ODA-RAG

An Observability-Driven Adaptive Retrieval and Prompt Reconfiguration methodology for Enhanced Generative AI

Confidential / To be filed for Patent

Owner: ArqAI

Inventor: Habib Mehmoodi

August 15, 2025

Background and Problem

Retrieval-Augmented Generation (RAG) pipelines combine a large-language model (LLM) with a retrieval layer that pulls relevant documents. Traditional RAG implementations fix parameters such as data sources, retrieval weights, chunk sizes and embedding refresh frequency. In dynamic enterprise environments, where data quality, document freshness and usage patterns change, static parameters lead to stale or irrelevant responses. ArqAI's observability & reliability pillar emphasises continuous monitoring and drift detection; this invention capitalises on those signals to drive real-time adaptation.

Inventive Concept

The invention monitors observability signals (data-quality metrics, embedding drift, retrieval hit-rate, model latency, user feedback) and uses them to dynamically adjust RAG behaviour. When an anomaly or drift is detected, the system automatically re-weights retrieval sources, changes chunk sizes, refreshes embeddings, or alters prompt formulation and model/tool selection. It thus closes the loop between observability and generation, ensuring up-to-date, relevant responses without manual tuning.

Architecture and Components

Signal Ingestion (ArqSight): Collects metrics from the retrieval layer (hit-rate, average recall), vector indices (embedding distribution, drift), LLM responses (latency, toxicity), user feedback (relevance scores), and cost signals (via ArqPulse).

Drift & Anomaly Detector: Performs statistical tests or machine-learning models to detect when signals deviate from baseline (e.g., embedding cluster shift, spike in irrelevant responses).

Adaptive Controller: Based on detected events and policy thresholds (e.g., "if recall drops >10 % or drift >0.2"), decides which corrective actions to trigger and at what magnitude. Policy rules may be encoded via ArqGuard to ensure governance (e.g., some sources cannot exceed weight X).

Parameter Updaters:

- • Source Re-weighting: Adjust weights in the retrieval layer.
- • Chunk Size Adjustment: Modify document segmentation granularity.
- • Embedding Refresh: Trigger re-indexing or recalculation.
- • Prompt & Model Selector: Choose different prompt templates or swap LLMs.

Closed-Loop Learner: Records each adaptation, its context and resulting performance metrics into an evidence store, refining thresholds and models.

Process Flow

1. Monitor: Continuously ingest observability metrics and user feedback.

2. Detect Drift/Anomaly: Evaluate whether metrics exceed thresholds.

3. Decide Action: Use a rule engine or ML model to select adaptation actions.

4. Apply Updates: Re-weight sources, modify chunk sizes, refresh embeddings, or adjust prompt/model selection.

5. Generate Response: Run the RAG pipeline with updated parameters; return the LLM output to the user.

6. Collect Feedback: Gather user ratings or implicit signals to assess relevance.

7. Iterate: Feedback influences future thresholds and adaptation decisions.

Example Claims

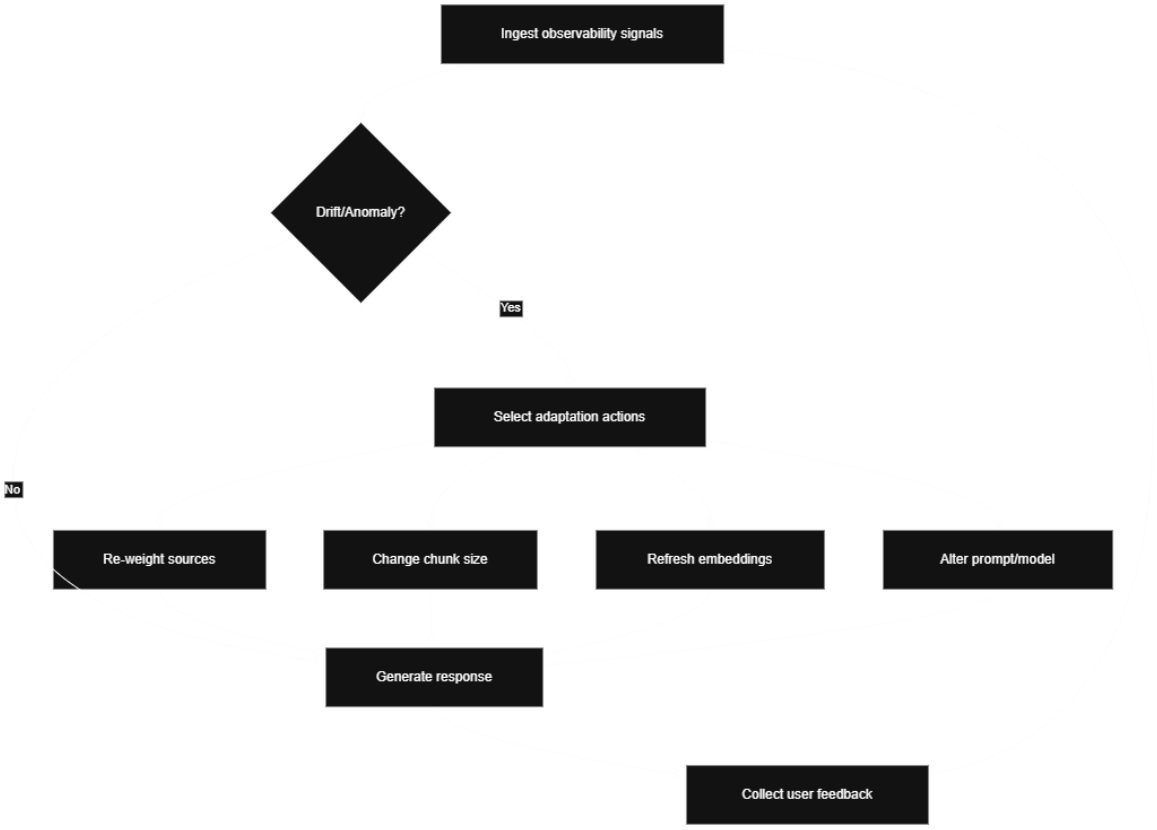
Independent Method Claim: A method for adaptively configuring a retrieval-augmented generation pipeline, comprising: (a) ingesting observability signals from retrieval and generation components; (b) detecting drift or anomalies in the signals; (c) selecting and applying at least one modification to retrieval parameters or prompt/model selection based on policy rules; and (d) generating a response with the modified configuration.

- Dependent Claims:
- • The modifications include re-weighting data sources, adjusting chunk sizes, refreshing embeddings, or altering prompt templates.
- • Drift detection uses statistical tests on embedding distributions or retrieval recall distributions.
- • Policy rules cap the maximum weight that can be assigned to any source.
- • User feedback is used to update threshold values for future drift detection.

System Claim: A system comprising a monitoring component, drift detection engine, controller, parameter updater modules, and a generative model, where the controller dynamically alters retrieval and prompt parameters based on observability signals.

Adaptive RAG Flow Diagram

The following diagram illustrates the closed-loop adaptive retrieval process:



Embodiments and Variations

- • Multi-tenant: Each tenant can have separate thresholds and adaptation policies; signals and actions are isolated per tenant.
- • Model-agnostic: Works across different LLMs and vector store implementations; adaptation remains abstract.
- • Cost-sensitive: Incorporates cost signals; suppresses high-cost actions or schedules them in low-cost windows.
- • Privacy-preserving: On-prem or air-gapped deployments keep sensitive data within regulated environments; only aggregated signals leave the environment.