GuardRail AI: Engineering a Cloud-Native, Zero-Trust RAG Pipeline for Enterprise PII Redaction

Naman
Lead Systems Architect
Live Application: guardrail-ai.vercel.app
Independent Research / Engineering Portfolio
Email: nl2833@nyu.edu

***Abstract*—The integration of Large Language Models (LLMs) into enterprise workflows is frequently obstructed by data privacy constraints. This paper presents GuardRail AI, a serverless, multi-cloud system designed to perform Retrieval-Augmented Generation (RAG) while ensuring zero-leakage of Personally Identifiable Information (PII). By utilizing a decoupled architecture consisting of deep-learning-based redaction (BERT), high-speed inference (Groq), and vector-optimized storage (pgvector), the system achieves secure document querying with sub-second latency. We analyze the architectural trade-offs necessitated by cloud compute constraints and document the engineering solutions implemented to resolve memory-bound and network-bound bottlenecks.**

***Index Terms*—Retrieval-Augmented Generation (RAG), PII Redaction, Cloud Computing, Natural Language Processing, Vector Databases.**

I. Introduction

As organizations transition toward AI-driven document analysis, the risk of exposing sensitive data—such as Social Security Numbers (SSNs) and financial records—to third-party LLM providers has increased. GuardRail AI addresses this by intercepting the data pipeline to perform automated sanitization before vectorization or generation occurs.

II. System Architecture

The system utilizes a distributed microservices approach to ensure scalability and separation of concerns.

A. Frontend and Backend Interoperability

The user interface is deployed on Vercel (Next.js), providing edge-rendering capabilities. The core logic resides in an asynchronous FastAPI (Python) environment, enabling concurrent processing of high-I/O document parsing tasks.

B. Storage and Vectorization

Rather than utilizing standalone vector stores, the system leverages Neon DB (Serverless PostgreSQL) with the *pgvector* extension. This allows for unified relational and vector data management, simplifying the schema for 384-dimensional embeddings generated via Sentence-Transformers.

C. Inference Layer

To optimize performance, inference is bifurcated:

- **LLM & Transcription:** Groq's LPU architecture is used for Llama-3 and Whisper to minimize response latency.
- **Redaction:** Hugging Face's serverless inference endpoints host the BERT-based Named Entity Recognition (NER) models.

III. Engineering Challenges and Solutions

A. Memory Optimization and OOM Resolution

During initial deployment, the inclusion of local PyTorch libraries exceeded the 512MB RAM threshold of the cloud container. The architecture was pivoted to a "Thin-Client" model, stripping heavy libraries and offloading compute to external REST APIs, reducing the container footprint by 90%.

B. Mathematical Safety in Vector Operations

System crashes were observed during semantic search when zero-vector embeddings (resulting from deprecated API responses) were processed. We implemented a robust similarity function to handle null-norm vectors:

$$\text{Sim}(A, B) = \begin{cases} \frac{\sum A_i B_i}{\sqrt{\sum A_i^2}\sqrt{\sum B_i^2}}, & \text{if } \|A\|, \|B\| \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

C. Cross-Origin Resource Sharing (CORS) Conflict

A "split-brain" configuration error, where a secondary FastAPI instance overwrote middleware settings, initially blocked Google OAuth preflight requests. Consolidating the application factory and dynamically injecting allowed origins resolved the authentication block.

IV. Performance Analysis and Findings

A. OCR Latency in Constrained Environments

Analysis revealed that Optical Character Recognition (OCR) via Tesseract on a single vCPU incurred a 3.5-minute processing time for 900KB documents. This necessitates the use of asynchronous task queues (Celery/Redis) to prevent request timeouts in production.

B. Heuristic Fallbacks for Distorted Text

To mitigate PII leakage caused by OCR-induced whitespace distortions (e.g., "u s e r @ d o m a i n . c o m"), we developed "OCR-Proof" Regex patterns. These act as a secondary safety net to the BERT NER model, ensuring high recall in noisy data environments.

V. Conclusion and Future Work

GuardRail AI demonstrates that enterprise-grade privacy is achievable in serverless environments. Future iterations will focus on migrating document parsing to AWS Lambda for better parallelization and integrating vision-capable LLMs to eliminate traditional OCR bottlenecks.