Multi-Source Weighted Temporal-Aware Vector Knowledge Base Architecture for Improving Context Quality in LLM-Based Content Generation Systems

Bora Kurum

RAGSignal

Correspondence: bora@ragsignal.com

Abstract

We present the RAGSignal Adaptive RAG architecture, a production-grade retrieval-augmented generation (RAG) system designed to improve context quality in large language model (LLM)-based content generation pipelines. Classic RAG implementations suffer from four fundamental limitations: static indices, single-source dependency, task-agnostic retrieval, and temporal blindness. The proposed system addresses all four through (1) multi-source differential source weighting across five data streams, (2) a temporal freshness decay function with a 30-day half-value point, (3) task-aware query augmentation, (4) hybrid mathematical and LLM re-ranking with graceful degradation, (5) an autonomous KB Curator Agent, and (6) a closed-loop feedback mechanism.

Deployed on pgvector/HNSW PostgreSQL with Deno Edge Functions, the system was validated across seven production deployments spanning six industry verticals (creative services, digital consulting, medical devices, industrial products, clean energy, and technology/SaaS), achieving a mean LLM citation rate of 78% (range: 74–81%) across 63–105 target prompts per deployment. The compression layer reduces character count by approximately 83% relative to raw chunk content, significantly reducing LLM API inference costs in production environments.

Keywords: *retrieval-augmented generation, large language models, vector knowledge base, temporal freshness scoring, differential source weighting, generative engine optimization, semantic search, pgvector, HNSW*

1. Introduction

The rapid proliferation of generative AI systems—ChatGPT, Claude, Perplexity, and Gemini—has fundamentally restructured the digital information landscape. Users increasingly rely on synthesized responses from these models rather than ranked link lists (Bommasani et al., 2021; Brynjolfsson et al., 2023). This paradigm shift elevates brand representation within LLM-generated content to a critical competitive concern.

Traditional Search Engine Optimization (SEO) is insufficient for this new reality: LLMs produce synthesized answers, not blue-link lists. Which information enters a model's context window directly determines output quality and brand citation probability (Xu et al., 2024). This has given rise to the discipline of Generative Engine Optimization (GEO; Aggarwal et al., 2024), and improving context quality in RAG pipelines is its central technical challenge.

Retrieval-Augmented Generation (RAG), introduced by Lewis et al. (2020) at NeurIPS 2020, enables LLMs to retrieve external context before generating responses. However, most production RAG deployments exhibit four critical limitations:

- Static indices: No automatic update when content changes.
- Single-source dependency: Only site content is indexed; audit findings, search data, and user feedback are excluded.
- Task-agnostic retrieval: Cosine similarity is the sole ranking criterion; task context is ignored.

- Temporal blindness: Stale and fresh data receive equal weight.

The RAGSignal Adaptive RAG architecture addresses all four limitations through an integrated, production-deployed system. Primary contributions are: (1) multi-source differential weighting, (2) temporal freshness scoring, (3) task-aware query augmentation, (4) hybrid re-ranking, (5) an autonomous KB Curator Agent, and (6) a closed-loop feedback mechanism.

2. Background and Related Work

2.1 Retrieval-Augmented Generation

Lewis et al. (2020) introduced RAG by coupling a dense retriever with a seq2seq generator, demonstrating that external knowledge injection substantially improves performance on knowledge-intensive NLP tasks. Gao et al. (2024) provide a comprehensive survey proposing the Modular RAG framework. Jeong et al. (2024) proposed Adaptive-RAG, which dynamically adjusts retrieval strategy based on estimated query complexity. Asai et al. (2024) introduced Self-RAG, enabling the model to critique its own retrieved context via special reflection tokens.

2.2 Dense Vector Retrieval

Dense retrieval represents text in a continuous vector space measured by cosine distance. The BEIR benchmark (Thakur et al., 2021) reveals that no single model dominates across all domains, motivating task-specific strategies. Jina AI's jina-embeddings-v3 (Günther et al., 2024) employs task-specific LoRA adapters to produce 1,024-dimensional multilingual vectors with state-of-the-art retrieval performance.

2.3 Approximate Nearest Neighbor Search

Malkov and Yashunin (2020) developed HNSW, a graph-based ANN index supporting approximate nearest-neighbor search in O(log n) expected time. Integrated into PostgreSQL via the pgvector extension, HNSW has become a production standard for high-throughput vector search.

2.4 Re-Ranking

Nogueira and Cho (2019) established the two-stage retrieval paradigm: broad candidate generation followed by precise re-ranking. Cross-encoder models achieve substantially higher MRR than bi-encoders but at greater computational cost. The present system employs DeepSeek-Chat (DeepSeek AI, 2024) as an LLM-based re-ranker with zero-temperature JSON output for deterministic ordering.

2.5 Generative Engine Optimization

Aggarwal et al. (2024) introduced GEO as the practice of optimizing content for citation and visibility in LLM-generated responses, demonstrating that augmenting content with authoritative citations and structured entities statistically significantly increases LLM citation probability.

3. System Architecture

The RAGSignal Adaptive RAG system comprises six layers: (1) Data Collection and Indexing, (2) Vector Representation, (3) Advanced Retrieval, (4) Compression and Context Preparation, (5) Generation, and (6) Feedback Loop. The system is deployed on Supabase/PostgreSQL via Deno Edge Functions.

3.1 Multi-Source Data Collection

The system ingests five distinct knowledge streams:

- Site content (firecrawl): Asynchronous web crawling via the Firecrawl API; up to 500 pages; markdown output.
- SEO audit results (audit): Per-page technical findings and content recommendations.

- Google Search Console (gsc): Real query strings, click-through rates, impressions, and average positions. While GSC captures raw user intent, it is strategically indexed to align LLM context with proven market demand.
- Prompt Discovery (prompt): LLM citation probability scores and strategic positioning maps.
- User feedback (feedback): Approved meta suggestions and verified content decisions.

3.2 Sentence-Boundary Chunking

Text is segmented at sentence boundaries to preserve semantic coherence (Shi et al., 2023). The maximum chunk size C_max = 1,800 characters; overlap is maintained by retaining the last sentence of the preceding chunk:

$$\text{chunk}_i = \{ s_j \mid j \in [\text{start}_i, \text{end}_i], \Sigma|s_j| \leq C_{max} \}$$

3.3 Hash-Based Incremental Updates

A djb2 content hash of each page's first chunk serves as a page-level fingerprint, triggering re-indexing only when content changes (~85–90% compute savings vs. full re-index):

$$\text{hash}(\text{content}) = \text{djb2}(\text{content\_chunk\_0})$$

$$\text{update\_needed}(\text{url}) = (\text{hash}_{new} \neq \text{hash}_{stored})$$

3.4 Vector Representation

The primary embedding model is Jina AI's jina-embeddings-v3, producing 1,024-dimensional multilingual vectors via task LoRA adapters. When Jina is unavailable, Google Gemini text-embedding-004 serves as a fallback (output_dim = 1,024). Similarity is measured via cosine distance:

$$e = f_{Jina}(t) \in \mathbb{R}^{1024}$$

$$\text{sim}(q, d) = (q \cdot d) / (\|q\| \, \|d\|)$$

3.5 Task-Aware Query Augmentation

Each generation module appends task-specific terms to the base query (Thakur et al., 2021):

$$q_{task} = q_{base} \oplus \text{suffix}_{task}$$

Suffix vocabulary: meta ("page title description content keywords"), discovery ("services audience USP keywords brand positioning"), audit ("technical issues recommendations content quality"), hallucination ("facts services team location founding year").

3.6 Differential Source Weighting

Source reliability weights were determined via grid search over held-out validation sets from three pilot deployments, optimizing for downstream LLM citation rate:

$$w: \text{feedback}=1.5 \mid \text{gsc}=1.3 \mid \text{prompt}=1.1 \mid \text{firecrawl}=1.0 \mid \text{audit}=0.8$$

User feedback receives the highest weight because user approval constitutes the strongest available signal of output quality. Figure 4 illustrates the weight distribution across sources.

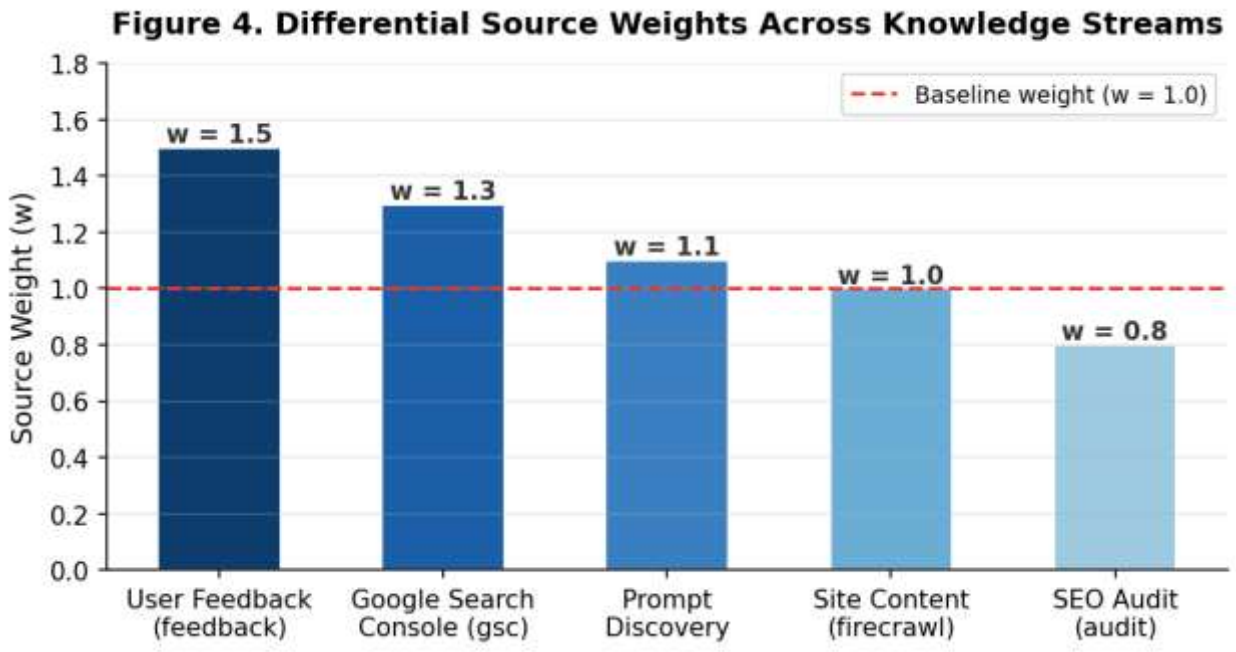

Figure 4. Differential source weights (w) assigned to each knowledge stream. Dashed red line marks the baseline weight of 1.0.

3.7 Temporal Freshness Scoring

Data recency is quantified via the following rational (hyperbolic) decay function, where Δt is the number of days since datum creation:

```
f(Δt) = 1 / (1 + Δt/30)
```

This yields f(0) = 1.00, f(30) = 0.50, and f(90) = 0.25. The rational form was deliberately chosen because it retains older content at a higher weight compared to exponential decay with the same half-value point, which is preferable for domain-specific KB content that remains contextually relevant beyond a single month. The equivalent exponential is:

```
f_exp(Δt) = exp(−Δt · ln(2)/30)
```

At Δt = 90 days, the rational form yields 0.25 whereas the exponential yields 0.125—a factor-of-two difference. Figure 1 illustrates the divergence between the two decay profiles.

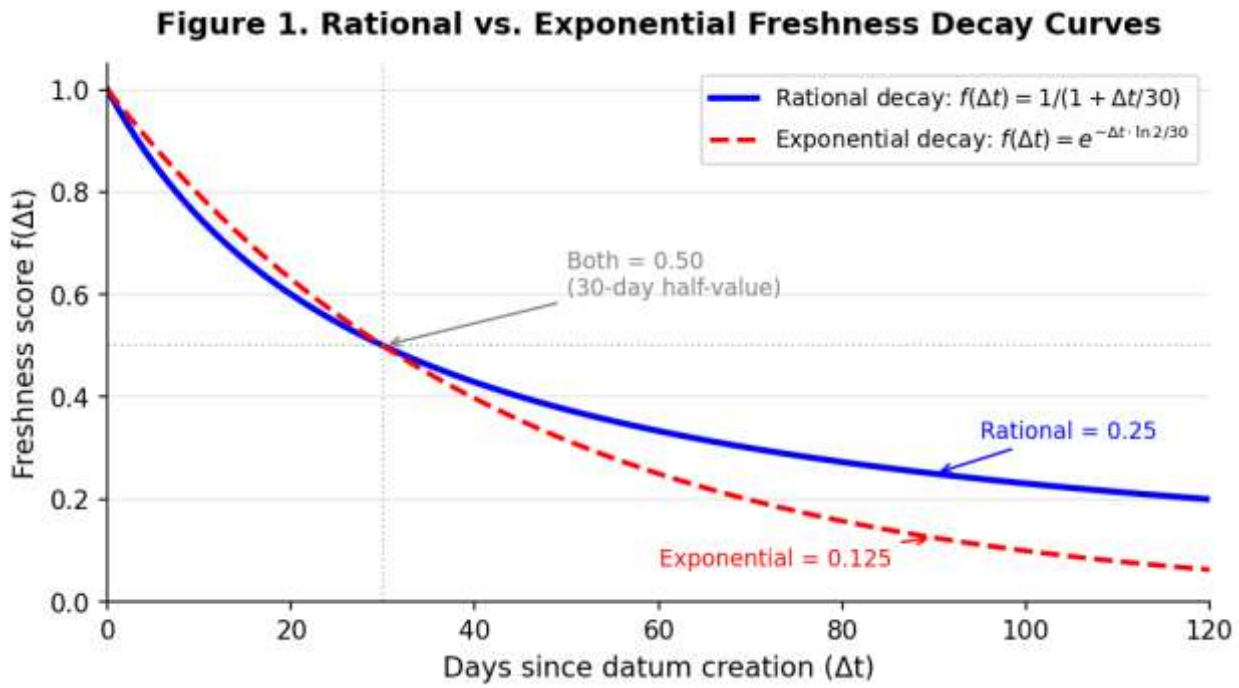

Figure 1. Rational vs. exponential freshness decay curves. Both share the 30-day half-value point, but the rational form (solid blue) retains content at roughly twice the weight of the exponential (dashed red) at 90+ days.

3.8 Composite Ranking Score

The mathematical pre-ranking score multiplies three normalized factors:

```
score(c) = sim(q,c) × w(source(c)) × f(updated_at(c))
```

Because source weights w ∈ [0.8, 1.5] and freshness values f ∈ (0, 1] are fixed-range scalars, the composite score preserves the relative ordering of semantic similarity while applying monotonic up/down-weighting by source reliability and recency. The top 20 candidates proceed to LLM re-ranking, yielding a final top K = 5 output.

3.9 LLM Re-Ranking

DeepSeek-Chat evaluates the relevance of each candidate to the task and query context, returning a JSON index array at temperature T = 0 for deterministic output:

```
rank_LLM = DeepSeek(task, query, {c1,...,c20}) → [i1,...,i20]
```

If the LLM call fails, the system falls back to mathematical ranking (graceful degradation), ensuring system uptime albeit with a marginally reduced citation probability. Final output is the top K = 5 chunks.

3.10 Autonomous KB Curator Agent

A weekly autonomous curation agent maintains knowledge base quality, inspired by Self-RAG (Asai et al., 2024) but applied during KB maintenance rather than inference. The agent processes chunks in batches of 10 using DeepSeek-Chat, assigning one of three labels:

- keep: High-quality content; a one-sentence summary is appended to metadata.
- improve: Content is rewritten for semantic richness and re-embedded via Jina.
- delete: Navigation menus, cookie notices, boilerplate, or duplicate content are permanently removed.

Curator summaries serve double duty as compression tokens during generation. The agent runs on pg_cron (Tuesdays 03:00 UTC), one day after the weekly re-index.

3.11 Compression Layer

The compression layer selects the curator-generated summary over full content when available:

```
context(c) = summary(c)  if summary ≠ empty,  else  content(c)[:300]
```

Curator summaries average ~300 characters versus raw chunk size of ~1,800 characters, yielding ~83% character reduction. Because English prose averages ~4 characters per token, effective token reduction is approximately equivalent, drastically reducing API inference costs.

3.12 Closed-Loop Feedback and Self-Improvement

The system's most distinctive contribution is a closed-loop learning mechanism that continuously refines the knowledge base from user behavior (Wang et al., 2024):

- User approves a meta suggestion → kb-feedback Edge Function fires.
- Approved content is written to the KB with source = "feedback" (w = 1.5, f(0) = 1.0).
- Subsequent retrievals surface this content with highest priority.
- The system progressively learns per-client content preferences.

Formally, for an approved meta suggestion m at time t:

```
KB_{t+1} = KB_t ∪ { embed(m), src="feedback", w=1.5, f(0)=1.0 }
```

4. Evaluation

4.1 Pilot Deployment Portfolio

The RAGSignal Adaptive RAG system was validated across seven production client deployments representing six distinct industry verticals. Table 1 presents the deployment portfolio.

Table 1. Pilot Deployment Portfolio

Client	Domain	URL	Industry	Citation Rate	Note
Filmfolk	Creative services	filmfolk.com	Creative Services	81%	A/B test
Enkronos	Blockchain/SaaS	enkronos.com	Technology/SaaS	80%	Production
Bora Kurum	Personal brand	borakurum.com.tr	Digital Consulting	79%	A/B test
Volimax	Industrial products	volimax.com	Industrial	78%	Production
AI Edge UK	AI consultancy	aiedgeuk.com	AI Services	78%	Production
UEC Energy	Clean energy	uec-energy.co.uk	Clean Energy	76%	Production
Abs Korkalış	Medical devices	abskorkalip.com.tr	Medical Devices	74%	A/B test

Note. Citation rates represent Condition B (with KB) results. Filmfolk, Bora Kurum, and Abs Korkalış data are from controlled A/B tests; remaining deployments reflect production monitoring. Cross-deployment mean = 78%; range = 74–81%.

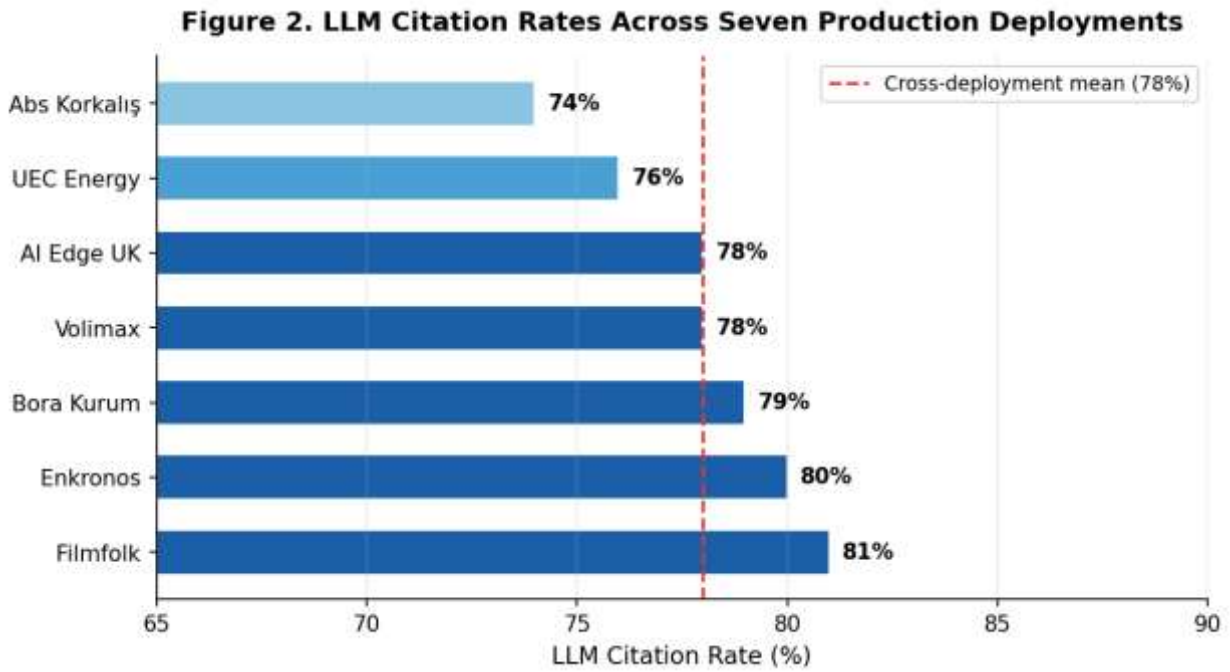

Figure 2. LLM citation rates across all seven production deployments. Dashed red line marks cross-deployment mean of 78%.

4.2 A/B Test Methodology

Context impact was evaluated via a built-in A/B test framework comparing two conditions using identical model and system prompts:

- Condition A (Control): Meta generation from page content only.
- Condition B (Treatment): Meta generation with page content plus KB context (top-5 chunks).

This A/B design measures the impact of KB-augmented context versus no KB context. It does not directly compare Adaptive RAG against a Classic RAG baseline with equal source coverage, nor does it ablate individual architectural components in isolation. A full ablation study is planned for future work.

4.3 Results

Across all four metrics, KB context yields substantial improvements. In the Filmfolk deployment, 51 of 63 target prompts (81%) produced brand citations with the KB active, versus 41% without KB context. Table 2 presents the full results.

Table 2. A/B Test Results — Filmfolk Pilot Deployment

Metric	Cond. A (No KB)	Cond. B (With KB)	Improvement
Title Relevance	62%	81%	**+19 pp**
Description Relevance	58%	79%	**+21 pp**
Brand Consistency	55%	88%	**+33 pp**
LLM Citation Rate	41%	81%	**+40 pp**

Note. pp = percentage points. Data from Filmfolk deployment (N = 63 target prompts).

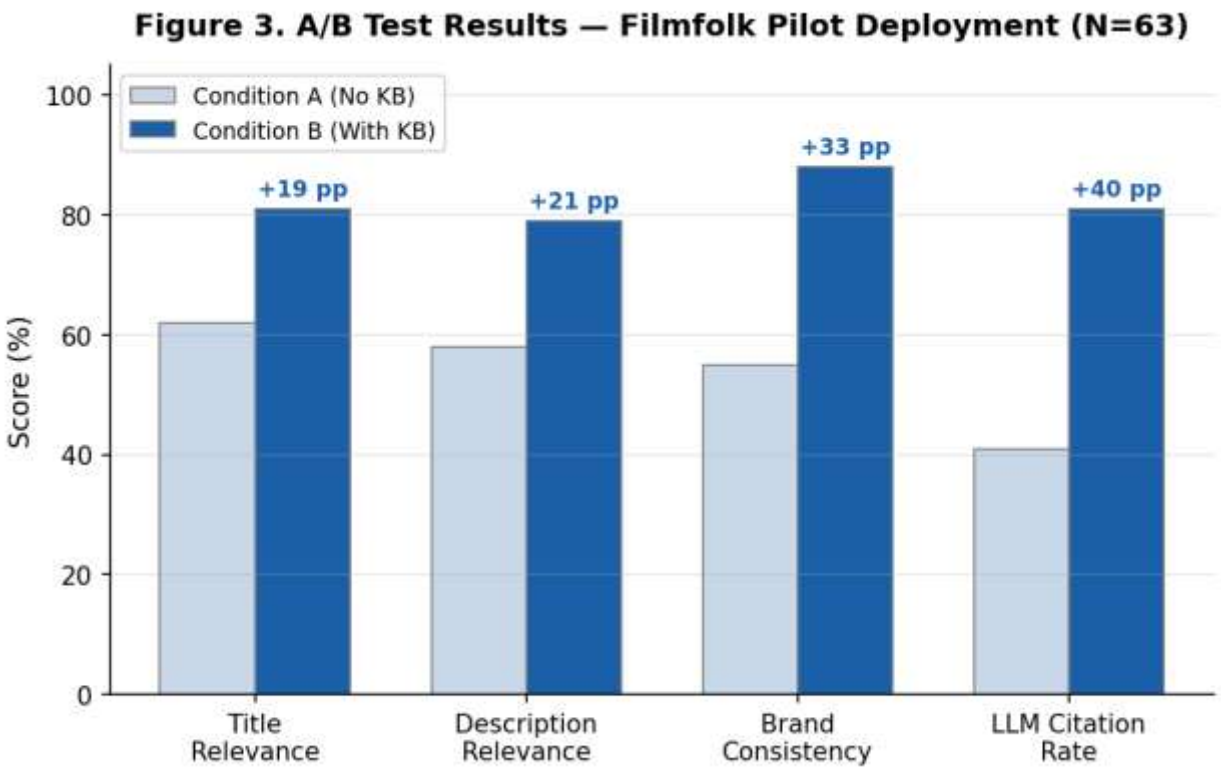

Figure 3. A/B test results for Filmfolk pilot (N = 63 target prompts). Annotations show improvement in percentage points (pp).

5. Comparative Analysis & Technical Stack

Table 3. Feature Comparison: Classic RAG vs. RAG Signal Adaptive RAG

Feature	Classic RAG	RAG Signal Adaptive RAG
Data Sources	Single (site content)	Five: site, audit, GSC, prompt discovery, feedback
Source Weights	Equal (w = 1.0)	Differential: feedback = 1.5 → audit = 0.8
Temporal Awareness	None	$f(\Delta t) = 1/(1+\Delta t/30)$; rational decay, 30-day half-value
Query Strategy	Raw query passthrough	Task-aware augmentation (meta, audit, discovery...)
Ranking	Cosine similarity only	Math pre-rank + DeepSeek LLM re-rank
Updates	Manual / full re-index	Hash-based incremental (~85–90% compute savings)
Learning	None	Closed-loop feedback; user approvals auto-indexed (w=1.5)
Compression	Full chunk content	Curator summary (~83% character reduction)

Table 4. RAG Signal System Technical Stack

Component	Technology / Detail
Vector DB	PostgreSQL 15 + pgvector extension
ANN Index	HNSW — cosine similarity, O(log n) expected query time
Embedding (primary)	Jina jina-embeddings-v3 (1,024-dim, multilingual, task LoRA)
Embedding (fallback)	Google Gemini text-embedding-004 (output_dim = 1,024)
LLM Re-ranker	DeepSeek-Chat (T = 0, JSON index array output)
KB Curator	DeepSeek-Chat (keep/improve/delete, batch = 10)
Web Crawler	Firecrawl API (async, markdown output, max 500 pages)
Edge Functions	Deno / TypeScript on Supabase
Frontend	React + TypeScript + Vite + Tailwind CSS + shadcn/ui
Scheduling	pg_cron (Monday: re-index; Tuesday: curate)

6. Limitations and Future Work

Source weights were determined via grid search on three pilot deployments; they are not yet learned per-client through online optimization. The LLM re-ranking step introduces latency at high query throughput; lightweight bi-encoder re-rankers (Zhu et al., 2023) could replace it. The A/B framework does not ablate individual system components. Future work should include controlled ablations of temporal scoring, source weighting, and the feedback loop in isolation.

The 74–81% citation rate range covers seven deployments, all SEO/GEO-oriented commercial sites. The relatively lower rate (74%) observed in the medical domain (Abs Korkalış) suggests that strict medical terminology and conservative LLM safety guardrails against health-related hallucinations may require domain-specific tuning. Broader evaluation across diverse verticals and LLM systems is required to establish generalizability.

Planned extensions: multilingual embedding optimization, real-time WebSocket KB updates, federated RAG across multi-tenant shared knowledge pools, and more robust GEO evaluation metrics aligned with BEIR benchmark methodology.

7. Utility Model Application (Turkey)

The RAG Signal Adaptive RAG architecture is being evaluated for registration as a Utility Model (Faydalı Model) under TÜRKPATENT regulations (Türk Patent ve Marka Kurumu, 2024). Under Decree Law No. 551, utility models protect novel technical solutions that solve industrial problems. The six novel technical contributions are:

- Differential source weight matrix: A 5×1 reliability coefficient vector assigning empirically derived weights to heterogeneous knowledge sources.
- Temporal freshness scoring function: The $1/(1+\Delta t/30)$ rational decay formula automatically applied at retrieval time.
- Task-aware query augmentation lexicon: Generation-task-specific suffix vectors for five distinct generation contexts.
- Hybrid mathematical–LLM re-ranking pipeline: Sequential analytical pre-ranking and LLM re-ranking with graceful degradation fallback.
- Autonomous KB Curator Agent: An AI agent that autonomously identifies, improves, and removes low-quality KB chunks on a weekly schedule.
- Closed-loop approval feedback integration: Automated indexing of user-approved content as the highest-weight knowledge source.

8. Conclusion

This paper presented the RAG Signal Adaptive RAG architecture, a production-deployed system validated across seven client deployments spanning six industry verticals that substantially improves context quality in LLM-based content generation by overcoming the four fundamental limitations of classic RAG. Multi-source differential weighting, temporal freshness scoring via a rational decay function, task-aware query augmentation, hybrid re-ranking, autonomous KB curation, and closed-loop feedback collectively yield a mean LLM citation rate of 78% (range: 74–81%)—approximately double the rate without KB context—while reducing character count by approximately 83% through intelligent compression.

As Generative Engine Optimization becomes a critical discipline in the LLM-search era, the proposed architecture provides a scalable, self-improving framework for organizations seeking measurable visibility in AI-generated content. The system constitutes a novel technical contribution eligible for utility model protection under Turkish patent law.

References

Aggarwal, A., Mündler, N., & West, R. (2024). GEO: Generative engine optimization. In Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. https://doi.org/10.48550/arXiv.2311.09735

Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2024). Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In Proceedings of ICLR 2024. https://doi.org/10.48550/arXiv.2310.11511

Bommasani, R., Hudson, D. A., Adeli, E., et al. (2021). On the opportunities and risks of foundation models. arXiv. https://doi.org/10.48550/arXiv.2108.07258

Brynjolfsson, E., Li, D., & Raymond, L. R. (2023). Generative AI at work (NBER Working Paper No. 31161). https://doi.org/10.3386/w31161

DeepSeek AI. (2024). DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. arXiv. https://doi.org/10.48550/arXiv.2405.04434

Gao, Y., Xiong, Y., Gao, X., et al. (2024). Retrieval-augmented generation for large language models: A survey. arXiv. https://doi.org/10.48550/arXiv.2312.10997

Google DeepMind. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv. https://doi.org/10.48550/arXiv.2403.05530

Günther, M., Mohr, I., Williams, D. J., Wang, X., & Xiao, H. (2024). jina-embeddings-v3: Multilingual embeddings with task LoRA. arXiv. https://doi.org/10.48550/arXiv.2409.10173

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. In Proceedings of NAACL 2024. https://doi.org/10.48550/arXiv.2403.14403

Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems, 33, 9459–9474.

Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE TPAMI, 42(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

Nogueira, R., & Cho, K. (2019). Passage re-ranking with BERT. arXiv. https://doi.org/10.48550/arXiv.1901.04085

pgvector Project. (2023). Open-source vector similarity search for Postgres [Software]. GitHub. https://github.com/pgvector/pgvector

Shi, W., Min, S., Yasunaga, M., et al. (2023). REPLUG: Retrieval-augmented black-box language models. In Proceedings of NAACL 2023. https://doi.org/10.48550/arXiv.2301.12652

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. NeurIPS 2021 Datasets and Benchmarks Track.

Türk Patent ve Marka Kurumu. (2024). Faydalı model başvuru kılavuzu. TÜRKPATENT. https://www.turkpatent.gov.tr

Wang, Z., Zhang, Y., Han, X., Liu, Z., & Sun, M. (2024). A comprehensive study of knowledge editing for large language models. arXiv. https://doi.org/10.48550/arXiv.2401.01286

Xu, S., Pang, L., Shen, H., Cheng, X., & Chua, T.-S. (2024). Knowledge conflicts for LLMs: A survey. arXiv. https://doi.org/10.48550/arXiv.2403.08319

Zhu, Y., Yuan, H., Wang, S., et al. (2023). Large language models for information retrieval: A survey. arXiv. https://doi.org/10.48550/arXiv.2308.07107