b1.58 matches the full-precision FP16/BF16 Transformer baseline in both perplexity and end-task performance while being substantially smaller, faster, and more energy-efficient at inference time. The method is not yet a drop-in replacement for post-training quantization of existing SLMs, as it requires retraining from scratch; nevertheless, it establishes that ternary quantization-aware training can match higher-precision baselines—a result with significant implications for the next generation of SLMs.

Summary. These techniques operate upstream of RAG. LoRA and QLoRA underpin nearly every generator-side training method in §4.2, while AWQ, GPTQ, and BitNet shape the inference-time footprint that ultimately determines whether a pipeline can be deployed at all.

4.5 SLM–LLM Collaboration

SLM–LLM collaboration treats the SLM and LLM as a pipeline rather than as alternatives. The SLM handles what it can do reliably; the LLM is invoked only when needed. This division of labor has grown into its own research area, surveyed by Fu et al.[45] The four systems below differ on what "needed" means and how the decision is made.

RouteLLM. RouteLLM[43] trains a binary router that chooses between a weak (cheap) model and a strong (expensive) model on a per-query basis. The router is trained on preference data, where queries with a preferred response from one model over the other supply the supervision. The reported result is over 2× cost reduction without substantially compromising quality; one configuration reaches an MT Bench score that is 95% of GPT-4's while making roughly half as many calls to the strong model as random routing. The router itself does not require an LLM teacher.

FrugalGPT. FrugalGPT[44] extends the routing idea into a cascade. Queries are submitted to progressively more expensive models, each output is scored by a learned quality scorer, and the cascade stops at the first model whose output crosses a quality threshold. Caching of previous query–response pairs and prompt-adaptation strategies further reduce cost. The paper reports that FrugalGPT can match GPT-4 performance at 98% lower cost on several benchmarks. As with RouteLLM, the routing and scoring components are themselves small models.

Speculative RAG. Speculative RAG[33] adapts the speculative decoding pattern to RAG. A small drafter (Mistral-7B in the paper's experiments) generates multiple candidate answers in parallel from different subsets of retrieved documents. A frozen verifier (Mixtral-8x7B) scores the drafts and selects the best one. The division of labor exploits SLM efficiency for the bulk of generation and reserves the larger model for the verification step. The pattern aligns naturally with multi-document RAG, since each candidate can come from a different evidence subset and the burden on any single context window is reduced.

Collab-RAG. Collab-RAG[46] pushes this division further. An SLM acts as a query decomposer, breaking a complex query into sub-questions that a frozen LLM reader can answer one at a time. The SLM is trained via iterative preference optimization where the LLM's answer quality feedback supplies the preference signal. No LLM-generated training data is required (no chain-of-thought distillation, no rationale generation), only the LLM's downstream answer-quality signal, which makes the method teacher-free in the §4.2 sense even though an LLM is part of the inference-time pipeline.

Summary. Collaboration methods route hard queries to a larger model, trading some SLM-deployment properties (private, offline, low-latency) for capability headroom. Collab-RAG is the cleanest fit for SLM-RAG: the SLM does structured query decomposition, the LLM does final synthesis over the retrieved evidence.

V. APPLICATIONS

The barriers and solutions in §3 and §4 are abstract until grounded in deployment. This section covers four application domains where SLM-RAG has been deployed, evaluated, or actively studied. The choice of methods, the failure modes, and the open problems differ by domain in instructive ways.

5.1 Healthcare and Biomedical

Healthcare is the canonical SLM-RAG domain: medical knowledge is large, structured, fast-changing, and subject to privacy constraints that rule out cloud LLM use in most clinical environments. Two papers in this survey document the specific shape of SLM-RAG work in this domain.

Privacy-preserving medical SLMs. Zhang et al.[47] address the gap between LLM medical proficiency and SLM deployability under HIPAA-style privacy constraints. Their method extracts keywords from medical input, prompts a cloud LLM to generate medical knowledge-intensive context from the keywords (no patient identifiers leave the device), and uses the generated context as additional input to a local SLM. The SLM is fine-tuned on this augmented data. The reported gain is up to 22.57% absolute accuracy over SLM fine-tuning without context, across three medical knowledge-intensive tasks. The pattern (keyword extraction → LLM context generation → SLM use) is a template for other privacy-restricted domains.

Domain-specialized retrieval. BMRetriever[48] argues the other half of the case: that strong domain-specific retrieval lets smaller models compete with larger general-purpose ones. The model is trained via unsupervised pre-training on large biomedical corpora followed by instruction fine-tuning on labeled biomedical pairs. Across 5 biomedical tasks and 11 datasets, the 410M variant outperforms baselines up to 11.7× larger, and the 2B variant matches models above 5B

parameters. For SLM-RAG, this suggests that retrieval-side investment can substitute for generator capacity, which is the optimistic version of the Phi-3 argument.

5.2 Privacy-Restricted Enterprise

Healthcare is one case of a broader pattern: enterprise deployment under strict privacy or sovereignty constraints, where sending queries or retrieved documents to cloud LLMs is prohibited. Three papers in this survey address differential privacy specifically for RAG.

DP token generation for RAG. Koga et al.[49] propose DPVoteRAG and DPSparseVoteRAG. DPVoteRAG partitions the sensitive corpus into disjoint subsets, sends each to its own LLM voter, and produces output tokens one at a time through majority voting. The composition property of differential privacy makes per-token voting expensive in privacy budget, so DPSparseVoteRAG only spends budget on tokens that genuinely depend on the sensitive corpus, falling back to the non-private model output when voters agree with it. The paper reports useful responses under privacy budgets of $\varepsilon \approx 10$.

Practical DP-RAG. Grislain[50] combines differentially private document selection with DP token generation, and finds it most useful for general knowledge extraction rather than precise per-document lookup, reflecting the privacy-utility trade-off: more noise means more privacy but less fidelity to any single source.

Cloud RAG with privacy. RemoteRAG[51] targets RAG-as-a-Service, where the corpus is in the cloud but the query must not be sent in plaintext. It introduces (n, ε)-DistanceDP with embedding perturbation: the user sends a perturbed query embedding and the cloud returns candidates without seeing the raw query, in two communication rounds.

5.3 On-Device and Mobile

The on-device deployment regime is where SLM-RAG has the clearest commercial traction. Apple Intelligence is the largest production deployment described in the surveyed papers.

Apple Intelligence. The Apple Intelligence technical report[14] describes a tiered architecture: an approximately 3B-parameter foundation model (AFM-on-device) runs locally on devices, and a larger server-based AFM serves Private Cloud Compute when on-device capacity is insufficient. AFM-on-device is initialized from a pruned 6.4B model and trained with distillation, then specialized at inference time via task-specific LoRA adapters that are swapped at runtime. This adapter swapping is the production-scale realization of the LoRA design pattern[38]: one base model, many lightweight task-specific overlays, each occupying a fraction of the storage and memory of a full fine-tune.

Training SLMs for deployment. DRAG[8] is one of the few papers in the surveyed literature designed specifically for the deployable-SLM endpoint. A GPT-4o-mini teacher generates multi-step reasoning traces, and the SLM is trained on these to improve its handling of retrieved evidence. Phi-3.5-mini improves from 35.26% to 40.13% exact match on the benchmark's E split; Qwen2.5-3B from 32.59% to 37.50%; Llama-3.1-8B from 45.07% to 51.73%. DRAG also includes a PII reduction step that removes personally identifying information from training data, reducing leakage by 95.7% — a feature designed for the on-device regime where training data and deployment data share users.

5.4 Education and Specialized Domains

Beyond healthcare, privacy, and consumer-device deployment, SLM-RAG arises in any domain where cloud LLM use is impractical on grounds of cost, latency, or institutional policy. Education is one such domain.

SLM-RAG for Educators. Reza et al.[52] describe an open-source framework that employs locally deployed 3B–7B SLMs for educator-facing content creation and assessment. The system combines RAG with cache-augmented generation, runs entirely on-premises to preserve data privacy, and uses an auxiliary LLM verifier to mitigate jailbreak risk; a college physics pilot validates the framework. The pattern mirrors the medical SLM case: in a domain where cost and privacy preclude cloud LLM use, an SLM-RAG system fills the gap.

Domain-Specific Fine-Tuning. RAFT[30] is the canonical domain-adaptation method for SLM-RAG. The generator is fine-tuned on a mixture of clean and distractor-containing retrieval examples: 80% of examples include the gold document alongside several distractors, while the remaining 20% contain only distractors, so that the model also learns to draw on its parametric knowledge. Training answers incorporate chain-of-thought reasoning. RAFT achieves gains

of up to 35% on HotpotQA and 76% on the Torch Hub benchmark over the base Llama-2-7B-chat model. As with Self-RAG, however, RAFT depends on a GPT-4-class teacher to generate its training data.

SimRAG. RAFT's reliance on a GPT-4 teacher poses a practical difficulty: specialized domains such as medicine and law often lack the labeled data a teacher would require, and may prohibit transmitting domain data to an external model altogether. SimRAG[54] addresses this by enabling the model to teach itself. The model is first equipped with basic question-answering and question-generation abilities, then applied to an unlabeled domain corpus to generate its own training questions. A filtering step retains only the synthetic question–answer pairs that pass a quality check, and the model is fine-tuned on the surviving examples. Across 11 datasets and two backbone models, this self-improving loop adapts the model to the target domain without any external teacher, providing a teacher-free counterpart to RAFT for settings in which a teacher is unavailable or disallowed.

Two patterns recur across the application domains discussed above.

Hybrid Local–Cloud Architectures Dominate. Apple Intelligence pairs an on-device 3B model with a server-side AFM. The medical SLM work of Zhang et al. routes a cloud LLM call for context generation while keeping patient data local. RemoteRAG partitions the RAG service itself across a perturbed-query client and a cloud retriever. Although pure on-device pipelines do exist—such as the education framework described above and MiniRAG-style fully-SLM stacks—the dominant production pattern is a controlled handoff between local and cloud components at carefully chosen boundaries. Section 6 revisits this as an open challenge: determining where the boundary should lie and how to evaluate it.

Privacy–Utility Trade-offs Are the Binding Constraint. Where SLM-RAG is preferred over LLM-RAG, the motivation is typically not accuracy but privacy or data sovereignty. The DP-RAG work illustrates this clearly: utility improves as the privacy budget ε increases, yet the deployment regime imposes an upper bound on ε that in turn caps attainable utility. This bound applies across healthcare, enterprise-data, and on-device personal-information settings alike. The constraint reframes the central research question in SLM-RAG: the objective is not merely to improve small-model performance in absolute terms, but to improve it under a fixed privacy budget. A second, orthogonal constraint—the dependence on large models at training time—is summarized in Figure 3 and examined more fully in §6.

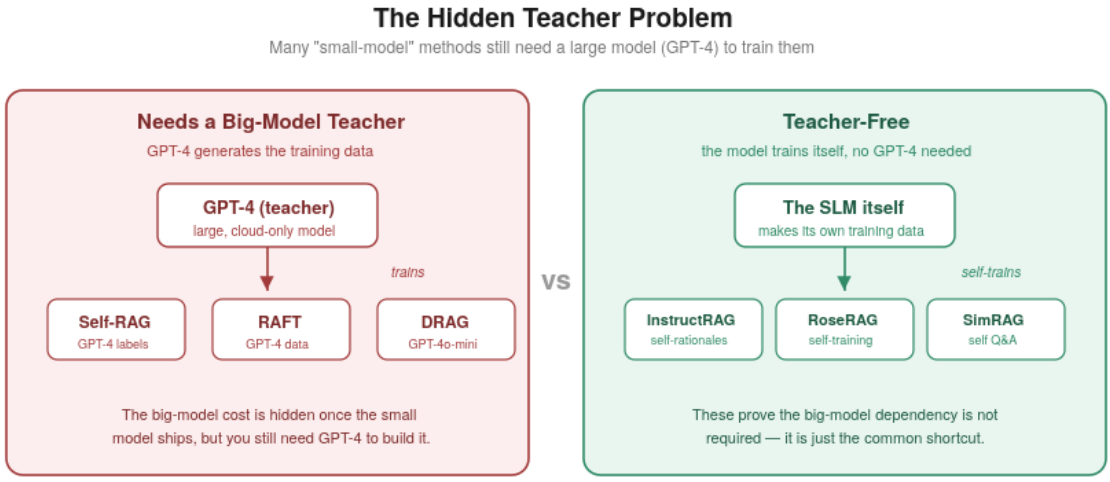

Figure 3: The hidden teacher problem. Several methods marketed for small models still require GPT-4 to generate their training data; a teacher-free group shows the dependency is not required.

VI. Open Challenges

The survey has surfaced six open problems that are not yet resolved by the methods covered in §4. Each is grounded in evidence from a specific paper or set of papers, and each constrains the design space for the next generation of SLM-RAG systems. Table 5 summarizes the challenges and maps them to the evidence sources.

6.1 Teacher-LLM Dependency in SLM-Native Methods

Self-RAG[15], RAFT[30], and DRAG[8] are presented as methods for SLMs, but each requires a GPT-4-class teacher to generate the training data. The deployed artifact is an SLM; the training pipeline is not. This conditions which organizations can build SLM-RAG systems and on what data. InstructRAG[31], RoseRAG[9], and Collab-RAG[46] demonstrate that teacher-free training is feasible, but they cover a narrow range of tasks and SLM sizes. The open question is whether teacher-free training can match teacher-supervised training across the full range of tasks where Self-RAG and RAFT show strong results. If so, the efficiency argument for SLM-RAG becomes substantially stronger. Conversely, if it cannot, the field's dependency on cloud LLM access for training is structural, not incidental.

6.2 Evaluation Under SLM Deployment Constraints

RAGAS[22] uses GPT-3.5 or stronger as a judge for faithfulness and relevance assessment. KILT[21] assumes every test query has a gold passage in the knowledge base. Both designs reflect LLM-RAG assumptions that do not transfer to the SLM deployment regime. An SLM-RAG system shipped to a device, an enterprise, or a clinic typically has no cloud LLM access, so RAGAS-style evaluation cannot run in the deployment environment. And SLM deployments must handle unanswerable queries (queries with no gold passage), which KILT excludes by construction. Recent benchmarks like RGB and RAGTruth improve on this picture but use 6–7B baselines that under-represent the sub-2B production target. Building evaluation methodology aligned with actual SLM deployment is an open challenge that conditions everything else in the field.

6.3 The Sub-2B Measurement Gap

Production SLM-RAG deployments increasingly target sub-2B models: Qwen3-0.6B, Qwen3-1.7B, Gemma 3 (1B), Llama-3.2 (1B). Most experimental evidence in the literature uses 6–7B models as the SLM baseline. RGB measures ChatGLM2-6B, RAGTruth measures Llama-2-7B and Mistral-7B, RAFT measures Llama-2-7B-chat. The qualitative patterns documented at 7B almost certainly carry to 1B (noise sensitivity worse, not better; hallucination worse, not better), but the numerical magnitudes are not directly measured. Closing this gap is a benchmark-construction problem rather than a method-design problem, and it has not received sustained attention.

6.4 Knowing When to Refuse

RGB reports that the best negative rejection score (correctly refusing to answer when no relevant evidence is retrieved) is 24.67% exact match, achieved by ChatGPT. SLMs perform substantially worse. Production deployments cannot ignore unanswerable queries. A medical SLM that confidently fabricates an answer when the patient's record does not contain the necessary information is worse than one that refuses. Negative rejection is treated as a secondary metric in most current benchmarks, when it is treated at all. For SLM-RAG, it is closer to a primary metric, and the methods covered in §4 do not target it directly.

6.5 Privacy Budget as a Binding Constraint

The DP-RAG line of work (Koga et al.[49], Grislain[50], RemoteRAG[51]) shows that differential privacy guarantees can be added to RAG, but the privacy budget ε determines how much utility is retained. In healthcare, enterprise, and on-device deployments, ε is bounded by policy and law before any system design choice is made. The open question is the right shape of the trade-off curve: what design choices in the SLM, in the retriever, in the corpus partition, in the voting protocol, push the curve upward? Most current DP-RAG work documents the curve rather than improves it.

6.6 Architecture, Training, and the Hardware Question

BitNet b1.58[42] matches full-precision baselines at the cost of from-scratch training on ternary weights. The authors argue this calls for hardware specifically designed for 1-bit inference. If such hardware becomes standard, the cost structure of SLM-RAG changes. The footprint of a 7B BitNet model is small enough that retrieval-side context budgets stop being binding, and generator-side training becomes accessible to a wider range of organizations. None of the methods in §4 are designed with this hardware shift in mind. Predicting which subset will survive a transition to ternary-quantized SLMs is open.

Table 5: Open challenges and their evidence sources

Challenge	Description	Primary Evidence	Status

Teacher-LLM dependency	Major generator-side training methods (Self-RAG, RAFT, DRAG) need GPT-4 teachers; SLM-deploy is downstream of LLM-train	Self-RAG[15], RAFT[30], DRAG[8] (need); InstructRAG[31], RoseRAG[9], Collab-RAG[46] (free)	Partial solutions exist, full closure open
Evaluation mismatch	RAGAS requires GPT-4 judge, KILT excludes unanswerable queries — neither matches SLM deployment regime	RAGAS[22], KILT[21]	Open
Sub-2B measurement gap	Production SLMs target sub-2B but benchmarks measure 6–7B	RGB[25], RAGTruth[24] (6–7B); Qwen3-0.6B, Gemma 3 (1B) (production)	Open
Negative rejection	SLMs hallucinate confidently rather than refuse when evidence is absent; best baseline is 24.67% EM	RGB[25]	Open, methods in §4 do not target this directly
Privacy–utility trade-off	DP-RAG utility depends on ε; production ε is bounded by policy	Koga et al.[49], Grislain[50], RemoteRAG[51]	Curve documented; improvement open
Hardware shift implications	BitNet b1.58 calls for 1-bit hardware; consequences for SLM-RAG design space are unstudied	BitNet b1.58[42]	Open

6.7 Future Research Directions

The open challenges above point toward several directions where progress would move the field most.

Teacher-free SLM-RAG. InstructRAG, RoseRAG, and Collab-RAG show that training without a frontier teacher is possible, but only across a narrow range of tasks and model sizes. Extending teacher-free training to match the breadth where Self-RAG and RAFT show strong results is the single change that would close the efficiency argument for SLM-RAG.

Sub-1B models. Production deployments increasingly target sub-1B models, yet the evidence base measures 6–7B. Benchmarks and methods built directly for the sub-1B regime, rather than extrapolated from larger models, are needed before deployment claims at this scale can be trusted.

Deployment-aware evaluation. Evaluation that runs without cloud-LLM judges and that scores negative rejection as a primary metric would align measurement with the regime SLM-RAG actually targets. This is a benchmark-construction problem the field has not yet solved.

Self-updating and on-device RAG. The on-device setting, where training data and deployment data share users, motivates corpora that update incrementally without full re-indexing and without leaking user data. SimRAG's self-improvement loop and the DP-RAG line are early steps; making them practical at device scale is open.

Multimodal and agentic SLM-RAG. This survey covers text RAG. Retrieval over images, audio, and structured data at SLM scale is largely unexplored, as is reliable agentic behavior below 2B parameters, where current decomposition and self-filtering methods fail. Both are natural extensions once the text-RAG foundations above are firmer.

VII. Conclusion

Retrieval-augmented generation was designed with large readers in mind, and this survey has shown that the pipeline does not transfer cleanly to the small-model setting. The case for retrieval is nonetheless compelling: smaller models store less parametric knowledge and therefore stand to benefit more from it, as the Phi-3 authors observe of their own model[6], and as Huang and Huang demonstrate in showing that a retrieval-augmented 7B model can outperform a closed-book 70B model. Retrieval is most needed precisely where large models cannot go—in on-device, private, low-latency deployment.

The barriers, however, are specific and measurable. Retrieved passages strain tight context windows (2K for TinyLlama, 8K for Gemma 2); accuracy degrades sharply as retrieval noise rises; grounded hallucination in SLMs such as Llama-2-7B and Mistral-7B occurs at 5–10 times the rate of GPT-4 even when evidence is supplied; indexing quality deteriorates at smaller scales; and agentic, iterative-reasoning methods fail to perform reliably below 2B parameters.

Proposed solutions intervene at distinct points in the pipeline: retrieval-side filtering, generator-side training, architectural redesign, efficiency techniques, and SLM–LLM collaboration. Yet a tension runs through the field. Many methods marketed as SLM-native depend on a GPT-4-class teacher during training—for Self-RAG's reflection tokens, RAFT's chain-of-thought traces, and DRAG's reasoning traces. The shipped artifact is an SLM, but the pipeline that

produces it is not. Teacher-free results from InstructRAG, RoseRAG, and Collab-RAG show this dependence is not inherent, though it does not yet span the same range of tasks and model sizes as teacher-supervised training.

Across application domains—healthcare, privacy-restricted enterprise, on-device assistants, and education—SLM-RAG is adopted for privacy, cost, or sovereignty rather than accuracy, and hybrid local–cloud architectures dominate production. The binding constraint is typically not accuracy alone but accuracy under a fixed privacy budget, which reframes the central problem from improving small models in general to improving them within the constraints of their intended use.

Several open problems remain sharp: evaluating systems without cloud LLM judges, measuring performance at sub-2B scale, achieving the negative rejection that production deployments require, characterizing rather than merely documenting the privacy–utility curve, and determining which methods survive ternary-quantized hardware. None is solved, but all appear tractable. The field is young enough that better benchmarks and teacher-free training recipes could reshape it, yet mature enough that its barriers and candidate solutions are well understood. The central lesson of this survey is that SLM-RAG should be treated as a distinct design regime, not a miniaturized version of LLM-RAG.

REFERENCES

[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems (NeurIPS), 33, 9459–9474.
[2] Huang, Y., & Huang, J. (2024). A survey on retrieval-augmented text generation for large language models. arXiv preprint arXiv:2404.10981.
[3] Nguyen, C. V., Shen, X., Aponte, R., Xia, Y., Basu, S., Hu, Z., Chen, J., Parmar, M., Kunapuli, S., Barrow, J., Wu, J., Singh, A., Wang, Y., Gu, J., Dernoncourt, F., Ahmed, N. K., Lipka, N., Zhang, R., Chen, X., Yu, T., Kim, S., Deilamsalehy, H., Park, N., Rimer, M., Zhang, Z., Yang, H., Rossi, R. A., & Nguyen, T. H. (2024). A survey of small language models. arXiv preprint arXiv:2410.20011.
[4] Subramanian, S., Elango, V., & Gungor, M. (2025). Small language models (SLMs) can still pack a punch: A survey. arXiv preprint arXiv:2501.05465.
[5] Chen, L., & Varoquaux, G. (2024). What is the role of small models in the LLM era: A survey. arXiv preprint arXiv:2409.06857.
[6] Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A. A., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., et al. (2024). Phi-3 technical report: A highly capable language model locally on your phone. arXiv preprint arXiv:2404.14219.
[7] Fan, T., Wang, J., Ren, X., & Huang, C. (2025). MiniRAG: Towards extremely simple retrieval-augmented generation. arXiv preprint arXiv:2501.06713.
[8] Chen, L., Liu, Q., He, Y., Zhang, X., & Chu, X. (2025). DRAG: Distilling RAG for SLMs from LLMs to transfer knowledge and mitigate hallucination via evidence and graph-based distillation. arXiv preprint arXiv:2506.01954.
[9] Li, T., Tan, C., Lv, Y., Hu, Z., Xu, C., & Ling, Z. (2025). RoseRAG: Robust retrieval-augmented generation with small-scale LLMs via margin-aware preference optimization. arXiv preprint arXiv:2502.10993.
[10] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics (TACL), 12, 157–173.
[11] Gemma Team, Google DeepMind. (2024). Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118.
[12] Qwen Team. (2024). Qwen2.5 technical report. arXiv preprint arXiv:2412.15115.
[13] Zhang, P., Zeng, G., Wang, T., & Lu, W. (2024). TinyLlama: An open-source small language model. arXiv preprint arXiv:2401.02385.
[14] Apple Intelligence Foundation Language Models Team. (2024). Apple Intelligence foundation language models. arXiv preprint arXiv:2407.21075.
[15] Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2024). Self-RAG: Learning to retrieve, generate, and critique through self-reflection. International Conference on Learning Representations (ICLR).
[16] Yan, S.-Q., Gu, J.-C., Zhu, Y., & Ling, Z.-H. (2024). Corrective retrieval augmented generation. arXiv preprint arXiv:2401.15884.
[17] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), 3982–3992.
[18] Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., & Zaharia, M. (2022). ColBERTv2: Effective and efficient retrieval via lightweight late interaction. Proceedings of the 2022 Conference of the North American Chapter of the ACL (NAACL), 3715–3734.
[19] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. Advances in Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS).
[20] Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2023). MTEB: Massive text embedding benchmark. Proceedings of the 17th Conference of the European Chapter of the ACL (EACL), 2014–2037.
[21] Petroni, F., Piktus, A., Fan, A., Lewis, P., Yazdani, M., De Cao, N., Thorne, J., Jernite, Y., Karpukhin, V., Maillard, J., Plachouras, V., Rocktäschel, T., & Riedel, S. (2021). KILT: A benchmark for knowledge intensive language tasks. Proceedings of the 2021 Conference of the North American Chapter of the ACL (NAACL), 2523–2544.
[22] Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2024). RAGAS: Automated evaluation of retrieval augmented generation. Proceedings of the 18th Conference of the European Chapter of the ACL: System Demonstrations, 150–158.

[23] Li, J., Cheng, X., Zhao, W. X., Nie, J.-Y., & Wen, J.-R. (2023). HaluEval: A large-scale hallucination evaluation benchmark for large language models. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 6449–6464.
[24] Niu, C., Wu, Y., Zhu, J., Xu, S., Shum, K., Zhong, R., Song, J., & Zhang, T. (2024). RAGTruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. Proceedings of the 62nd Annual Meeting of the ACL, 10862–10878.
[25] Chen, J., Lin, H., Han, X., & Sun, L. (2024). Benchmarking large language models in retrieval-augmented generation. Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 38(16), 17754–17762.
[26] Qu, R., Tu, R., & Bao, F. (2024). Is semantic chunking worth the computational cost? arXiv preprint arXiv:2410.13070.
[27] Jiang, H., Wu, Q., Luo, X., Li, D., Lin, C.-Y., Yang, Y., & Qiu, L. (2024). LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. Proceedings of the 62nd Annual Meeting of the ACL, 1658–1677.
[28] Xu, F., Shi, W., & Choi, E. (2024). RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation. International Conference on Learning Representations (ICLR).
[29] Zhu, C., & Zeng, M. (2022). Impossible triangle: What's next for pre-trained language models? arXiv preprint arXiv:2204.06130.
[30] Zhang, T., Patil, S. G., Jain, N., Shen, S., Zaharia, M., Stoica, I., & Gonzalez, J. E. (2024). RAFT: Adapting language model to domain specific RAG. arXiv preprint arXiv:2403.10131.
[31] Wei, Z., Chen, W.-L., & Meng, Y. (2025). InstructRAG: Instructing retrieval-augmented generation via self-synthesized rationales. International Conference on Learning Representations (ICLR).
[32] Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., & Yih, W. (2024). REPLUG: Retrieval-augmented black-box language models. Proceedings of the 2024 Conference of the North American Chapter of the ACL (NAACL), 8371–8384.
[33] Wang, Z., Wang, Z., Le, L., Zheng, H. S., Mishra, S., Perot, V., Zhang, Y., Mattapalli, A., Taly, A., Shang, J., Lee, C.-Y., & Pfister, T. (2025). Speculative RAG: Enhancing retrieval augmented generation through drafting. International Conference on Learning Representations (ICLR).
[34] Guo, Z., Xia, L., Yu, Y., Ao, T., & Huang, C. (2024). LightRAG: Simple and fast retrieval-augmented generation. arXiv preprint arXiv:2410.05779.
[35] Gutiérrez, B. J., Shu, Y., Gu, Y., Yasunaga, M., & Su, Y. (2024). HippoRAG: Neurobiologically inspired long-term memory for large language models. Advances in Neural Information Processing Systems (NeurIPS).
[36] Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., & Larson, J. (2024). From local to global: A graph RAG approach to query-focused summarization. arXiv preprint arXiv:2404.16130.
[37] Jiang, Y., Fu, Y., Zhao, X., Ge, S., Han, D., Zhang, C., & Huang, Z. (2025). HyperRAG: Enhancing quality-efficiency tradeoffs in retrieval-augmented generation with reranker KV-cache reuse. arXiv preprint arXiv:2504.02921.
[38] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. International Conference on Learning Representations (ICLR).
[39] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. Advances in Neural Information Processing Systems (NeurIPS), 36.
[40] Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., & Han, S. (2024). AWQ: Activation-aware weight quantization for LLM compression and acceleration. Proceedings of Machine Learning and Systems (MLSys), 6.
[41] Frantar, E., Ashkboos, S., Hoefler, T., & Alistarh, D. (2023). GPTQ: Accurate post-training quantization for generative pre-trained transformers. International Conference on Learning Representations (ICLR).
[42] Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., Dong, L., Wang, R., Xue, J., & Wei, F. (2024). The era of 1-bit LLMs: All large language models are in 1.58 bits. arXiv preprint arXiv:2402.17764.
[43] Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2025). RouteLLM: Learning to route LLMs from preference data. International Conference on Learning Representations (ICLR).
[44] Chen, L., Zaharia, M., & Zou, J. (2024). FrugalGPT: How to use large language models while reducing cost and improving performance. arXiv preprint arXiv:2305.05176.
[45] Fu, Z., Chen, K., Yu, L., Li, Y., Jin, Y., & Wang, Z. (2025). A survey on collaborating small and large language models for performance, cost-effectiveness, cloud-edge privacy, and trustworthiness. arXiv preprint arXiv:2505.07460.
[46] Xu, R., Qi, Z., Wang, Z., Ahmed, N. K., Li, J., & Yang, C. (2025). Collab-RAG: Boosting retrieval-augmented generation for complex question answering via white-box and black-box LLM collaboration. arXiv preprint arXiv:2504.04915.
[47] Zhang, X., Li, S., Yang, X., Tian, C., Qin, Y., & Petzold, L. R. (2024). Enhancing small medical learners with privacy-preserving contextual prompting. International Conference on Learning Representations (ICLR). arXiv:2305.12723.
[48] Xu, R., Shi, W., Yu, Y., Zhuang, Y., Zhu, Y., Wang, M. D., Ho, J. C., Zhang, C., & Yang, C. (2024). BMRetriever: Tuning large language models as better biomedical text retrievers. arXiv preprint arXiv:2404.18443.
[49] Koga, T., Wu, R., Zhang, Z., & Chaudhuri, K. (2025). Privacy-preserving retrieval-augmented generation with differential privacy. arXiv preprint arXiv:2412.04697.
[50] Grislain, N. (2024). RAG with differential privacy. arXiv preprint arXiv:2412.19291.
[51] Cheng, Y., Zhang, L., Wang, J., Yuan, M., & Yao, Y. (2024). RemoteRAG: A privacy-preserving LLM cloud RAG service. arXiv preprint arXiv:2412.12775.
[52] Reza, Z., Mazur, A., Dugdale, M. T., & Ray-Chaudhuri, R. (2025). Small models, big support: A local LLM framework for teacher-centric content creation and assessment using RAG and CAG. arXiv preprint arXiv:2506.05925.
[53] Jeong, S., Baek, J., Cho, S., Hwang, S. J., & Park, J. C. (2024). Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. Proceedings of the 2024 Conference of the North American Chapter of the ACL (NAACL), 7036–7050.
[54] Xu, R., Liu, H., Nag, S., Dai, Z., Xie, Y., Tang, X., Luo, C., Li, Y., Ho, J. C., Yang, C., & He, Q. (2025). SimRAG: Self-improving retrieval-augmented generation for adapting large language models to specialized domains. arXiv preprint arXiv:2410.17952.