

Safe and Sufficient Caching: Per-Request 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introduces safety failures: stale evidence from updated documents, unauthorized content crossing access boundaries, and superseded passages no longer reflecting source truth.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 whose premise is that cache-reuse validity in RAG is *context-conditioned*: admissibility depends on the requester and live source state, not on query similarity alone. SSC caches evidence chunks (not generated answers) with per-chunk provenance metadata, and treats validity as a hard constraint 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 production applications such as enterprise search, customer support, and document-grounded assistants. These systems serve heavy request volumes over corpora that evolve through updates, access-control changes, and content deprecation. Yet 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 creates a deployment tension. Production traffic is highly repetitive - users across an organization query overlapping documents about the same topics - and the natural response is caching. GPTCache (Bang et al., 2023) and RAGCache (Zhu et al., 2024) report large 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. Each is a violation along one dimension of the *request context*:

- **Source-state (freshness):** A team queries a regulatory-policy document; the policy is amended to satisfy a new regulation, but the cached evidence still reflects the superseded requirement, so an automated job grounded in it becomes non-compliant.
- **Requester (authorization):** 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.
- **Source-identity (provenance):** A cached passage cites a source version that has since been retracted, so an auditor cannot reproduce the citation and the answer fails review in a regu-

lated report.

These failures cannot be caught by Exact Match or F1. A system may score well on public benchmarks while routinely leaking unauthorized content in deployment. They share a single root cause that prior caching work does not model: classical caching assumes validity is *context-independent* (a cached entry is reusable until invalidated, identically for every reader), whereas in production RAG whether a chunk may be reused depends on the request context - who is asking, the live source state, and the governing policy. This gap motivates treating evidence validity as a *context-conditioned*, hard architectural constraint, evaluated per request rather than as a ranking signal.

We propose SSC (Safe and Sufficient Caching), a validated evidence reuse framework for production RAG. The key design choice is to 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, while the utility layer optimizes selection within the safe candidate set. The result captures caching’s latency benefit (22% reduction) without the violations that make naive caching undeployable in governed environments. SSC is task-agnostic, retriever-agnostic, and training-free as it acts as a thin validation layer between retrieval and generation that leaves the retriever, generator, and document store unchanged. It does not aim to improve retrieval accuracy; it makes existing RAG pipelines production-safe and more efficient under repeated-query workloads.

In summary, our contributions are:

- A formulation of cache reuse as *context-conditioned validity* where a chunk is reusable only if it satisfies validity predicates over the request context, of which access authorization and freshness are two instances, with room for further governance predicates.
- A constrained-selection framework that caches evidence (not answers) and treats validity as a hard feasible-set constraint, yielding a

by-construction soundness guarantee (zero invalid reuse) unavailable to time- or similarity-keyed caches, with utility ranking optimizing within the safe set.

- A production evaluation: 22% latency and 34% retrieval-cost reduction at quality parity with full retrieval and zero violations, outperforming both naive caching (unsafe) and TTL invalidation (which cannot achieve both).

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. Both assume *static* reuse validity (a cached item is reusable whenever a similar query recurs), whereas production RAG requires *request-conditioned* validity, where admissibility depends on the requester, live source state, and governing policy. TTL approximates only the time dimension with a fixed horizon. Our freshness gate instead enforces a read-time coherence invariant between the evidence memory and the document store - the strong-coherence corner of the classical invalidation-vs-TTL design space - and adds a second axis, read

authorization, that data-only coherence ignores; hence no TTL setting reaches zero violations (§4).

Production RAG. Industry deployments face constraints beyond accuracy: latency budgets, access control, versioning, and compliance (Verma et al., 2026a,b,c; Gupta et al., 2026). 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 a shared enterprise RAG assistant. A compliance officer asks “*What is the mandatory data-retention window for EU customer accounts?*”; the system retrieves and caches passages from a restricted regulatory-policy document. Hours later the policy is amended to shorten the window, yet a naive answer or passage cache still serves the superseded requirement-and an automated deletion job grounded in that answer is now itself non-compliant. Separately, a contractor issuing a semantically similar query is served the same passage despite lacking clearance for its source. One cache thus produces both a regulatory violation and an unauthorized disclosure-failures invisible to Exact Match or F1.

SSC resolves this by caching *evidence chunks*, not answers, and re-validating each cached chunk against the current request context-requester authorization and source-version freshness-before reuse, discarding any that fail and re-retrieving. The answer always reflects current, authorized evidence: the contractor never receives the restricted passage, and the stale retention figure is never served.

Formally, SSC casts retrieval-augmented generation as a constrained evidence selection problem: for each query, maximize a utility objective over a compact evidence set subject to hard validity constraints (access control, freshness) 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.

Classical caching (CPU caches, web caches, memoization) assumes *context-independent* validity: a cached entry is reusable until invalidated, identically for every reader. Production RAG breaks this assumption. Whether a chunk may be reused is a function of the *request context* $x=(u, t, g)$ -the requesting user u , the serving time t , and the governance state g of the source. We formalize this as a *context-conditioned validity* predicate $\Phi_i(c, x) \in \{0, 1\}$, a binary check of chunk provenance against the request context. Access control and freshness are two instances:

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

$$F(c) = \mathbb{K}[c\text{'s source version is unchanged}]. \quad (3)$$

The same interface admits further governance predicates (license compatibility, data residency, PII redaction, embargo); we instantiate and evaluate the two above and leave the rest as interface members. The valid pool retains chunks satisfying all active predicates-here A and F :

$$\mathcal{B}_v = \{c \in \mathcal{B} : \bigwedge_i \Phi_i(c, x)=1\}. \quad (4)$$

The predicates define a *feasible set*, not a ranking signal: $\mathcal{B}_v$ is the constraint region and the utility U below is the objective. A stale or unauthorized chunk is removed before scoring regardless of its relevance, so validity is lexicographically prior to utility; no finite-penalty relaxation $U - \mu(1 - \Phi)$ could substitute, since for any μ a sufficiently relevant but invalid chunk would re-enter. This yields the safety guarantee: served evidence satisfies Φ *by construction*, independent of weights or cache contents-a soundness property that no time-based (TTL) or similarity-keyed cache can state (conditional on metadata fidelity; see Limitations).

3.3 Utility-Based Ranking

For each valid chunk, SSC computes a utility score:

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

All four terms are normalized to $[0, 1]$ so the weighted sum is scale-consistent. $R(c, q) \in [0, 1]$

is the min-max-normalized hybrid retriever score (Eq. 1). $T(c) \in [0, 1]$ is a source trust prior: a fixed lookup mapping document type, ownership, and publication channel to a scalar (e.g., official policy 1.0, informal wiki 0.3). $L(c) \in [0, 1]$ is normalized serving cost, 0 for a memory hit and the retrieval-plus-rerank cost (relative to the budget) for a fresh chunk. The non-negative weights default to $\beta_r=0.5$, $\beta_t=0.2$, $\beta_h=0.2$, $\beta_l=0.1$ (summing to one); the method is training-free, so these are fixed hyperparameters tuned once on a held-out split and frozen across datasets (Appendix A, sensitivity in Appendix C).

Historical usefulness $H(c)$ estimates the probability that reusing c yields a satisfactory answer, where $\text{hits}(c)$ counts prior servings of c that produced a satisfactory answer (per the same task metric used for evaluation) and $\text{misses}(c)$ counts those that did not:

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

The $+1/+2$ is additive (Laplace) smoothing with a uniform Beta(1, 1) prior (Laplace, 1814), so an unseen chunk ($\text{hits}=\text{misses}=0$) receives the neutral value $H=1/2$ while evidence with a consistent track record is promoted.

The final evidence set selects the (at most) K valid chunks of highest utility:

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

with $K=20$ (ties broken by retriever relevance; if $|\mathcal{B}_v| < K$ all valid chunks are returned), and the answer is generated as $y = \mathcal{G}(q, E^*)$, where $\mathcal{G}$ is the frozen LLM generator.

Why not rank-only? Relevance-only selection (the **Validated cache** baseline) is safe but misses optimization opportunities the other terms capture: trust priors break ties toward authoritative sources (official policy over informal wiki), historical utility promotes chunks that previously produced satisfactory answers, and cost-awareness prefers already-cached evidence when quality is comparable.

3.4 Evidence Memory

The evidence memory Γ stores reusable chunks with provenance (text, source ID, version hash, embedding, access metadata, trust score, usage counters); a chunk is admitted when $U(c, q) \geq \theta_{\text{admit}}$ (with $\theta_{\text{admit}}=0.3$). Cached evidence is never trusted by default: every memory-backed chunk

Algorithm 1 SSC serving procedure

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

Ensure: grounded answer y

1: $\mathcal{B} \leftarrow \mathcal{R}(q, \mathcal{C}) \cup \mathcal{M}(q, \Gamma)$ (Eq. 1)

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

3: $E^* \leftarrow \text{TopK}_{c \in \mathcal{B}_v} U(c, q)$; admit qualifying c to Γ

4: **return** $y \leftarrow \mathcal{G}(q, E^*)=0$

is rechecked at serving time against current permissions and document state, and chunks are evicted when their source changes, freshness fails, or historical utility stays low.

3.5 Production Design Rationale

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, leaving the retriever, generator, and document store unchanged for incremental deployment.

The checks gate reuse, not retrieval. The candidate pool (Eq. 1) unions freshly retrieved and memory-backed chunks. On freshly retrieved evidence, $A(c, u)$ and $F(c)$ hold by construction, so the checks do no work without an evidence memory - “retrieval + guardrails” with no reuse is a no-op. They are load-bearing only because reused evidence may have changed authorization or version since admission. Efficiency therefore comes from the memory and not the guardrails. The ablation study (§4.5) attributes the latency reduction to the evidence memory and cost-aware selection, while removing a gate changes latency only marginally. The exact “reuse + two checks” system is the **Validated cache** baseline, which SSC dominates via utility ranking over the validated pool. The scientific core is context-conditioned validity plus utility-based selection over the safe set-not the checks in isolation.

4 Experiments and Results

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 is an enterprise workload of $\sim 120\text{K}$ questions over an internal index of 2.3M documents carrying per-document access-control lists (14 distinct roles) and full version history. It exhibits the production characteristics absent from public benchmarks: 38% of questions recur within a 24h window, 9% of source documents are revised over the evaluation period, and queries span all 14 roles, so the same passage is authorized for some requesters and not others. These properties are what make safe reuse both worthwhile and non-trivial.

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: $8 \times \text{A100 GPUs}$ (80GB).

Baselines. We utilize these published RAG systems for quality context: Vanilla RAG ($k=20$), Iterative RAG (Trivedi et al., 2023), AutoRAG (Kim et al., 2024), and DeepRAG (Guan et al., 2025). Caching variants that isolate design choices are: **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), **Validated cache** (evidence reuse + both hard validity checks + relevance-only ranking - i.e. the “reuse with two guardrails” configuration, isolating the contribution of utility ranking), and **SSC** (hard constraints + full utility ranking).

Metrics. For Public datasets, we evaluate Exact Match (EM) and F1. For InternalQA dataset, we evaluate 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). For the historical-utility counters (Eq. 6), a serving is a *hit* when its answer meets the task threshold (token-F1 ≥ 0.6 against the reference) and a *miss* otherwise.

Method	WebQ		HotpotQA		InternalQA	
	EM	F1	EM	F1	ΔEM	ΔF1
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
Validated cache	60.4	75.2	49.3	66.7	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.

Method	Lat. Red.	Retr. Red.	Thru.	Invalid
Answer cache	34.0%	46.0%	1.48×	9.8%
Retriever cache	20.0%	31.0%	1.25×	3.4%
Evidence cache	25.0%	38.0%	1.31×	6.1%
Validated 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.

4.2 Quality Preservation

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

This table confirms that SSC’s efficiency and safety benefits (shown next) come at zero quality cost. On WebQuestions and HotpotQA - static suites with negligible recurrence - SSC matches no-reuse and DeepRAG exactly: reuse cannot help here, and the question is whether it harms quality, which it does not. On InternalQA, whose recurring requests and versioning motivate SSC, utility ranking yields a small +0.1 pp effect. Answer caching instead degrades quality (-0.7 to -0.9 pp) because cached responses cannot adapt when source documents change.

4.3 Safety and Efficiency

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

Naive caching shows a fundamental trade-off: higher efficiency correlates with higher violation rates. Answer caching attains the best raw effi-

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.

ciency (34% latency, $1.48\times$ throughput) but serves invalid evidence 9.8% of the time - roughly 1,000 of every 10K responses. SSC breaks this: 22% latency reduction and $1.28\times$ throughput with zero violations. In absolute terms, mean end-to-end latency drops from 3.18s (no reuse) to 2.48s, and the effect is larger at the tail - p95 falls from 6.4s to 4.7s (27%) - because memory hits skip the retrieval stage whose cost dominates slow requests. Its 2 pp mean gain over the **Validated cache** row comes from the cost term raising the memory hit rate. That row is exactly the “reuse + two checks” configuration, which SSC dominates (+0.1 quality) through utility ranking - the checks establish a safe pool and the contribution is what SSC does *within* it.

4.4 TTL Comparison

Time-to-live invalidation is the standard production approach. However TTL has its limitations, which are depicted in Table 3.

TTL faces an inherent dilemma: short windows (1h) minimize violations but yield negligible efficiency; long windows (72h) maximize efficiency but accumulate 8.2% invalid reuse. No setting reaches zero violations because time is a proxy for document state, not a guarantee. SSC matches TTL=24h efficiency (22% vs. 21%) with zero violations by checking actual document state per request.

4.5 Ablation Study

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

The components separate into three groups. *Safety gates* (access, freshness): removing either is marginally faster not by skipping a sub-ms check but by admitting *unsafe* reuse, hence the 4–5% jump in invalid reuse (freshness removal also costs 0.1 pp quality); these are non-negotiable. *Quality signals* (trust, historical utility): metadata lookups

Variant	Δ Qual.	Δ Lat.	Δ Invalid
Full SSC	0.0	0.0	0.0
w/o access gate	0.0	−2.0	+4.2
w/o freshness gate	−0.1	−1.0	+5.1
w/o trust prior	−0.1	0.0	0.0
w/o historical utility	−0.1	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.

ranking within the validated pool, with no safety or latency effect; each costs ≈ 0.1 pp, jointly the +0.1 pp lift over the relevance-only **Validated cache** baseline. *Efficiency mechanisms*: the cost term saves 4% latency and the evidence memory 18% (eliminating repeated retrieval), neither affecting quality or safety.

Robustness under stress. Across four stress conditions (Appendix E), efficiency scales with reuse (11% latency at low repetition, 30% at high), and the gates hold under document drift (31% reuse from unchanged sources) and role variation (28% reuse from commonly accessible ones)—zero violations throughout.

5 Conclusion

SSC treats cache-reuse validity in production RAG as a hard, context-conditioned constraint that guarantees zero invalid reuse while utility ranking optimizes within the safe set, cutting latency 22% and retrieval cost 34% at quality parity. For governed deployments, evidence validity must be an architectural constraint, not a post-hoc audit.

Because validity is expressed as a predicate family over the request context, access control and freshness are instances rather than special cases: the same interface extends to licensing, data-residency, PII-redaction, and embargo constraints without retraining or retriever changes. We see two open directions: (i) approximate predicates for corpora with partial or noisy provenance metadata, where the by-construction guarantee must degrade gracefully, and (ii) learned utility weights that adapt per deployment while leaving the hard validity layer untouched.

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 (insufficiently 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 7), 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, Zequi 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)*.
- Swapnil Gupta, Akshay Verma, Khushi Gupta, Prateek Sircar, and Deepak Gupta. 2026. [Csmad: Hallucination detection via multi-agent debate with nli-verified contradictory statements](#). In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26)*, pages 4626–4630. ACM.
- 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*.
- Pierre-Simon Laplace. 1814. *Essai philosophique sur les probabilités*. Courcier, Paris.
- 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.

Akshay Verma, Siddharth Pillai, Prateek Sircar, and Deepak Gupta. 2026c. **NEST: Nested evidence survival for retrieval**. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 507–516, San Diego, California, USA. Association for Computational Linguistics.

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 5 shows component-level latency for cache hits vs. misses, with each component expressed as a fraction of the no-reuse end-to-end latency (so the no-reuse pipeline sums to 1.00 and the columns sum to the stated totals).

Component	Hit	Miss
Query encoding	0.05	0.05
Memory lookup	0.01	0.01
Validation	0.00	0.00
Retrieval	0.00	0.66
Scoring	0.04	0.05
Generation	0.24	0.24
Total	0.34	1.01

Table 5: Latency breakdown, normalized to no-reuse end-to-end latency (= 1.00). Columns sum to the stated totals.

At this corpus scale (50M passages, hybrid retrieval), retrieval is the dominant cost in the no-reuse pipeline. A cache hit skips retrieval entirely, saving $\approx 66\%$ of per-query latency; generation is then the dominant remaining component. On misses, memory lookup and validation add only $\approx 1\%$ overhead (total $1.01\times$). Blending hits and misses at the observed 34% reuse rate gives $0.34\times 0.34 + 0.66\times 1.01 \approx 0.78$, i.e. the reported 22% end-to-end latency reduction; the validation checks themselves (sub-ms) contribute negligibly to this saving.

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 6: 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 Robustness Under Production Stress

Table 7 reports the four stress conditions summarized in §4.5. The zero-violation guarantee holds in every case.

Condition	Reuse	Lat. Red.	Invalid
Low repetition	18%	11%	0%
High repetition	46%	30%	0%
Document drift	31%	20%	0%
Role variation	28%	18%	0%

Table 7: Robustness on InternalQA. Safety guarantee holds across all production stress conditions.

F 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.