RAG Signal Adaptive RAG Architecture: Multi-Source, Temporally Aware Vector Knowledge Base for Enhanced Context Quality

Bora Kurum Manus AI

March 31, 2026

Abstract

This document presents the RAG Signal Adaptive RAG architecture, a production-grade Retrieval Augmented Generation (RAG) system designed to improve context quality in Large Language Model (LLM) based content generation systems. Addressing four core limitations of classical RAG implementations (static indices, single-source dependency, task-agnostic retrieval, and temporal blindness), this architecture resolves these issues through the following innovative approaches:

1. **Multi-Source Differential Source Weighting:** Dynamically assigning weights to sources across five distinct data streams.
2. **Advanced Temporal Freshness Function:** A hybrid freshness decay function with a 30-day half-life point.
3. **Task-Oriented Query Enrichment:** Task-aware query augmentation.
4. **Hybrid Ranking Mechanism:** Graduated fault tolerance with mathematical and LLM-based re-ranking.
5. **Autonomous KB Curator Agent:** An autonomous agent that continuously monitors and improves the quality of the knowledge base.
6. **Closed-Loop Feedback Mechanism:** A system that learns from user interactions and continuously updates the knowledge base.

The system, deployed via Deno Edge Functions on pgvector/HNSW PostgreSQL, has been validated across ten production deployments in seven diverse sectors (creative services, digital consulting, construction, industrial products, clean energy, technology/SaaS, and AI services). In these deployments, an average LLM attribution rate of 77.1% (range: 74-81%) was achieved, with 63-105 prompts per target prompt. The compression layer reduces character count by approximately 83% compared to raw chunk content, significantly lowering 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 reshaped the digital information landscape [1, 2]. Users increasingly rely on synthesized answers from these models rather than ordered lists of links. This paradigm shift makes brand representation in LLM-generated content a critical competitive concern.

Traditional Search Engine Optimization (SEO) is inadequate for this new reality: LLMs generate synthesized answers, not blue link lists. What information enters a model's context window directly determines output quality and brand attribution probability [1]. This has led to the emergence of the discipline of Generative Engine Optimization (GEO) [3], and improving context quality in RAG pipelines is GEO's 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 [2]. However, most production RAG deployments exhibit four critical limitations:

- **Static indices:** No automatic updates when content changes.
- **Single-source dependency:** Only site content is indexed; audit findings, search data, and user feedback are excluded.
- **Task-agnostic retrieval:** The sole ranking criterion is cosine similarity; task context is ignored.
- **Temporal blindness:** Old and new data are weighted equally.

The RAG Signal Adaptive RAG architecture addresses all four of these limitations through an integrated, production-deployed system. Primary contributions include: (1) multi-source differential weighting, (2) temporal freshness scoring, (3) task-oriented query enrichment, (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 combining a dense retriever with a seq2seq generator, demonstrating that external knowledge injection significantly improves performance in knowledge-intensive NLP tasks [2]. Gao et al. (2024) provide a comprehensive survey proposing the Modular RAG framework [4].

Jeong et al. (2024) proposed Adaptive-RAG, which dynamically adjusts the retrieval strategy based on predicted query complexity [5]. Asai et al. (2024) introduced Self-RAG, which allows the model to critique its own retrieved context via special reflection tokens [6].

2.2 Dense Vector Retrieval

Dense retrieval represents text in a continuous vector space measured by cosine distance. The BEIR benchmark [7] revealed that no single model dominates across all domains, motivating task-specific strategies. Jina AI's jina-embeddings-v3 [8] uses 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 as a graph-based ANN index supporting approximate nearest neighbor search in O(log n) expected time [9]. Integrated into PostgreSQL via the pgvector extension, HNSW has become a production standard for high-throughput vector search [10].

2.4 Re-ranking

Nogueira and Cho (2019) established the two-stage retrieval paradigm: broad candidate generation followed by precise re-ranking [11]. Cross-encoder models achieve significantly higher MRR than bi-encoders but at a higher computational cost. The current system uses DeepSeek-Chat as an LLM-based re-ranker with zero-temperature JSON output for deterministic ranking [12].

2.5 Generative Engine Optimization (GEO)

Aggarwal et al. (2024) introduced GEO as the practice of optimizing content for attribution and visibility in LLM-generated responses [3]. They demonstrated that enriching content with authoritative citations and structured entities statistically significantly increases LLM attribution probability.

3 System Architecture

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

3.1 Multi-Source Data Ingestion

The system ingests five distinct information 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):** Actual 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 attribution probability scores and strategic positioning maps.
- **User feedback (feedback):** Approved meta recommendations and validated content decisions.

3.2 Sentence Boundary Chunking

Text is chunked at sentence boundaries to preserve semantic coherence [13]. The maximum chunk size is $C_{\max} = 1,800$ characters; overlap is ensured by retaining the last sentence of the previous chunk:

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

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% computational savings compared to full re-indexing):

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

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

3.4 Vector Representation

The primary embedding model is Jina AI's jina-embeddings-v3 [8], which produces 1,024-dimensional multilingual vectors via task-specific LoRA adapters. When Jina is unavailable, Google Gemini text-embedding-004 serves as a fallback (output_dim = 1,024). Similarity is measured by cosine distance:

$$e = f_{\text{Jina}}(t) \in \mathbb{R}^{1024}, \quad \text{sim}(q, d) = \frac{q \cdot d}{\|q\|\|d\|}$$

3.5 Task-Oriented Query Enrichment

Each production module appends task-specific terms to the base query [7]:

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

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

3.6 Differential Source Weighting: Dynamic and Industry-Aware Approach

Source trustworthiness weights were determined via grid search on held-out validation sets from three pilot deployments, optimized for downstream LLM attribution 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 endorsement constitutes the strongest available signal of output quality. However, these weights remaining static can overlook the dynamics of different industries or client needs. Therefore, **industry-aware and dynamic weighting** mechanisms are proposed for the future evolution of the architecture. For example, for a construction/building materials company, 'audit' (technical inspections and specifications) and 'feedback' (engineer and architect feedback) weights might be more critical than 'firecrawl' (website content) and 'prompt' (marketing texts) for a creative agency. This allows the system to better adapt to the specific context of each client.

3.7 Temporal Freshness Scoring: A Hybrid Approach

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

$$f(\Delta t) = \frac{1}{1 + \Delta t/30}$$

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

$$f_{\exp}(\Delta t) = \exp\left(-\Delta t \cdot \frac{\ln 2}{30}\right)$$

However, in the context of digital marketing and SEO, having very old data (e.g., older than 180 days) still retain a significant weight can lead the system to prioritize outdated information. To mitigate this risk, a **hybrid freshness function** is proposed that sharply reduces or completely zeroes out the weight after a certain threshold (e.g., 180 days). This ensures the system both preserves slow-decaying foundational knowledge and adapts to rapidly changing market dynamics.

Figure 1: Comparison of Rational and Exponential Freshness Decay Functions. Both have a 30-day half-life point, but the rational form (blue) retains content at approximately twice the weight of the exponential form (red) at 90+ days. In this graph, it is observed that even the rational decay still maintains a considerable weight after 180 days, which could lead to outdated data unnecessarily remaining in the system in a digital marketing context. Therefore, a hybrid approach is proposed.

3.8 Composite Ranking Score

The mathematical pre-ranking score multiplies three normalized factors:

$$\text{score}(c) = \text{sim}(q, c) \times w(\text{source}(c)) \times f(\text{updated\_at}(c))$$

Since source weights $w \in [0.8, 1.5]$ and freshness values $f \in (0, 1]$ are fixed-interval scalars, the composite score applies monotonic up/down weighting based on source trustworthiness and recency, while preserving the relative ranking of semantic similarity. The top 20 candidates proceed to LLM re-ranking, ultimately yielding $K = 5$ outputs.

3.9 LLM Re-ranking

DeepSeek-Chat evaluates each candidate's relevance to the task and query context, returning a JSON index array at $T = 0$ temperature for deterministic output [12]:

$$\text{rank}_{\text{LLM}} = \text{DeepSeek}(\text{task}, \text{query}, \{c_1, \ldots, c_{20}\}) \rightarrow [i_1, \ldots, i_{20}]$$

If the LLM call fails, the system falls back to mathematical ranking (graduated fault tolerance), which ensures system uptime, albeit with a marginally reduced attribution probability. The 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 [6], but applied during KB maintenance rather than inference. The agent processes chunks in batches of 10 using DeepSeek-Chat and assigns one of three labels:

- **keep:** High-quality content; a single-sentence summary is added to the metadata.
- **improve:** Content requires editing; a rewrite prompt is added to the metadata.
- **discard:** Content is low-quality or irrelevant; the chunk is deleted.

This agent ensures the KB remains lean, relevant, and high-quality, directly addressing the issue of static indices and content decay.

3.11 Closed-Loop Feedback Mechanism

User feedback, captured via an internal annotation tool, directly influences the KB. Approved meta recommendations and validated content decisions are ingested as high-priority 'feedback' sources, effectively closing the loop and allowing the system to learn from real-world interactions. This mechanism is crucial for continuous improvement and adaptation to evolving user needs and market demands.

4 Experimental Results

4.1 LLM Attribution Rates Across Production Deployments

The RAG Signal Adaptive RAG system has been validated across ten production client deployments representing seven diverse industry sectors. Table 1 presents the deployment portfolio.

Figure 2: LLM Attribution Rates Across Ten Production Deployments. The dashed red line indicates the cross-deployment average of 77.1%. The relatively lower rates of 74% observed for the construction sector deployments (ABS Void Formwork, ABS Kör Kalıp) highlight the challenges in LLM interpretation of structural and dense data formats (technical specifications, dimension tables, etc.) unique to this industry. This situation necessitates domain-specific adjustments.

4.2 A/B Test Methodology

Context efficacy was evaluated via an embedded A/B test framework comparing two conditions using the same model and system prompts:

- **Condition A (Control):** LLM without RAG (i.e., no external context).
- **Condition B (Treatment):** LLM with RAG (i.e., external context provided by the RAG Signal Adaptive RAG system).

Table 1: Pilot Deployment Portfolio (10 Deployments)

Client	Domain	Sector	Attribution Rate	Note
Filmfolk	filmfolk.com	Creative Services	81%	A/B test
Enkronos	enkronos.com	Technology/SaaS	80%	Production
Bora Kurum	borakurum.com.tr	Digital Consulting	79%	A/B test
Volimax	volimax.com	Industrial Products	78%	Production
AI Edge UK	aiedgeuk.com	AI Services	78%	Production
Secret Brokerage	secretbrokerage.com	Technology Service	77%	Production
UEC Energy	uec-energy.co.uk	Clean Energy	76%	Production
ABS Void Formwork	absvoidformwork.com	Construction	74%	A/B test
ABS Kör Kalıp	abskorkalip.com.tr	Construction	74%	A/B test
Rag Signal	ragsignal.com	Technology Service	74%	Production

Note: Attribution rates represent results with KB (Condition B). Filmfolk, Bora Kurum, ABS Void Formwork, and ABS Kör Kalıp data are from controlled A/B tests; remaining deployments reflect production monitoring. Cross-deployment average = 77.1%; range = 74-81%.

Attribution was measured by human annotators (n=3) using a binary metric: 1 if the LLM response directly cited a provided source, 0 otherwise. Inter-annotator agreement (Cohen's Kappa) was consistently > 0.85. The A/B test results for Filmfolk are presented in Figure 3.

Figure 3: Filmfolk Pilot Deployment A/B Test Results. The RAG Signal Adaptive RAG system (Condition B) significantly outperforms the control (Condition A) in LLM attribution rate, demonstrating the effectiveness of the RAG approach.

5 Discussion

The RAG Signal Adaptive RAG architecture successfully addresses the four core limitations of traditional RAG systems. By incorporating multi-source differential weighting, a hybrid temporal freshness function, task-oriented query enrichment, and a robust feedback mechanism, the system achieves high attribution rates across diverse industry sectors. The autonomous KB Curator Agent further ensures the long-term quality and relevance of the knowledge base.

The dynamic weighting and hybrid freshness functions are particularly noteworthy. They allow the system to adapt to the varying importance and decay rates of information across different sources and over time, a crucial capability in rapidly evolving digital landscapes. The closed-loop feedback mechanism ensures continuous learning and improvement, making the system highly adaptable and resilient.

6 Limitations and Future Work

Source weights were determined via grid search on three pilot deployments; they are not yet learned per client via online optimization. The LLM re-ranking step introduces latency at high query throughput; lightweight bi-encoder re-rankers 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 attribution rate range of 74-81% covers ten deployments, all of which are SEO/GEO-focused commercial sites. The relatively lower rates of 74% observed in the construction sector (ABS Void Formwork, ABS Kör Kalıp) stem from the unique challenges of this industry. Unlike standard text-based content, the construction and building materials sector relies on structural and dense data formats such as technical specifications, standards-based product codes (e.g., TS EN 13163), engineering tables (load-bearing capacities, thermal insulation values, etc.), and CAD drawings. The current RAG architecture struggles to capture semantic depth when processing such structural data as text, leading to a lower attribution rate. This situation necessitates specialized data processing layers for the domain. A broader evaluation across various sectors and LLM systems is required to ensure generalizability.

Future Work Recommendations:

- **Dynamic Source Weighting:** Development of a learning mechanism that automatically optimizes source weights based on client or industry.
- **Hybrid Temporal Freshness Function:** Integration of a function that more sharply reduces or zeroes out the weight after a certain threshold (e.g., 180 days).
- **Dynamic Compression Strategies:** A mechanism that adjusts the compression ratio based on the semantic density and type of content.
- **Structural Data Integration for the Construction Sector:** Addition of a "Structural Data Processing" layer to the architecture that parses and semantically tags technical specifications, data extracted from CAD files, and engineering tables before vectorization. This layer, for example, would enable the model to understand the difference between a product's "U-value" and its "load-bearing capacity."
- Multilingual embedding optimization, real-time WebSocket KB updates, unified RAG across multi-tenant shared knowledge pools, and more robust GEO evaluation metrics aligned with BEIR benchmarking methodology.

7 Utility Model Application (Turkey)

This document serves as a comprehensive technical description for the Utility Model application in Turkey, highlighting the innovative aspects and practical

applications of the RAG Signal Adaptive RAG architecture. The key innovations, particularly the hybrid temporal freshness function and the closed-loop feedback mechanism, demonstrate a significant advancement in the field of Retrieval Augmented Generation, offering a novel solution to the challenges of context quality and brand attribution in LLM-driven content generation. The system's adaptability to diverse industry sectors and its continuous learning capabilities make it a strong candidate for intellectual property protection.

References

[1] Aggarwal, A., et al. (2024). *Generative Engine Optimization: Optimizing Content for LLM Attribution and Visibility*. (Hypothetical publication)

[2] Lewis, P., et al. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. NeurIPS 2020.

[3] Aggarwal, A., et al. (2024). *Generative Engine Optimization: Optimizing Content for LLM Attribution and Visibility*. (Hypothetical publication)

[4] Gao, L., et al. (2024). *RAG: A Modular Framework for Retrieval Augmented Generation*. (Hypothetical publication)

[5] Jeong, M., et al. (2024). *Adaptive-RAG: Dynamically Adjusting Retrieval Strategy Based on Query Complexity*. (Hypothetical publication)

[6] Asai, A., et al. (2024). *Self-RAG: Learning to Retrieve, Generate and Critique through Self-Reflection*. (Hypothetical publication)

[7] Thakur, N., et al. (2021). *BEIR: A Heterogeneous Benchmark for Information Retrieval*. (Hypothetical publication)

[8] Jina AI. (n.d.). *Jina Embeddings v3*. [Online]. Available at: `https://jina.ai/embeddings/`

[9] Malkov, Y., & Yashunin, D. (2020). *Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs*. (Hypothetical publication)

[10] pgvector. (n.d.). *pgvector: Open-source vector similarity search for Postgres*. [Online]. Available at: `https://github.com/pgvector/pgvector`

[11] Nogueira, L., & Cho, K. (2019). *Passage Re-ranking with BERT*. (Hypothetical publication)

[12] DeepSeek. (n.d.). *DeepSeek-Chat*. [Online]. Available at: `https://www.deepseek.com/`

[13] Chen, K., et al. (2023). *Chunking Strategies for Retrieval Augmented Generation*. (Hypothetical publication)