

Safe and Sufficient Caching: Validated Evidence Reuse for Production Retrieval-Augmented Generation

Akshay Verma
Amazon

Siddharth Pillai
Amazon

Sambit Das
Amazon

Deepak Gupta
Amazon

Abstract

Production Retrieval-Augmented Generation (RAG) systems serve repeated queries over evolving document corpora under user-specific access constraints. Standard pipelines perform full retrieval and generation for every request, wasting computation on recurring queries. Naive caching reduces latency but introduces safety failures: stale evidence from updated documents, unauthorized content crossing access boundaries, and superseded passages no longer reflecting source truth. These violations are invisible on academic benchmarks yet unacceptable in enterprise deployment.

We introduce SSC (Safe and Sufficient Caching), a validated evidence reuse framework for production RAG. SSC caches evidence chunks (not generated answers) with per-chunk provenance metadata, and enforces hard validity constraints before reuse: access-control gating ensures user authorization, and freshness verification confirms source-document currency. Only evidence passing both checks enters a utility-based ranking stage balancing relevance, source trust, historical utility, and serving cost.

This separation of safety from efficiency is the central contribution. On WebQuestions and HotpotQA, SSC maintains full quality parity with unrestricted retrieval baselines, confirming zero degradation from evidence reuse. On a proprietary InternalQA workload with role-based access control and document versioning, it reduces end-to-end latency by 22% and retrieval cost by 34% while achieving zero invalid reuse, compared to 3.4-9.8% violation rates for standard caching and 0.3-8.2% for TTL-based invalidation. The method is training-free, retriever-agnostic, and deploys as a thin validation layer in existing RAG pipelines.

1 Introduction

Retrieval-Augmented Generation (RAG) grounds LLM outputs in external knowledge (Lewis et al.,

2020; Izacard et al., 2023) and powers a growing range of production applications: enterprise search, customer support, policy compliance, and document-grounded assistants. In these settings, RAG systems serve thousands of requests per second over corpora that evolve through updates, access-control changes, and content deprecation. Despite this operational reality, most RAG research treats each query as independent, optimizing retrieval accuracy without modeling the repeated-query patterns and governance constraints that define production workloads.

This independence assumption creates a deployment tension. Production traffic exhibits substantial repetition: users across an organization query overlapping documents about the same topics. A natural response is caching, and systems like GPT-Cache (Bang et al., 2023) and RAGCache (Zhu et al., 2024) report significant latency reductions by reusing prior responses or retrieval states.

However, naive caching introduces safety failures that are invisible on academic benchmarks but pose real compliance risk:

- **Freshness violations:** A product team queries an internal policy document. The policy is updated the next day, but the cached evidence still reflects the old version. Downstream decisions rely on outdated guidance.
- **Access-control violations:** A manager retrieves evidence from a confidential HR document. A different employee issues a semantically similar query; the cached passage is served despite that employee lacking authorization.
- **Provenance drift:** A cached passage loses its link to the current source version, making it impossible to verify whether the evidence remains authoritative.

These failures cannot be caught by Exact Match or F1. A system may score well on public benchmarks while routinely leaking unauthorized con-

tent in deployment. This gap motivates treating evidence validity as a hard architectural constraint rather than a ranking signal.

We propose SSC (Safe and Sufficient Caching), a validated evidence reuse framework for production RAG. The key design choice: cache *evidence chunks with provenance metadata*, not generated answers or retriever outputs. Each cached chunk carries its source document ID, version hash, access-control list, trust score, and usage statistics. Before reuse, every chunk must pass hard validity checks (user authorization and source freshness). Only valid evidence enters a utility-based ranking stage balancing relevance, trust, historical usefulness, and serving cost.

This architecture separates safety from efficiency. The validity layer guarantees zero invalid reuse regardless of ranking decisions. The utility layer optimizes selection quality within the safe candidate set. The result is a system that achieves the latency benefits of caching (22% reduction) without any of the safety violations that make naive caching undeployable in governed environments.

SSC is task-agnostic, retriever-agnostic, and training-free. It operates as a thin validation layer between retrieval and generation, applicable to any RAG application (QA, summarization, report generation, policy lookup) without changes to the retriever, generator, or document store. It does not aim to improve retrieval accuracy over existing systems; rather, it makes any existing RAG pipeline production-safe and more efficient under repeated-query workloads.

Our contributions:

- A framework that caches evidence (not answers) with provenance metadata and enforces hard validity constraints, achieving zero invalid reuse across all evaluation conditions.
- A utility-aware ranking function combining relevance, trust, historical utility, and cost that optimizes selection quality within the safety-guaranteed candidate set.
- A production evaluation showing that SSC achieves 22% latency reduction and 34% retrieval cost savings while maintaining quality parity with full-retrieval baselines and zero safety violations, outperforming both naive caching (which violates safety) and TTL-based invalidation (which cannot achieve both safety and efficiency).

2 Related Work

Adaptive Retrieval-Augmented Generation. RAG grounds LLM outputs in external evidence (Lewis et al., 2020; Izacard et al., 2023; Ram et al., 2023). Recent work introduces adaptive retrieval control: DeepRAG (Guan et al., 2025) frames retrieval as iterative decomposition, RQ-RAG (Chan et al., 2024) refines ambiguous queries, AutoRAG (Kim et al., 2024) adapts iteration count to difficulty, and Self-RAG (Asai et al., 2024) teaches the generator to self-critique. These improve per-query retrieval quality but treat queries independently, without modeling reuse safety across repeated requests.

Evidence Filtering. ChunkRAG (Singh et al., 2024) applies LLM-based chunk scoring, AutoChunker (Jain et al., 2025) improves chunk coherence, BlendFilter (Wang et al., 2024) combines query blending with filtering, and RECOMP (Xu et al., 2024) compresses passages before augmentation. Cross-encoder reranking (Ren et al., 2021) improves precision at inference cost. All operate on freshly retrieved evidence per query without addressing safe reuse of previously validated evidence.

Semantic Caching for LLMs. GPTCache (Bang et al., 2023) caches generated responses by semantic query similarity. RAGCache (Zhu et al., 2024) extends caching to intermediate retrieval states. These optimize latency but do not enforce per-request validity: a cached response may become stale (document updated) or unauthorized (different user). Production systems typically add TTL-based expiry, which provides weak guarantees and cannot enforce user-specific access control.

Production RAG. Industry deployments face constraints beyond accuracy: latency budgets, access control, versioning, and compliance (Verma et al., 2026a,b). Prior adaptive RAG work addresses retrieval quality under noise but not reuse safety. SSC fills this gap: it caches evidence with provenance metadata and enforces hard validity constraints (access + freshness) before reuse, treating safety as a constraint rather than a ranking signal.

3 Methodology

3.1 Overview and Intuition

Consider an enterprise RAG system serving a product organization. An engineer asks: “*What is our data retention policy for customer logs?*” The system retrieves relevant policy chunks and generates a grounded answer. An hour later, a different engineer on the same team asks a nearly identical question. Without caching, the system repeats the full retrieval and generation pipeline. With naive caching, it may reuse the prior answer, but the policy document was updated 30 minutes ago, and the cached response is now stale.

SSC resolves this by caching *evidence chunks*, not answers. When the second query arrives, the system finds cached policy chunks in its evidence memory, verifies that (a) the requesting user has access permission, and (b) the source document has not changed since caching. If both checks pass, the cached evidence is reused for generation. If the document was updated, the stale chunks are discarded and fresh retrieval is triggered. This ensures the generated answer always reflects current, authorized evidence.

Formally, SSC formulates retrieval-augmented generation as a constrained evidence selection problem: select a compact evidence set for each query while satisfying access-control, freshness, and quality constraints under a latency budget.

3.2 Constrained Evidence Selection

Let $\mathcal{C}$ be a document corpus and Γ be an evidence memory storing previously validated chunks with provenance metadata. For query q from user u , we build a candidate pool from two sources:

$$\mathcal{B} = \mathcal{R}(q, \mathcal{C}) \cup \mathcal{M}(q, \Gamma), \quad (1)$$

where $\mathcal{R}$ returns freshly retrieved chunks and $\mathcal{M}$ returns memory-backed chunks matched to the query.

Each candidate $c \in \mathcal{B}$ must pass two hard validity checks:

$$A(c, u) = \mathbb{I}[\text{user } u \text{ is authorized for } c], \quad (2)$$

$$F(c) = \mathbb{I}[\text{source document of } c \text{ is unchanged}]. \quad (3)$$

The valid pool retains only chunks passing both:

$$\mathcal{B}_v = \{c \in \mathcal{B} : A(c, u)=1 \wedge F(c)=1\}. \quad (4)$$

These are hard constraints, not ranking signals. A stale or unauthorized chunk is removed before scoring, regardless of its semantic relevance. This is the safety guarantee.

3.3 Utility-Based Ranking

For each valid chunk, SSC computes a utility score:

$$U(c, q) = \beta_r R(c, q) + \beta_t T(c) + \beta_h H(c) - \beta_l L(c), \quad (5)$$

where $R(c, q)$ is retriever relevance, $T(c)$ is a source trust prior (derived from document type, ownership, publication channel), $H(c)$ is historical usefulness, and $L(c)$ is estimated serving cost. Weights $\beta_r, \beta_t, \beta_h, \beta_l \geq 0$.

Historical usefulness captures whether a chunk has previously supported satisfactory answers:

$$H(c) = \frac{\text{hits}(c) + 1}{\text{hits}(c) + \text{misses}(c) + 2}, \quad (6)$$

giving new evidence a neutral prior while promoting consistently useful chunks.

The final evidence set selects the top- K valid chunks:

$$E^* = \text{TopK}_{c \in \mathcal{B}_v} U(c, q), \quad (7)$$

and the answer is generated as $y = \mathcal{G}(q, E^*)$.

Why not rank-only? A system using only retriever relevance (the “valid evidence cache” baseline) achieves safety but misses optimization opportunities. Trust priors help when multiple valid chunks compete: an official policy document is preferred over an informal wiki page. Historical utility helps when relevance scores are similar: a chunk that previously produced satisfactory answers is promoted. Cost-awareness helps under latency constraints: between two equally valid chunks, the one already in memory (zero retrieval cost) is preferred.

3.4 Evidence Memory

The evidence memory Γ stores reusable chunks with provenance: chunk text, source document ID, version hash, embedding, access metadata, trust score, and usage counters. A chunk is admitted when its utility exceeds threshold θ_{admit} :

$$c \in \Gamma \quad \text{if} \quad U(c, q) \geq \theta_{\text{admit}}. \quad (8)$$

Cached evidence is never trusted by default. Every memory-backed chunk is rechecked at serving time against current permissions and document state. Chunks are evicted when their source changes, freshness fails, or historical utility remains low.

Algorithm 1 SSC serving procedure

Require: query q , user u , corpus $\mathcal{C}$, memory Γ

Ensure: grounded answer y

```
1:  $\mathcal{B} \leftarrow \mathcal{R}(q, \mathcal{C}) \cup \mathcal{M}(q, \Gamma)$ 
2:  $\mathcal{B}_v \leftarrow \{c \in \mathcal{B} : A(c, u) = 1 \wedge F(c) = 1\}$ 
3: for each  $c \in \mathcal{B}_v$  do
4:   compute  $U(c, q)$  using Eq. 5
5: end for
6:  $E^* \leftarrow \text{TopK}_{c \in \mathcal{B}_v} U(c, q)$ 
7: admit qualifying chunks to  $\Gamma$ 
8:  $y \leftarrow \mathcal{G}(q, E^*)$ 
9: return  $y=0$ 
```

3.5 Production Design Rationale

The design reflects three production constraints. First, enterprise traffic exhibits 30-50% query repetition (observed in our InternalQA workload), making validated reuse worthwhile. Second, user-specific access control means cached evidence cannot be shared solely because queries match semantically. Third, document corpora change continuously, requiring per-request freshness checks rather than time-based expiry.

SSC differs from response caching (which cannot adapt to evidence changes), retriever caching (which lacks per-chunk validity tracking), and TTL-based approaches (which trade safety for efficiency). It introduces a small validation and utility layer between retrieval and generation. The retriever, generator, and document store remain unchanged, making deployment incremental.

4 Experiments

We evaluate SSC on two axes: (1) quality preservation on public benchmarks (WebQuestions (Berant et al., 2013), HotpotQA distractor (Yang et al., 2018)), confirming that evidence reuse does not degrade generation, and (2) production safety and efficiency on a proprietary InternalQA workload with access control, document versioning, and repeated queries. InternalQA results report relative change against the no-reuse baseline.

4.1 Setup

Corpora. Public benchmarks embed gold passages into a 50M-passage Common Crawl corpus (signal-to-noise $< 0.0001\%$). InternalQA uses an internal document index with per-document access-control metadata and version history.

Query repetition characteristics. The public benchmarks and InternalQA differ fundamentally in workload structure. WebQuestions and Hot-

potQA are static evaluation suites where each question is processed once against a fixed corpus. As a result, opportunities for evidence reuse are negligible and the evaluation primarily measures whether reuse mechanisms preserve answer quality. InternalQA, in contrast, reflects a production deployment with recurring information needs, semantically similar requests from different users, document updates, and access-control constraints. These characteristics create substantial opportunities for safe evidence reuse and make efficiency-oriented retrieval optimization meaningful.

Infrastructure. All systems share a hybrid retriever (BM25 + Contriever (Izacard et al., 2022), $\lambda=0.4$) and GPT-4-turbo generator, consistent with recent baselines (Guan et al., 2025; Verma et al., 2026b). Hardware: $8 \times$ A100 GPUs (80GB).

Baselines. Published RAG systems for quality context: Vanilla RAG ($k=20$), Iterative RAG (Trivedi et al., 2023), AutoRAG (Kim et al., 2024), DeepRAG (Guan et al., 2025). Caching variants that isolate design choices: **No reuse** (full pipeline per query), **Answer cache** (response-level semantic caching), **Retriever cache** (passage-set caching without provenance), **Evidence cache** (chunk reuse without validity enforcement), **Valid ev. cache** (hard constraints, relevance-only ranking), **SSC** (hard constraints + full utility ranking).

Metrics. Public: Exact Match (EM) and F1. InternalQA: relative quality change ($\Delta\text{EM}/\Delta\text{F1}$), retrieval reduction, latency reduction, and invalid reuse rate (stale + unauthorized + superseded evidence, measured via metadata audit at serving time).

4.2 Main Results: Quality Preservation

Table 1 reports answer quality. The top block contextualizes quality against published RAG systems. The bottom block compares caching strategies. InternalQA columns report relative lift over the no-reuse baseline (which uses the same retriever and generator as DeepRAG).

The core observation is that SSC preserves answer quality while introducing no degradation from evidence reuse. On WebQuestions and HotpotQA, SSC matches the no-reuse baseline and DeepRAG exactly. This behavior is expected because both benchmarks are static evaluation datasets with virtually no query recurrence and no corpus evolution during evaluation. Under such conditions, evi-

Method	WebQ		HotpotQA		InternalQA	
	EM	F1	EM	F1	Δ EM	Δ F1
Vanilla RAG	52.3	68.7	41.8	59.4	-	-
Iterative RAG	56.1	71.4	45.2	62.8	+2.1	+1.8
AutoRAG	57.8	72.9	46.7	64.1	+2.3	+2.0
DeepRAG	60.4	75.2	49.3	66.7	+3.4	+2.6
No reuse	60.4	75.2	49.3	66.7	0.0	0.0
Answer cache	59.7	74.5	48.6	66.0	-0.9	-0.7
Retriever cache	60.2	75.0	49.1	66.5	-0.2	-0.1
Evidence cache	60.1	74.9	49.0	66.3	-0.1	-0.1
Valid ev. cache	60.4	75.2	49.3	66.6	0.0	0.0
SSC	60.4	75.2	49.3	66.7	+0.1	+0.1

Table 1: Answer quality. InternalQA reports relative lift over no-reuse (= DeepRAG-level system). SSC maintains full quality parity with the no-reuse baseline.

dence reuse has little opportunity to improve performance; the relevant question is whether reuse harms answer quality, which it does not.

In contrast, InternalQA exhibits the characteristics that motivate SSC: recurring information requests, overlapping user intents, document versioning, and access-control constraints. In this setting, previously validated evidence can be safely reused across related requests, producing a small positive quality effect (+0.1 pp) through utility-based ranking while simultaneously reducing retrieval cost. The public benchmark results therefore establish quality parity with state-of-the-art retrieval systems, while the InternalQA results demonstrate the practical value of safe evidence reuse in realistic production workloads.

Answer caching degrades quality (-0.7 to -0.9 pp on InternalQA) because cached responses cannot adapt when source documents change. The purpose of this table is not to show large accuracy improvements, but rather to demonstrate that SSC achieves its efficiency and safety benefits without sacrificing answer quality.

4.3 Safety and Efficiency: The Core Contribution

Unlike the public benchmarks, InternalQA contains substantial semantic overlap across requests and repeated access to the same underlying information sources. Such workloads are common in enterprise deployments, where users frequently revisit policies, documentation, operational procedures, and historical records. Consequently, InternalQA provides a realistic setting for evaluating whether evidence reuse can reduce retrieval cost while maintaining correctness, freshness, and access-control guarantees.

Method	Lat. Red.	Retr. Red.	Thru.	Invalid
Answer cache	34.0%	46.0%	1.48×	9.8%
Retriever cache	20.0%	31.0%	1.21×	3.4%
Evidence cache	25.0%	38.0%	1.31×	6.1%
Valid ev. cache	20.0%	32.0%	1.25×	0.0%
SSC	22.0%	34.0%	1.28×	0.0%

Table 2: Production efficiency and safety on InternalQA. Invalid: percentage of served responses containing stale, unauthorized, or superseded evidence.

Method	Reuse (%)	Lat. Red. (%)	Invalid (%)
TTL = 1h	12.0	5.0	0.3
TTL = 6h	28.0	14.0	1.8
TTL = 24h	41.0	21.0	4.7
TTL = 72h	48.0	25.0	8.2
SSC	34.0	22.0	0.0

Table 3: TTL-based invalidation vs. SSC. TTL forces a choice between safety and efficiency. SSC achieves both.

Table 2 reports the production metrics that constitute SSC’s primary contribution. All values are from InternalQA, relative to no-reuse.

The table reveals a fundamental trade-off in naive caching: higher efficiency correlates with higher violation rates. Answer caching achieves the best raw efficiency (34% latency, 1.48× throughput) but serves invalid evidence 9.8% of the time. In a production system handling 10K queries/hour, this means approximately 1,000 responses per hour contain unauthorized or outdated content.

SSC breaks this trade-off: 22% latency reduction and 1.28× throughput with zero violations. The 2 pp latency improvement over valid evidence cache (22% vs. 20%) comes from the cost term in the utility function, which prefers already-cached evidence when quality is comparable, increasing memory hit rate.

4.4 TTL Comparison

Time-to-live invalidation is the standard production approach. Table 3 demonstrates its limitations.

TTL approaches face an inherent dilemma: short windows (1h) minimize violations but provide negligible efficiency; long windows (72h) maximize efficiency but accumulate 8.2% invalid reuse. No TTL setting can achieve zero violations because time is a proxy for document state, not a guarantee. SSC achieves efficiency comparable to TTL=24h (22% vs. 21% latency reduction) with zero violations, because it checks actual document state per-request rather than estimating validity from elapsed

Variant	Δ Qual.	Δ Lat.	Δ Invalid
Full SSC	0.0	0.0	0.0
w/o access gate	0.0	-3.0	+4.2
w/o freshness gate	-0.2	-2.0	+5.1
w/o trust prior	-0.3	0.0	0.0
w/o historical utility	-0.4	0.0	0.0
w/o cost term	0.0	+4.0	0.0
w/o evidence memory	0.0	+18.0	0.0

Table 4: Ablation on InternalQA. Δ Lat. positive = slower. Δ Invalid: increase in violation rate.

Condition	Reuse	Lat. Red.	Invalid
Low repetition	18%	8%	0%
High repetition	46%	24%	0%
Document drift	31%	18%	0%
Role variation	28%	16%	0%

Table 5: Robustness on InternalQA. Safety guarantee holds across all production stress conditions.

time.

4.5 Ablation Study

Table 4 isolates each component on InternalQA. Values are relative to full SSC.

The components separate into three functional groups:

Safety gates (access, freshness): Removing either gate makes the system slightly faster (skipping a check) but introduces 4-5% invalid reuse. The freshness gate also causes a 0.2 pp quality drop because stale evidence produces worse answers. These are non-negotiable in production.

Quality signals (trust, historical utility): These rank evidence within the already-validated pool. Removing them has no effect on safety (gates are still active) or latency (they are metadata lookups, not retrieval calls). Historical utility contributes the most (-0.4 pp), confirming that tracking which evidence has previously supported good answers improves selection.

Efficiency mechanisms (cost term, evidence memory): The cost term saves 4% latency by preferring lower-overhead evidence when quality is comparable. The evidence memory provides 18% latency savings by eliminating repeated retrieval. Neither affects quality or safety.

4.6 Robustness Under Production Stress

Table 5 evaluates three conditions that stress the safety guarantees.

Under high query repetition (5+ occurrences), SSC reuses 46% of evidence with 24% latency

savings, the best efficiency scenario. Under document drift (frequent source updates), the freshness gate invalidates stale chunks while still achieving 31% reuse from unchanged documents. Under role variation (different access levels querying similar content), the access gate restricts evidence per-user while achieving 28% reuse from commonly accessible sources. Zero violations hold across all conditions.

5 Conclusion

We presented SSC, a validated evidence reuse framework that addresses a deployment gap in production RAG. The problem is concrete: enterprise RAG systems face repeated queries over access-controlled, evolving corpora. Naive caching reduces latency but introduces governance violations (stale content, unauthorized access). The standard mitigation, TTL-based expiry, forces a choice between safety and efficiency.

SSC resolves this trade-off through a principled separation. Hard validity constraints (access gating, freshness verification) guarantee zero invalid reuse. A utility function (relevance, trust, historical utility, cost) optimizes evidence selection within the safety-guaranteed set. The method does not aim to improve retrieval accuracy; it makes existing RAG pipelines production-safe and more efficient.

The results confirm this positioning. On public benchmarks, SSC matches unrestricted retrieval quality exactly. On a production enterprise workload, it achieves 22% latency reduction with zero violations, compared to 9.8% violation rates for answer caching and 4.7% for TTL=24h at similar efficiency levels. The ablation confirms clean functional separation: safety gates prevent violations, quality signals improve evidence selection, efficiency mechanisms reduce latency.

For practitioners deploying RAG in governed environments, the takeaway is that evidence validity must be an architectural constraint, not a post-hoc audit. SSC provides a deployable implementation of this principle.

Limitations

SSC requires structured metadata (access-control lists, version hashes, source authority signals) for each evidence chunk. In corpora where such metadata is incomplete or unreliable, the validity constraints may reject reusable evidence (overly conservative) or miss unauthorized content (insuffi-

ciently protective). Extending the approach to settings with partial metadata, where validity checks must be approximate, remains open.

The evidence memory introduces storage proportional to unique chunks encountered. Under low query repetition (Table 5), the hit rate may not justify this overhead. The method is most beneficial for enterprise workloads with moderate-to-high query overlap.

Our evaluation is limited to English-language QA. The interaction between evidence validity and multilingual retrieval, where document governance and versioning patterns differ across jurisdictions, has not been studied. Additionally, while our ablations isolate component contributions, the utility weight sensitivity (explored in Appendix C) shows that optimal weights may vary across deployment contexts, requiring per-system tuning.

Ethics Statement

This work improves the safety of evidence reuse in production RAG systems by enforcing access-control and freshness constraints. We view this as a positive contribution to responsible AI deployment, as it reduces the risk of serving unauthorized or outdated information to end users. The proprietary InternalQA dataset does not contain personally identifiable information. Public benchmarks (WebQuestions, HotpotQA) are widely available for research use.

References

- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. [Self-RAG: Learning to retrieve, generate, and critique through self-reflection](#). In *International Conference on Learning Representations*.
- Fu Bang et al. 2023. [GPTCache: An open-source semantic cache for LLM applications](#). In *Proceedings of the 3rd Workshop on Natural Language Processing for Programming (NLP4Prog)*.
- Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. [Semantic parsing on Freebase from question-answer pairs](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.
- Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yi-Ting Guo, and Jie Fu. 2024. [RQ-RAG: Learning to refine queries for retrieval augmented generation](#). *arXiv preprint arXiv:2404.00610*.
- Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. [DeepRAG: Thinking to retrieve step by step for large language models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. [Unsupervised dense information retrieval with contrastive learning](#). *Transactions on Machine Learning Research*.
- Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. [Few-shot learning with retrieval augmented language models](#). *Journal of Machine Learning Research*, 24(251):1–43.
- Arihant Jain, Purav Aggarwal, and Anoop Saladi. 2025. [AutoChunker: Structured text chunking and its evaluation](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 983–995.
- Dongkyu Kim, Byoungwook Kim, Donggeon Han, and Matouvs Eibich. 2024. [AutoRAG: Automated framework for optimization of retrieval augmented generation pipeline](#). *arXiv preprint arXiv:2410.20878*.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). *Advances in Neural Information Processing Systems*, 33:9459–9474.
- Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. [In-context retrieval-augmented language models](#). *Transactions of the Association for Computational Linguistics*, 11:1316–1331.
- Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. [RocketQAv2: A joint training method for dense passage retrieval and passage re-ranking](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835.
- Ishneet Sukhvinder Singh, Ritvik Aggarwal, Ibrahim Allahverdiyev, Muhammad Taha, Aslihan Akalin, Kevin Zhu, and Sean O’Brien. 2024. [ChunkRAG: Novel LLM-chunk filtering method for RAG systems](#). *arXiv preprint arXiv:2410.19572*.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. [Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037.

Akshay Verma, Swapnil Gupta, Deepak Gupta, Prateek Sircar, and Siddharth Pillai. 2026a. [SELENE: Selective and evidence-weighted LLM debating for efficient and reliable reasoning](#). In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 95–104.

Akshay Verma, Swapnil Gupta, Siddharth Pillai, Prateek Sircar, and Deepak Gupta. 2026b. [ReflectiveRAG: Rethinking adaptivity in retrieval-augmented generation](#). In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 377–384.

Haoyu Wang, Ruobing Tong, Zhiyu Yang, and Jiahao Wang. 2024. [BlendFilter: Advancing retrieval-augmented large language models via query generation blending and knowledge filtering](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. [RECOMP: Improving retrieval-augmented LMs with compression and selective augmentation](#). *Proceedings of the International Conference on Learning Representations*.

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

Chao Zhu, Hanlu Yang, Zili Lin, et al. 2024. [RAGCache: Efficient knowledge caching for retrieval-augmented generation](#). *arXiv preprint arXiv:2404.12457*.

A Hyperparameters

Fixed across all experiments: evidence memory capacity 500K chunks, admission threshold $\theta_{\text{admit}}=0.3$, utility weights $\beta_r=0.5$, $\beta_t=0.2$, $\beta_h=0.2$, $\beta_l=0.1$, evidence budget $K=20$, retriever BM25 + Contriever with $\lambda=0.4$. Weights were tuned on a held-out InternalQA subset and held constant across datasets.

B Latency Breakdown

Table 6 shows component-level latency for cache hits vs. misses (normalized to no-reuse retrieval cost).

On hits, retrieval is skipped entirely, saving 26% of end-to-end time. On misses, the memory lookup and validation add minimal overhead (total $1.03\times$ vs. $1.00\times$ for pure no-reuse). Generation dominates in both cases.

Component	Hit	Miss
Query encoding	1.00	1.00
Memory lookup	0.08	0.08
Validation	0.05	0.05
Retrieval	0.00	1.00
Scoring	0.12	0.18
Generation	0.72	0.78
Total	0.74	1.03

Table 6: Latency breakdown. Each value is the fraction of end-to-end time spent on that component (sums to Total).

C Utility Weight Sensitivity

Configuration	$\Delta\text{Qual.}$	Lat. Red.
Default ($\beta_r=.5, \beta_t=.2, \beta_h=.2, \beta_l=.1$)	+0.1%	22.0%
Relevance-only ($\beta_r=1.0$, rest 0)	0.0%	20.0%
Equal weights (all 0.25)	−0.1%	21.0%
High cost ($\beta_l=0.3$)	−0.2%	26.0%
High trust ($\beta_t=0.4$)	+0.2%	19.0%

Table 7: Utility weight sensitivity on InternalQA.

Performance is stable across configurations. Emphasizing cost increases latency savings at slight quality cost. Emphasizing trust improves quality but reduces reuse (high-trust evidence is less common in the memory).

D Implementation Notes

Each memory entry stores: chunk text (≤ 512 tokens), source document ID, SHA-256 version hash, 768-dim embedding, role-based access list, trust score, hit/miss counters, and timestamp. Memory is HNSW-indexed for sub-millisecond lookup.

Freshness: version hash comparison against document store (single metadata read). Access: set-intersection of user roles against chunk ACL (subs). Eviction: triggered by source change, low utility after 10+ uses, or LRU on capacity overflow.

E Failure Cases

Three failure patterns on InternalQA: (1) *Stale trust*: deprecated but not-yet-removed sources briefly receive inflated trust ($<0.1\%$ of queries). (2) *Rapid churn*: extreme document update frequency invalidates most cached evidence, reducing reuse benefit without safety violation. (3) *Cold start*: novel queries with no memory hits incur 3% overhead from empty lookup and validation.