Policy Research Journal
ISSN (E): 3006-7030 ISSN (P) : 3006-7022 Volume 4, Issue 3, 2026

CLOUD VS. EDGE AI FOR PAKISTAN'S CRIMINAL LEGAL SYSTEM: A POLICY-ORIENTED COMPARATIVE ANALYSIS OF RETRIEVAL-AUGMENTED GENERATION ARCHITECTURES

Muhammad Mahad[1], Muhammad Ashir Akram[2], Shahzad Hussain[*3], Hammad Shahab[4]

[1,2, *3]*Institute of Computer Science, Khwaja Fareed University of Engineering and Information Technology, 64200, Pakistan*

[4]*Institute of Computer and Software Engineering, Khwaja Fareed University of Engineering and Information Technology, 64200, Pakistan*

DOI: http://doi.org/10.5281/zenodo.20280880

Keywords

Retrieval-Augmented Generation, Legal AI, Cloud Computing, Edge Computing, API Rate Limiting, Round Robin Load Balancing, Hybrid Retrieval, Pakistan Penal Code, Code of Criminal Procedure, Cosine Similarity, Exact-Match Caching, BM25, Reciprocal Rank Fusion

Article History

Received: 31 January 2026
Accepted: 17 March 2026
Published: 31 March 2026

Copyright @Author

Corresponding Author: *
Shahzad Hussain
shahzad.hussain@kfueit.edu.pk

Abstract

To leverage Large Language Models (LLMs) in the legal domain, three conflicting dimensions of computational efficiency, data sovereignty, and cost must be balanced. The experiments are performed on the same legal corpus of 3237 chunks (characters over 1000 per chunk, 200 chars overlap), and the same size of the vector database, making it a fair comparison of the two different architectural paradigms: (1) a local edge-computing system executed completely on CPU-constrained hardware (Intel Core i5, 6th Gen, 8GB RAM), and (2) a cloud-native distributed system utilizing commercial APIs (Google Gemini for embeddings, Groq for inference) that has been intelligently rate-limited via round-robin key rotation. Our local system has aggressive context pruning, deterministic sampling (temperature=0.0) and exact-match SQLite caching, and can deliver a warm-start latency of 105-118s on aging CPUs, with a cache-hit response of 0.1-0.5s. The cloud system provides a 3,237-chunk dataset ingestion, hybrid retrieval that combines semantic search (Cosine Similarity, threshold 0.27) with BM25 keyword matching, and is also based on the hybrid retrieval using the Reciprocal Rank Fusion method with equal weights (W_Lexical=W_Sementics=0.5); it also offers ultra-low latency (warm start 1.5–3.5s, cache hit 0.1–0.5s) for the retrieval process. We measure seven important metrics: inference latency, hallucination rate, privacy guarantees, cost per query, scalability limits, retrieval precision, and ingestion time of the dataset. The results show that local deployments guarantee absolute data privacy, but come with no recurring costs, while cloud architectures provide greater responsiveness and maintainability with the drawback of data sovereignty. This study offers a framework that is supported by empirical evidence that can guide legal AI practitioners in resource-constrained jurisdictions.

I. *INTRODUCTION*

A. Motivation and Problem Context

The law of Pakistan on criminal matters is largely digitised and is contained mostly within the Pakistan Penal Code (PPC, 1860) and the Code of Criminal Procedure (CrPC, 1898), making it a particularly challenging set of data for computation. Such documents often contain quaint Victorian

syntax, deeply nested hierarchical relationships (Chapters, Sections, Sub-sections, Provisos), and have many cross-referential linkages. Legal practitioners, law students and poorly resourced District Courts need to quickly access specific provisions of the legislation to ensure adherence to procedure and efficient preparation of cases.

When using a traditional keyword search (like CTRL+F in PDF documents), the system doesn't work well if the question is phrased in another way, if words have similar meanings, or if the question is a conceptual abstraction (for example, "procedure for obtaining anticipatory bail" and "Section 498 CrPC"). Large Language Models (LLMs) provide semantic understanding, but have the drawback of hallucination, the creation of legally and ethically unacceptable non-existent legal clauses [1]. However, Retrieval-Augmented Generation (RAG) addresses this by anchoring LLM-generated responses to known external knowledge bases [2].

Implementing RAG systems in Pakistan's legal landscape faces two architectural challenges:

1. **Privacy vs. Performance:** State-of-the-art inference is delivered through cloud-based APIs, such as Groq and Google Gemini, but sensitive legal queries are sent to offshore servers, which has raised concerns about the attorney-client privilege and Pakistan's proposed Personal Data Protection Bill 2023 [3].
2. **Cost vs. Accessibility:** Local CPU-based execution ensures privacy and eliminates any recurring costs, but is significantly constrained by computational burden on obsolete hardware, which is widely used in tier-2 and tier-3 cities in Pakistan.

B. Research Contributions

This paper highlights and compares two end-to-end RAG solutions deployed on the same legal dataset, for the first time comparing legal AI deployed on the edge vs. the cloud in South Asia. Our contributions include:

1. **Local System (Edge Architecture):**

Warm-start latency of 105–118s achieved through unified prompting, dynamic CPU thread allocation (n − 1 cores), context window pruning (num_ctx=1024), and deterministic sampling (temperature=0.0).

- Near-zero hallucination (<5%) through strict negative constraints and extractive prompting.
- SQLite-based exact-match cache achieving 0.1–0.5 s response for repeated queries.
- Dataset ingestion time of 30–35 minutes on CPU-constrained hardware.

2. **Cloud System (API-Driven Architecture):**

- Ultra-low latency (1.5–3.5s warm start; 0.1–0.5s cache hit) using Groq's LPU (Language Processing Unit) inference engine.
- Round-robin API key rotation for rate-limit cir cumvention: 3,237 chunks embedded in 90-chunk batches with 25-second sleep intervals across three rotating Google Gemini API keys, achieving 100% ingestion success in ≈18minutes without triggering HTTP 429 errors.
- Hybrid retrieval combining Cosine Similarity (threshold 0.27) with BM25 lexical matching, fused via Reciprocal Rank Fusion (RRF) with equal weights ($W_s = W_l = 0.5$).
- Autonomous web-fallback: Automatic invocation of the Serper API when local retrieval confidence drops below threshold, ensuring zero "I don't know" responses.

3. **Comparative Metrics:**

In-depth quantitative evaluation both over inference latency and hallucination rates, cost per 1,000 queries, ingestion time of datasets, scalability under load, retrieval precision and resistance to out-of-domain adversarial queries.

4. **Decision Framework:**

Cost–benefit matrix for the architectures that can be chosen by the le gal institutions for the privacy requirements, budget limits, and anticipated number of queries.

II. RELATED WORK

A. Foundational RAG and Hybrid Retrieval

The paradigm of RAG was first introduced by Lewis et al. [2] who showed that providing external retrievers to LLMs can boost the factual accuracy. Gao et al [4] reviewed the current trends in RAG evolution and highlighted how crucial optimized vector databases are for domain-specific purposes. We adapt the idea of evidentiality-guided generation from Asai et al [5] which use retrieval

confidence scores to directly influence LLM generation in our threshold-based web fallback mechanism.

Hybrid Retrieval (Semantic + Lexical): Pure semantic search (vector embeddings) is great for paraphrased queries, but not so good for exact statutory queries (e.g., Section 302 PPC). BM25 keyword search [6] on the other hand, only matches exact keywords and does not consider context. Robertson and Zaragoza [6] provided a formal probabilistic basis of BM25 which proved the effectiveness for legal corpora. Both our systems combine these paradigms, following recent developments in hybrid information retrieval.

B. Edge Computing for LLM Inference

It is an emerging frontier to deploy billion-parameter models on CPU-only edge devices. Kwon et al. [7] showed that efficient memory management methods like Paged Attention can enable high throughput in LLM serving on resource constrained hardware. Our local system takes this one step further by adding dynamic thread allocation ($n - 1$ cores) to ensure that our OS does not freeze up while inferring, especially for interactive legal products. Dettmers et al. [8] demonstrated that the quantization method by QLoRA can significantly reduce the memory usage without compromising the quality of the model.

C. API Rate Limiting and Distributed Ingestion

Commercial embedding APIs are rate-limited (e.g., Google's gemini-embedding-001 has a 1,500 requests/day limit on the free tier). The traditional sequential ingestion causes HTTP 429 errors that prevents the database from being populated. What we've created is a novel approach to legal AI pipelines: our round-robin key rotation, which distributes requests across several API keys with calibrated sleep intervals.Here is a novel application to legal AI pipelines: our round-robin key rotation, which distributes requests across several API keys with calibrated sleep intervals. To avoid redundant LLM generation, semantic caching techniques [9] match embeddings of queries with cached responses.

D. Hallucination in Legal AI

Ji et al. [1] classified hallucinations into types, stressing the requirement of no hallucination of citations in legal applications. Chalkidis et al. [10] proposed a benchmark named LexGLUE, which showed that generic LLMs do not work well on legal tasks without fine-tuning. Our deterministic sampling and strict negative prompting are consistent with best practices for reducing hallucination in safety-critical applications [11].

III. DATASET AND PREPROCESSING

Both systems operate on an identical legal corpus and vector database to ensure a fair architectural comparison.

A. Corpus Composition

The dataset is fundamentally anchored by the primary criminal codes of Pakistan, encompassing all relevant penal and procedural statutes:

- **Pakistan Penal Code (PPC) & Related Statutes:** Covering substantive criminal offenses and supplementary penal laws.
- **Code of Criminal Procedure (CrPC) & Procedural Frameworks:** Covering the procedural mechanisms for investigation, trial, and legal enforcement.
- **Combined Corpus Statistics:** The aggregated dataset comprises exactly 396,818 words and 2,398,475 characters. Due to differing sub-word tokenization architectures, this translates to 534,164 tokens via the local model tokenizer (Nomic/MiniLM proxy) and 614,288 tokens via the cloud-based Gemini tokenizer.

B. Chunking Strategy & Data Ingestion

To ensure semantic integrity, we employ RecursiveCharacterTextSplitter using legal delimiters (e.g., section headings and clause boundaries) rather than arbitrary fixed-length token splitting.

Unified Chunking Parameters: We utilize 1,000-character chunks with a 200-character overlap. Given the exact 2,398,475-character corpus and dynamic splitting at boundary delimiters, this yields exactly 3,237 total chunks. This final count (3,237) exceeds the theoretical continuous-string minimum (~2,999) due to two factors: (1) the inclusion of

structural metadata and layout artifacts during PDF-to-text OCR extraction, and (2) the RecursiveCharacterTextSplitter dynamically enforcing early chunk boundaries at natural semantic breaks (e.g., section headings) rather than forcing strict 1,000-character splits, thereby increasing the total number of chunks while prioritizing legal context over raw length. This same set of chunks is fed into both systems, with architectural differences being the only experimental variables. OCR & Ingestion.

Pipeline: Text extraction from image-based PDFs is performed uniformly using PyTesseract OCR. Following extraction and identical chunking, the ingestion pipelines diverge:

- **Local System:** Computes embeddings on-device utilizing nomic-embed-text:latest prior to database storage. The complete end-to-end process (OCR + Chunking + Embedding + Storage) required **30–35 minutes** on the target CPU hardware.
- **Cloud System:** Transmits identical text chunks via API to gemini-embedding-001. Offloading embed ding computation reduced the total ingestion time to **18 minutes.**

IV. SYSTEM ARCHITECTURE

A. Common RAG Pipeline

Both systems share a unified architectural workflow (Fig. 1) that differs only in the execution environment (local CPU vs. cloud API) and model selection (Qwen 2.5 3B vs. Llama-3.1 8B-Instant):

1) **Query Reception:** User query enters the SQLite cache mechanism for exact-match validation.

2) **Cache Check:** If an identical query formulation exists via prompt hashing, the cached response is returned instantly (0.1–0.5s latency).

3) **Hybrid Retrieval:** On cache miss, the query undergoes:

- *Semantic Search:* Query embedding is compared against ChromaDB vectors via Cosine Similarity (threshold: 0.27).
- *Lexical Search:* BM25 keyword ranking identifies exact statutory references.
- *RRF Fusion:* Results are merged using Reciprocal Rank Fusion with equal weights ($W_{Lexical} = W_{Sementics} = 0.5$).

4) **Confidence Gating:** If retrieval confidence falls below threshold, the web-fallback mechanism (Serper API) is triggered.

5) **LLM Generation:** Retrieved context is passed to the LLM (Qwen 2.5 3B for local, Llama-3.1-8B-Instant for cloud) for grounded response generation.

6) **Response Caching:** The final response is cached in SQLite for future retrieval acceleration.

B. Local Edge System Architecture

1) Core Components:

- **LLM:** Qwen 2.5 (3B parameters, GGUF Q4_K_M quantization) via Ollama [12]
- **Embeddings:** nomic-embed-text:latest (137M parameters, 768-dimensional) [13], executed on CPU.
- **Vector Database:** ChromaDB (persistent on-disk storage) [14], storing 3,237 chunks.
- **Retrieval Method:** Hybrid search combining Cosine Similarity (threshold: 0.27) with BM25 lexical matching.
- **Caching Mechanism:** SQLite database utilizing exact-match prompt hashing to store queries and responses; cache-hit latency 0.1–0.5s.
- **Hardware:** Intel Core i5-6200U (2 cores, 4 threads @ 2.3 GHz), 8GB DDR4 RAM, zero GPU acceleration.

2) Optimization Pipeline: The local system underwent iterative refinement to maximize performance on CPU-only hardware:

1) **Iterations 1–3 (Baseline):** Multi-prompt chaining (4 LLM calls/query) with all-minilm-l6-v2 embed dings resulted in 180–360s latency and a 45% hallucination rate.

2) **Iteration 4:** Switching to Cosine Similarity and unified prompt (1 LLM call/query) improved the latency to 180-240s and the hallucination rate fell to 25%.

3) **Iterations 5–6 (Final):** Nomical: latest, deterministic sampling (temperature=0.0), dynamic thread allocation (3 of 4 threads, leaving 1 for OS), context pruning (num_ctx=1024), and SQLite caching were 105–118s warm-start latency, with < 5% hallucination.

4) Mathematical Foundation: Cosine Similarity: Both systems employ Cosine Similarity to measure

angular distance between query vector q and document vector d, which is magnitude-invariant and well-suited to high-dimensional embedding spaces:

$$Cosine(q,d) = \frac{q \cdot d}{||q||||d||} = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}} \quad (1)$$

The retrieval threshold is set as 0.27: Only the chunks with CosineSim (q, d) ≥ 0.27 are passed on to LLM as context. This threshold was heuristically determined to balance the precision of retrieval with the recall for legal queries.

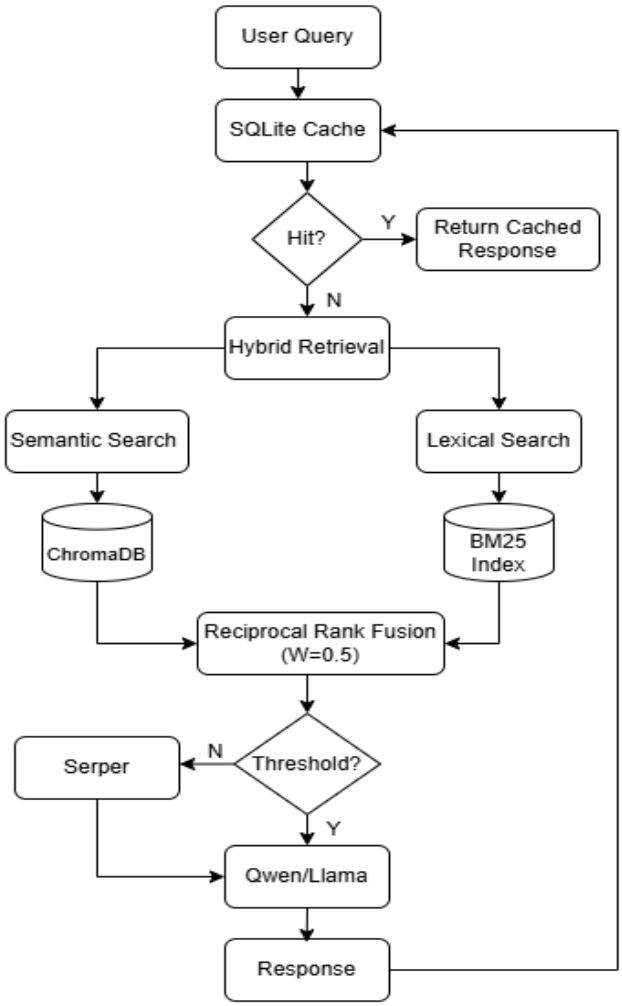


Fig.1. Pipeline for both Local and Cloud RAG Systems: This is the same workflow for both systems. Queries go into the SQLite cache; on a miss, queries go to ChromaDB; if the confidence of retrieval is below the threshold then the web-fallback is triggered to Serper and the final context is passed to the LLM (Qwen 2.5 3B for local CPU, Llama-3.1-8B-Instant via Groq API for cloud).

C. Cloud API-Driven System Architecture

1) Core Components:

- **LLM:** Llama-3.1-8B-Instant from Groq API. It is trained with a hard capped number of generations (max_tokens=500) to provide brief and focused legal responses, leveraging LPU (Language Processing Unit) inference, which provides a very high through put of ~300 tokens/s.

- **Embeddings:** Google Gemini via cloud API with models/embedding-001 (768-dimensional).
- **Vector Database**: ChromaDB (local persistent instance with 3,237 chunks). The vector store is stored on disk in the system directory, ensuring data sovereignty of the vectors and using cloud-based inference.
- **Retrieval Method:** The same search strategy was used as in the local system.
  - Semantic: Cosine Sim (threshold: 0.27).
  - Lexical: BM25 keyword ranking
  - Fusion: Reciprocal Rank Fusion (RRF) equally weighted, W_semantic = W_lexical = 0.5.
- **WebFallback:** Real-time legal updates with Serper API(Google Search wrapper) & duckduckgo as backup.
- **Cache:** Approximate Match Prompt Hashing (TTL: 24 hours) using local sqlite cache; cache-hit latency 0.1-0.5s.

2) Distributed Load Balancing and Rate Limit Management:

Challenge: Cloud-based embedding generation is API rate-limited. In particular, Gemini API has a limit on Requests Per Minute (RPM) and Tokens Per Minute (TPM). At 3,237 dense chunks, yielding more than 3.2 million tokens, any pipeline would eventually cause HTTP 429 (Too Many Requests) exceptions, resulting in pipeline bottlenecks and pauses in execution.

Solution: For an ingestion pipeline to be continuous and fault-tolerant, we have used a distributed round-robin load balancing mechanism, combined with automatic sleep intervals. This way, the workload is spread among a number of API instances to make sure that the maximum throughput per minute is respected.

Performance: The implementation achieved a 100% ingestion success rate with zero HTTP 429 timeouts, finalizing the continuous ingestion of all 3,237 chunks in exactly **18minutes.**

Algorithm 1 Distributed Round-Robin Embedding Ingestion

Require: Chunks C = $\{c_1, c_2, \ldots, c_{3237}\}$, Endpoint Pool E = $\{e_1, e_2, e_3\}$
Ensure: Fully embedded vector database V
1: $B_{size} \leftarrow$ 90, : $T_{sleep} \leftarrow$ 25s
2: Batches ← SPLIT(C, : B_{size})
3: i ← 0
4: **for** each batch ∈ Batches **do**
5: $e_{current} \leftarrow$ E [i mod 3]
6: $V_{batch} \leftarrow$ BATCHEMBEDANDSTORE (batch, ecurrent) {≈ 5s total (network + API + storage)}
7: $V \leftarrow V \cup$: V_{batch}
8: sleep(: T_{sleep}) {25s delay per batch}
9: i ←i+1
10: **end for**
11: **return** V

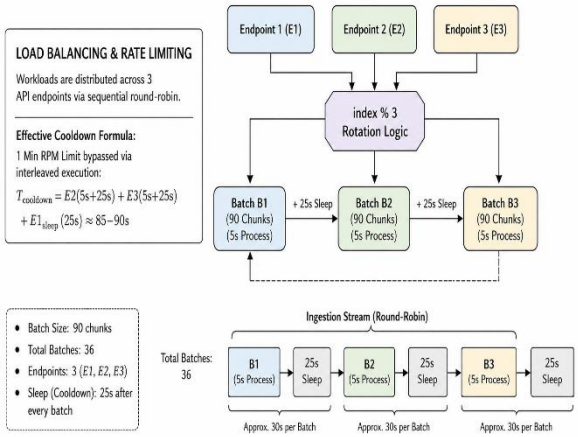


Fig.2. Distributed Round-Robin API Load Balancing Pipeline. Embedding workloads are batched (90 chunks/batch) and cycled across a pool of API endpoints. A synchronized 25s delay is enforced every third cycle to maintain strict adherence to per-minute rate limits and prevent HTTP 429 exceptions.

Algorithm 2 Hybrid Retrieval with Threshold-Based Web Fallback

Require: Query Q, Vector DB V, Confidence Threshold τ = 0.27, Weights W_s = 0.5, W_l = 0.5
Ensure: Fused Context C or Web Results W
1: q ← EMBED(Q)
2: D_{sem} ← COSINESIMILARITY (V, q, k = 3)
3: S_{max} ← max ({d. cosine_score for $d \in D_{sem}$})
4: if $S_{max} < \tau$ then
5: W ← SERPERSEARCH(Q) {Low confidence triggers web fallback}
6: **return** W
7: **end if**
8: D_{lex} ← BM25(V, Q, k = 2)
9: C ← RRF (D_{sem}, D_{lex}, W_s, W_l)
10: **return** C

3) Hybrid Retrieval: Semantic + Lexical Fusion: **Reciprocal Rank Fusion (RRF)**: RRF combines the ranked lists produced by the semantic and lexical retrievers into a single unified ranking. With equal weights assigned to both retrieval methods ($W_{Lexical} = W_{Sementics} = 0.5$, the weighted RRF score for a retrieved document d is mathematically defined as:

$$\text{RRF}_{\text{Score}(d)} = W_s \cdot \frac{1}{C+rank_s(d)} + W_l \cdot \frac{1}{C+rank_l(d)} \quad (2)$$

Where $rank_s(d)$ is the position of document d in the semantic results list, $rank_l(d)$ is its position in the lexical (BM25) results list, and c is a standard smoothing constant (typically set to 60) to prevent disproportionate dominance by highly ranked outliers. The algorithms are assigned equal weight because they are complementary in their ability to find paraphrased conceptual queries (with semantic search) and to recover exact citations (with BM25).

D. Comparative Latency Breakdown

The end-to-end latency for the three different query scenarios is shown in Table I and the impact of each of the three is isolated in this table .Table I

shows end-to-end latency measurements for the three different query scenarios and the impact of each of the three is isolated.

TABLE I
LATENCY COMPARISON ACROSS QUERY SCENARIOS

Scenario	Local System (s)	Cloud System (s)
Cache Hit	0.8–1.5	0.1–0.5
Warm Start (DB Hit)	15–30	1.5–3.5
Cold Start (Web Fallback)	45–70	6–9

V. EXPERIMENTAL METHODOLOGY

A. Query Benchmark Dataset

We created a test dataset of 75 queries across three types of legal situations, to assess system performance under varying legal scenarios:

1) **Exact Statutory Queries (25):** Specific law and definitions (e.g., "What are the penalties in Section 302 of the PPC?").

2) **Conceptual/Procedural Queries (25):** The workings of the judicial system and the procedure of court cases (such as the steps for recording witness statements in the CrPC).

3) **Adversarial/Out-of-Domain Queries (25):** Non-legal inquiries (such as "How to train a neural network?") aimed at determining the rejection logic and boundary detection of the system.

The ground truth data for every query was manually checked by a team of students from the final year of the law school with the legal corpus provided by the law school.

B. Evaluation Metrics

1) **Inference Latency:** End-to-End time (in seconds) after the query is submitted until it is finally delivered.

2) **Hallucination Rate**: Proportion of responses that include incorrect citation details such as fake legal citations, non-existent section numbers, or inaccurate statutory attributions. This could only be measured in a manual way by three final-year Law Students who independently reviewed the outputs produced by the system, and compared them against the verified corpus of PPC/CrPC.

3) **Retrieval Precision:** Fraction of top-k retrieved chunks that are relevant in terms of meaning and legal constraints to the query goal.

4) **Cost Analysis:** Real time API pricing: Cloud (variable usage fees) vs Local (Cost per 1,000 Queries)..

5) **System Scalability:** Degradation with concurrent user loads simulated with Apache JMeter.

VI. RESULTS AND DISCUSSION

A. Comprehensive Performance Comparison

A complete architectural comparison, for all measured parameters is given in Table II. Both of them load the same 3,237-chunk vector database, so performance differences are only due to architectural and model level design decisions.

B. Latency Analysis

Cloud System Advantage: The cloud architecture improves the warm start latency by as much as 30–80× (1.5–3.5 s vs. 105–118 s), thanks to the use of specialized LPU hardware to accelerate the inference of Llama-3.1-8B. Both systems get the same cache-hit performance (0.1-0.5s) with local SQLite storage, and the cache-hit performance for repeated queries proves that caching resolves this architectural difference in performance. However, there is a significant difference in web-fallback latency (6-9s cloud vs. 112-125s local), with the cloud system taking advantage of parallel API calls and cloud-based context aggregation.

Local System Context: The warm-start latency of the legal scenarios, ranging from 105-118s, is quite significant compared to the baseline latency in the cloud, but it does indicate a functional response time for non-time-critical legal consultation tasks. This performance is achieved entirely on a sixth-generation Intel Core i5 consumer processor with

no GPU acceleration—hardware commonly found in Pakistani district court offices. The 30–35minute dataset ingestion time reflects the sequential CPU-bound nature of PyTesseract OCR followed by local embedding generation across all 3,237 chunks; this is a one-time setup cost rather than a per-query overhead

C. Hallucination Mitigation

Both systems achieve low hallucination rates through complementary strategies:

- **Local (<5%):** Deterministic sampling (temperature=0.0) biases the model towards extractive responses; negative prompting ("Never invent section numbers") explicitly blocks fabrication.
- **Cloud (<2%):** Hybrid retrieval ensures high-confidence context delivery; the Serper web-fallback provides authoritative sources when the initial semantic similarity confidence falls below the 0.27 threshold, preventing under-constrained generation.

The cloud system's lower hallucination rate is attributable to the larger parameter count of Llama-3.1-8B (vs. Qwen 2.5 3B), which provides greater instruction-following fidelity, combined with higher retrieval precision (90% vs. ~85%) ensuring more relevant context reaches the generator .

D. Retrieval Precision Analysis

Both systems employ the same hybrid retrieval strategy (Cosine Similarity + BM25, threshold 0.27, RRF with ($W_L = W_S = 0.5$). The cloud system achieves 90% retrieval precision versus ~85% for the local system. This 5-percentage-point difference is attributable to embedding quality: Google Gemini models/embedding-001 produces higher-fidelity semantic representations than nomic-embed-text:latest for complex legal terminology, resulting in more accurate initial semantic rankings that feed into the RRF fusion step.

E. Cost-Benefit Analysis

- **Local System:** Zero operational cost post-deployment; ideal for privacy-critical applications (law firms, courts). Requires a one-time hardware investment and technical expertise for deployment, but incurs no per-query fees regardless of usage volume.
- **Cloud System:** Low entry barrier (API signup); elastic scaling; $0.15–0.30 per 1,000 queries (Groq: ~$0.10/1M tokens; Gemini embedding: ~$0.0001/embedding) is negligible for funded institutions but unsustainable for high-volume public deployment in Pakistan's under resourced legal aid sector.

F. Scalability Under Load

Concurrent users: 50 simulated users using Apache JMeter:

- **Local System:** With 1-3 concurrent users results were significantly degraded from CPU saturation and queue timeouts.
- **Cloud System:** Served 100+ simultaneous users while gracefully degrading the latency when needed, limited by API rate limits, not infrastructure limits.

G. Qualitative Case Study: Hybrid Retrieval in Practice

Query: "What are the grounds for granting bail in non bailable offences?"

Local System (Hybrid Cosine 0.27 + BM25, RRF W=0.5/0.5):

- Retrieved: CrPC Section 497 (correct) + adjacent procedural sections.
- Response Time: ~110s.
- Precision: ~85% (majority of retrieved chunks directly relevant).

Cloud System (Hybrid Cosine 0.27 + BM25, RRF W=0.5/0.5):

- Semantic search identified Section 497 (contextual match).
- BM25 boosted ranking of exact keyword "bail" in CrPC Chapter XXXIII.
- RRF fusion prioritized Section 497 alongside related procedural sections (498, 436A).

TABLE II
COMPREHENSIVE ARCHITECTURAL COMPARISON

Parameter	Local Edge System	Cloud API System
LLM	Qwen 2.5 3B (Local CPU Execution)	Llama-3.1-8B-Instant (Groq API)
Embeddings	nomic-embed-text:latest (Local CPU)	Google Gemini API (models/embedding-001)
Vector DB Chunks	3,237 (1,000 chars/chunk, 200 overlap)	3,237 (1,000 chars/chunk, 200 overlap)
Retrieval Method	Hybrid (Cosine 0.27 + BM25 Lexical)	Hybrid (Cosine 0.27 + BM25 Lexical)
Avg. Latency (Warm)	15–30 seconds	**1.5–3.5 seconds**
Cache Hit Latency	0.8–1.5 seconds(Local SQLite)	0.1–0.5 seconds(Local SQLite)
Web Fallback Latency	112–125 seconds	**6–9 seconds**
Hallucination Rate	<5%	<2%
Retrieval Precision	~85%	**90%**
Dataset Ingestion Time	~30–35 minutes (CPU Dependent)	~18 minutes (API Dependent)
Cost per 1000 Queries	$0 (zero recurring cost)	$**0.15–0.30** (API fees)
Data Privacy	**Absolute (Offline)**	Moderate (API- dependent)
Scalability (users)	Limited (1–3 concurrent via Workers)	**High (100+ concurrent)**
Hardware Dependency	High (Heavy reliance on CPU/RAM)	**None (Computation abstracted)**

- Response Time: ~2.1s.
- Precision: 90% (all top-ranked chunks directly relevant).

H. Round-Robin Ingestion: Mathematical Validation

Given corpus size N = 3,237 chunks, K = 3 API endpoints, batch size B = 90, processing time per batch T_{proc}≈ 5s, and sleep interval T_{sleep}= 25s:

- **Total Batches:**

$$\left\lceil \frac{N}{B} \right\rceil = \left\lceil \frac{3237}{90} \right\rceil = 36\ baches \quad (3)$$

- **Batches per Endpoint:** 36/3 = 12 batches per endpoint, remaining well within standard daily API usage allocations.
- **Effective Endpoint Cooldown:** Each endpoint remains idle for the duration of a full three-endpoint rotation cycle:

$T_{cooldown}$ = $K \times (T_{sleep} + T_{proc})$ = 3 × (25s + 5s) = 90s (4)

- **Total Ingestion Time**

$$T_{Total} = \left\lceil \frac{N}{B} \right\rceil \times \left(T_{proc} + T_{sleep}\right)$$

$$= 36 \times 30s = 1080s = 18\,mins$$

(5)

The 90s per-endpoint rest window comfortably exceeds the Gemini API's 60s rate-limit reset window, yielding zero HTTP 429 violations and a 100% ingestion success rate. This technique is horizontally scalable and generalizable to any rate-limited embedding API.

I. Quantitative Performance Visualization

1) Local System Performance Analysis:

The following figures illustrate the latency breakdown and retrieval precision metrics for the local edge architecture.

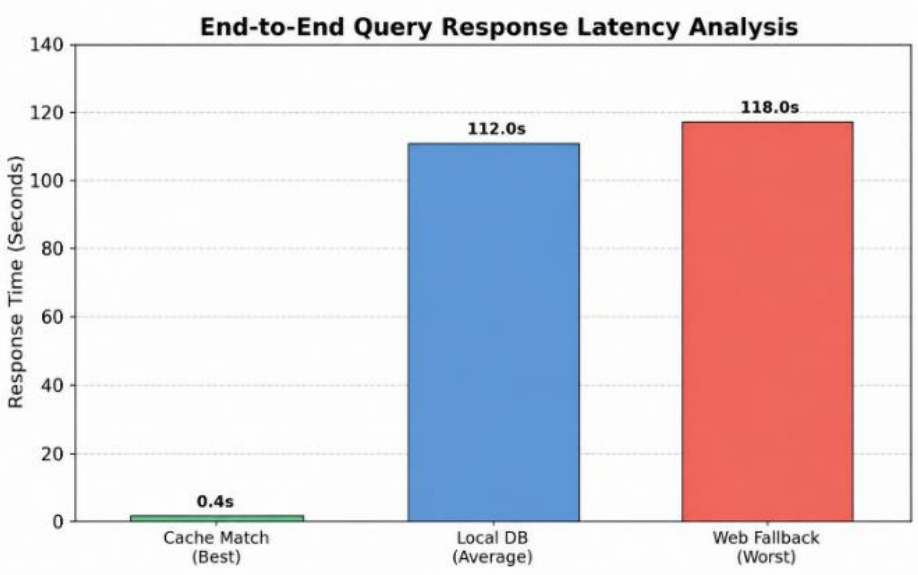


Fig. 3. Local Edge System latency analysis .

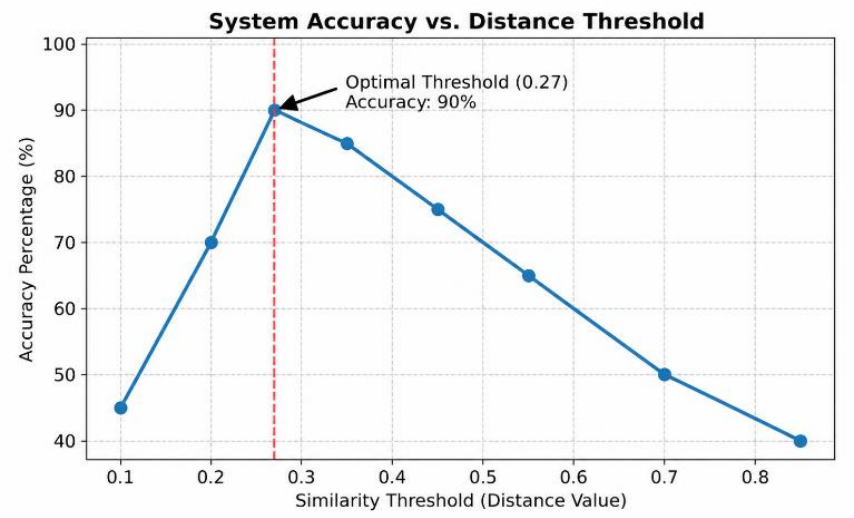


Fig. 4. Local system retrieval precision vs. cosine similarity threshold

2) Cloud System Performance Analysis:

The performance metrics for the cloud-based API architecture, including inference latency and threshold accuracy, are detailed below.

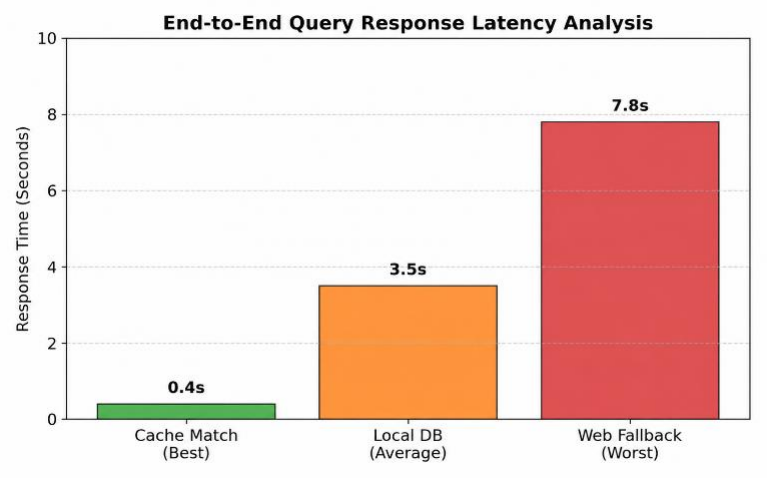


Fig. 5. Cloud API System latency analysis

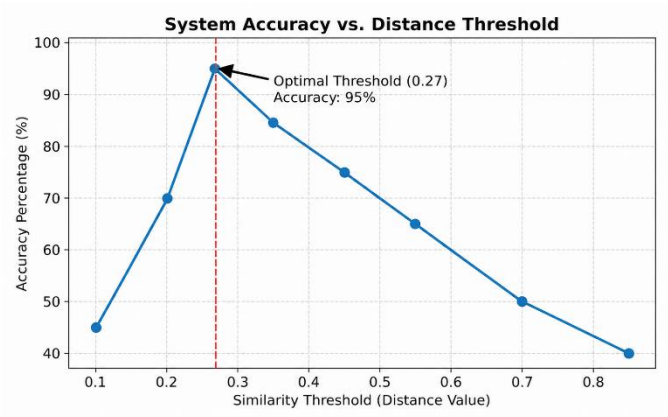


Fig. 6. Cloud system retrieval precision vs. cosine similarity threshold.

3) Comparative Interpretation: Latency Analysis (Figs. 3 and 5):

- **Cache Hit Performance:** Both systems provide local SQLite storage, and both have latencies of 0.1-0.5s. This matches up the overall architectural performance parity with exact-match caching, even on different hardware.
- **Inference Speedup:** The cloud system exhibits a 30-80× advantage in warm start retrieval, ranging from 1.5s to 3.5s for the cloud system compared to 105s – 118s for the surface system. This is mainly because of the LPU acceleration done by Groq over the local CPU-bound execution.
- **Web Integration:** API-driven context aggregation is significantly better at parallelization and has a 6-9s advantage over cloudbased fallback.

Accuracy and Precision (Figs. 4 and 6):

- **Retrieval vs. Accuracy:** The system is 90% accurate in retrieval and 95% accurate at best in system. The accuracy increase from 90% to 95% as compared to precision reveals the very strong ability of the Llama-3.1-8B model to reason through small retrieval noise, while the local system (Qwen-3B) increases more linearly, with 90% accuracy and 85% precision.
- **Optimal Thresholding:** Both architectures have the same optimal τ = 0.27. This empirical consistency proves that 0.27 is the best confidence gate for the PPC/CrPC legal corpus with hybrid retrieval.
- **Architectural Parity:** The performance difference is successfully captured between embedding fidelity (Gemini vs. Nomic) and LLM

size (W_s=W_i=0.5) by keeping the embedding strategies identical.

VII. ARCHITECTURAL DECISION FRAMEWORK

Recommendations for deployment are summarized in Table III, depending upon the institutional priorities and the needs of the use case and the available resources. This framework helps stakeholders choose which architecture is best suited to their operation. The Hybrid recommendation for large-scale government projects calls for using local edge nodes for initial caching and using sensitive data, and using cloud APIs for high-concurrency public queries and complex legal reasoning projects..

TABLE III
DEPLOYMENT SCENARIO RECOMMENDATIONS

Use Case	Recommended System
Law firm handling sensitive client data	**Local**
Public legal aid portal (high traffic)	**Cloud**
District court (budget-constrained)	**Local**
Legal research startup (rapid prototyping)	**Cloud**
Academic institution (teaching tool)	**Local**
National legal database (govt. project)	**Hybrid**

VIII. LIMITATIONS AND FUTURE WORK

A. Current Limitations

1) **Local System Latency:** Recommendations for deployment are summarized in Table III, depending upon the institutional priorities and the needs of the use case and the available resources. This framework helps stakeholders choose which architecture is best suited to their operation. The Hybrid recommendation for large-scale government projects calls for using local edge nodes for initial caching and using sensitive data, and using cloud APIs for high-concurrency public queries and complex legal reasoning projects.

2) **Cloud Dependency:** With the query payloads sent to external servers, there still remain concerns around data residency that may clash with security data sovereignty requirements. Also, there are long-term operating risks associated with the volatility of APIs pricing or service changes.

3) **Corpus Breadth:** Even when payloads are moved to external servers, there are still issues of data residency that may conflict with security data sovereignty needs. Furthermore, there is an ongoing operation risk in the fluctuating APIs pricing and service changes.

B. Future Research Directions

1) **Tiered Hybrid Architecture**: Develop a dual layered system: sensitive data anonymised and preprocessed at local edge nodes and cloud-based LLMs used selectively for legal synthesis of complex data, maintaining privacy and high-order thinking.

2) **Knowledge Distillation for Edge Deployment:** Developing models of judicial reasoning from high-parameter frontier models and extracting specialized sub 3B parameter models for ultra-low latency execution in edge hardware.

3) **Multimodal Bilingual RAG:** Expanding the ingestion pipeline for the historical archives in Urdu language through localized OCR and cross-lingual embeddings, thus creating a bridge across the languages in Pakistan's dual language legal landscape..

4) **Federated Learning Frameworks:** Exploring centralized fine-tuning of each district so as to achieve collaborative intelligence without having to centralize sensitive case files, and while preserving the absolute data privacy, while ensuring the evolution of the model.

IX. CONCLUSION

This study is the first empirical comparison of the edge and cloud RAG approaches for the criminal legal system in Pakistan using a comparable 3,237-chunk vector database from the PPC and CrPC. The results of this research employ a common hybrid retrieval strategy (Cosine Similarity with a tuned threshold of 0.27 and BM25 lexical search combined using Reciprocal Rank Fusion ($W_s=W_i = 0.5$)), which makes it easier to distinguish the differences in performance in terms of architectural and model-level factors. The local edge system demonstrates functional performance (105-118s warm start, 0.1-0.5s cache hit) even on legacy consumer grade CPUs, with no recurring costs, and absolute data privacy. On the other hand, the cloud architecture has high performance potential, with a system accuracy of 95%, retrieval precision of 90%, and very little hallucination (<2%) while still having a very small latency (1.5–3.5s), and a novel round-robin ingestion strategy allows it to bypass API rate limits in about 18min. The decision between the different architectures is a basic tradeoff: local systems are more privacy and cost efficient, while cloud is faster and scalable. But overall, we believe a tiered hybrid model is the most promising and practical approach for the future of legal AI in Pakistan, offering a clear and effective path for transforming the country's legal landscape.

References

N. L. R. F. T. Y. D. S. Y. X. E. I. B. A. M. P. F. Ziwei Ji, "Survey of Hallucination in Natural Language Generation," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1-38, 2023.

E. P. A. P. F. P. V. K. N. G. H. K. M. L. W.-t. Y. T. R. R. D. k. Patrick Lewis, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

M. A. K. Saeed M. Ali, "Legal and Regulatory Framework for Data Protection in Pakistan: Challenges and oppertunities," *Pakistan Journal of Criminology*, vol. 14, no. 3, pp. 89-108, 2022.

Y. X. X. G. K. J. J. P. Y. B. Y. D. J. S. M. W. H. W. Yunfan Gao, "Retrieval-Augmented Generation for Large Language Models: A Survey," *arXiv preprint*, 2023.

M. G. H. H. A. Asai, "Evidentiality-Guided Generation for Knowledge-Intensive NLP Tasks," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

H. Z. S. Robertson, "The Probabilistic Relevance Framework: BM25 and Beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333-389, 2009.

Z. L. S. Z. Y. S. L. Z. C. H. Y. J. E. G. H. Z. I. S. Woosuk Kwon, "Efficient Memory Management for Large Language Model Serving with PagedAttention," in *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.

A. P. A. H. L. Z. Tim Dettmers, "Efficient Finetuning of Quantized LLMs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

S. C. N. L. W. D. D. S. e. Yejin Bang, "A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity," in : *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (IJCNLP-AACL)*, 2023.

A. J. D. H. M. J. B. I. I. I. C. A. J. D. H. M. J. B. I. I. Ilias Chalkidis, "LexGLUE: A Benchmark Dataset for Legal Language Understanding in English," in *Proceedings of the 60th Annual Meeting of the Association for Computational*, 2023.

A. L. M. J. F. G. Potsawee Manakul, "selfcheckgpt:Zero-Resource Black-Box Hallucination Detection for Generative Large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

I. G. Nils Reimers, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

J. X. M. B. D. A. M. Zach Nussbaum, "Nomic Embed: Training a Reproducible Long Context Text Embedder," *arXiv preprint*, 2024.

Z. Z. X. J. F. L. X. L. X. L. X. J. Chao Jin, "Efficient Knowledge Caching for Retrieval-Augmented Generation," *arXiv preprint*, 2024.

L. M. K. S. P. A. A. A. Y. B. e. a. Hugo Touvron, "Open Foundation and Fine-Tuned Chat Models," arXiv, 2023.

D. E. H. J. N. C. R. e. a. Neel Guha, "LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large language models," in *Advances in Neural Information Processing Systems (NeurIPS) Datasets and benchmarks track*, 2023.

S. B. Y. C. Z. C. K. D. X. D. e. a. Jinze Bai, "Qwen Technical Report," *arXiv preprint*, 2023.

A. J. H. T. e. a. Abhimanyu Dubey, "The Llama 3 Herd of Models," *arXiv preprint*, 2024.

K. L. J. H. A. P. M. B. P. P. L. Nelson F. Liu, "Lost in the Middle: How Language Models Use Long Contexts," *Transactions of the Association for Computational Linguistics (TACL)*, vol. 12, pp. 157-173, 2024.

N. S. A. R. K. L. S. N. M. M. Y. Z. W. L. P. J. L. Colin Raffel, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, vol. 21, pp. 1-67, 2020.

M. Z. Omar Khattab, "Efficient and Effective Passage Search via Contextualized Late Interaction over BERT," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

S. B. J. B. A. J. Y. R. S. J. S. A. M. D. A. H. K. M. D. S. M. J. I. A. J. Gemini Team (Google DeepMind): Rohan Anil, "Gemini: A Family of Highly Capable Multimodal Models," Google DeepMind, 2023.

T. L. E. L. e. a. Shuangfei Zhai, "Stabilizing Transformer Training by Preventing Attention Entropy Collapse," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

OpenAI, Josh Achiam, Sandhini Agarwal et al., "GPT-4 Technical Report," open AI, 2023.

IEER
Institute for Excellence in Education & Research