Framework Comparison for Legal RAG Systems: LangChain vs LlamaIndex

Systematic empirical comparison of LangChain and LlamaIndex on the CUAD-RAG legal benchmark

Presented by: Triyasha Ghosh Dastidar (tg2936)

Motivation & Problem Definition

Why RAG for legal NLP

- Domain-specific knowledge and factual grounding for contracts
- Reduces hallucinations; improves answer fidelity
- Handles complex terminology and long contracts

Why framework orchestration matters

- Retrieval effectiveness (recall@k) drives context quality
- Tool retrieval/invocation and agent reasoning affect reliability
- Latency/overhead trade-offs impact deployability

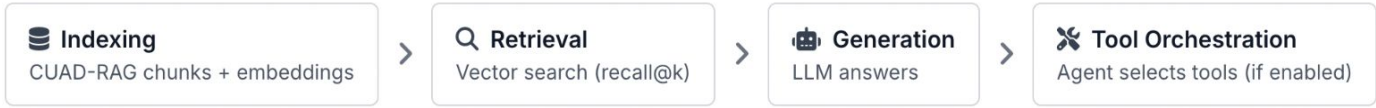
Current gaps in evaluations

- Lack of systematic framework-level comparisons
- Metrics often overlook tool accuracy and orchestration overhead
- No unified, reproducible pipeline integrating retrieval, generation, tools

Problem statement

How do LangChain and LlamaIndex compare for legal RAG under identical datasets, retrievers, LLMs, embeddings, and evaluation protocols?

System Overview (3A): Architectures & Models



Architectures evaluated

- Simple RAG: retrieval + single-step generation
- Tool-use Agent RAG: agent controller invokes specialized tools

Goal

Isolate framework orchestration behavior under identical models, embeddings, prompts, and evaluation protocols.

Models & embeddings

GPT-3.5-turbo
Used for Simple RAG (temperature = 0)

GPT-4o-mini
Used for tool-use agents (temperature = 0)

text-embedding-3-large
Embeddings for chunk vectors (shared across frameworks)

Note: Prompts, retrievers, and model settings held constant to ensure comparability.

Dataset & Indexing (3B): CUAD → CUAD-RAG

CUAD → CUAD-RAG transformation

CUAD
41 clause categories

→

Queries
Natural-language questions

→

Gold spans
Mapped to contract text

- One query per contract; multiple gold spans retained
- Ambiguous/multipart contracts filtered to enforce one-to-one mapping

Chunking strategy

500 tokens | 50 overlap

- LangChain: **RecursiveCharacterTextSplitter**
- LlamaIndex: **SentenceSplitter**

Vector stores & similarity

- LangChain → **FAISS** (cosine similarity)
- LlamaIndex → **native vector store** (cosine similarity)
- Shared embeddings: **text-embedding-3-large**

Test set & document characteristics

- 1,000 queries (standardized test set)
- Contracts range **5,000–50,000 tokens**
- Many queries require multi-location answer synthesis

Orchestration & Evaluation (3C)

Frameworks compared

LangChain

Modular chain-based design; LCEL

LlamaIndex

Sophisticated retrieval & agent orchestration

Agent tools

- **extract_dates** Temporal extraction
- **rag_qa** Retrieval-based QA

Retrieval depth sweep

k = 10 | k = 20 | k = 100

Evaluation framework

Answer quality

- F1, BLEU
- ROUGE-1/2/L
- BARTScore (semantic)

Retrieval

- Recall@k

Efficiency

- Latency (avg per query)
- Tool efficiency (expected vs invoked)

Tool usage

- Call counts & patterns

Experiment tracking

Weights & Biases (per-query and aggregate logs, execution traces)

Framework Comparison: LangChain vs LlamaIndex

Context: legal RAG on CUAD-derived benchmark ($k \in \{10, 20, 100\}$; tools: rag_qa, extract_dates)

Aspect	LangChain	LlamaIndex
Architecture	Chain/Agent/Tool primitives; prompt-first orchestration Flexible composition, rich integrations	Index → Retriever → QueryEngine graph Document-centric RAG with node abstractions
Retrieval	Pluggable retrievers; simple vector search by default Easy swapping and rerank via integrations	Node parsers, query transforms, hybrid/fusion Stronger recall@k in our setup
Tool orchestration	Mature agent patterns (ReAct, tool/function calling) Wide tool ecosystem; lower coordination overhead	Tools as QueryEngines/ToolSpecs; index routing Strong for RAG-centric tools and routing tasks
Performance (this project)	Lower latency on average Similar F1 despite lower recall at high k	Higher recall@k; better semantic scores (e.g., BARTScore/ROUGE) Slightly higher latency due to query transforms

Prefer LangChain when:

- Latency is critical and tool use is rich/heterogeneous
- You need broad integrations and rapid prototyping

Prefer LlamaIndex when:

- Retrieval quality and index routing are primary
- You want stronger semantic faithfulness in RAG

4A. Quantitative Results

Comparison on 1,000 CUAD-RAG queries (k ∈ {10, 20, 100})

Dataset: CUAD-RAG Benchmark

System Performance Metrics

Metric breakdown by Retrieval Depth (k)

FRAMEWORK	K	RECALL@K	LATENCY (S)	TOOL EFF.	F1 SCORE
LangChain	10	0.309	**4.89**	0.841	0.317
	20	0.342	**4.84**	0.873	0.335
	100	0.342	**4.86**	0.876	0.335
LlamaIndex	10	0.516	8.49	0.935	0.319
	20	**0.656**	9.52	**0.950**	**0.341**
	100	**0.795**	**30.88**	**0.957**	0.333

Semantic Quality (at k=20)

METRIC	LANGCHAIN	LLAMAINDEX
BLEU	0.301	**0.308**
ROUGE-L	0.405	**0.413**
BARTScore	-1.189	**-1.156**

Key Performance Deltas

- **Retrieval Superiority:** LlamaIndex achieves **2.3× higher recall** at k=100 compared to LangChain.
- **Latency Stability:** LangChain maintains stable **~4.8s latency** regardless of retrieval depth, whereas LlamaIndex spikes to 30s.
- **Semantic Alignment:** Despite similar F1 scores, semantic metrics (BARTScore, ROUGE) consistently favor LlamaIndex.

4B. Key Insights

Trade-off Analysis & Architectural Implications

Analysis based on 1,000 test cases

Retrieval vs. Latency Trade-off

A fundamental trade-off exists between completeness and speed. LlamaIndex prioritizes **depth** (high recall) at the cost of exponential latency growth (up to 30s). LangChain optimizes for **stability** (~4.8s) but sacrifices retrieval coverage.

```
Data: Recall +48% vs Latency +535% (LlamaIndex k=100)
```

The "F1 Illusion" & Semantics

While token-level F1 scores are comparable (~0.33), they mask quality differences. Better retrieval directly correlates with higher **semantic alignment** (BARTScore/ROUGE), meaning LlamaIndex answers are factually closer to gold standards despite similar keywords.

```
= Insight: Retrieval quality ≠ Generation tokens, but = Meaning
```

Retrieval Scaling Dynamics

Frameworks respond differently to increased context. LangChain's recall **plateaus** after k=20, suggesting architectural limits in candidate processing. LlamaIndex continues to gain recall up to k=100, effectively utilizing the expanded context window.

```
Trend: LangChain saturates early; LlamaIndex scales
```

Orchestration Efficiency

LlamaIndex demonstrates higher **tool efficiency** (0.95 vs 0.87), indicating more precise agentic reasoning. It invokes tools slightly more often but with significantly better alignment to user intent, reducing wasted computational cycles on irrelevant actions.

```
Stat: +8% Efficiency gap favors LlamaIndex
```

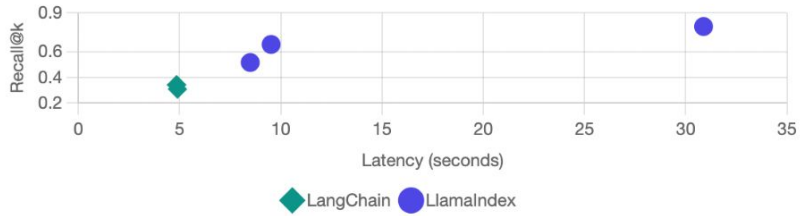
5. Demo & Visualization

Performance dynamics and agent behavior analysis

Live Metrics from WandB

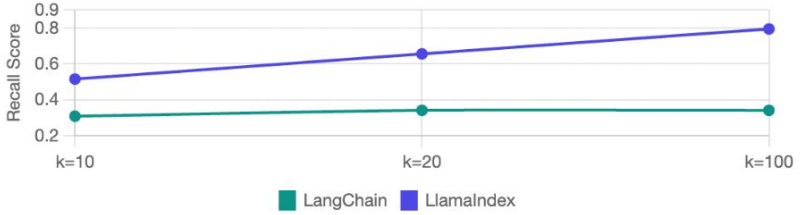
Recall vs. Latency Trade-off

Critcal Insight



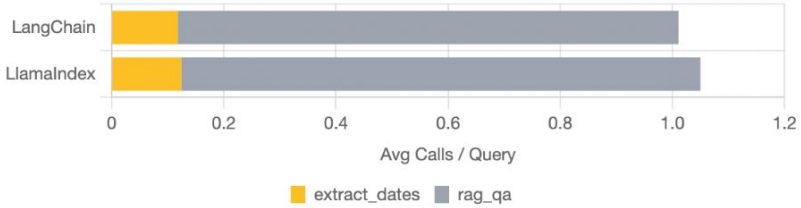
Notice: LlamaIndex (Indigo) pushes right (latency) for height (recall), while LangChain (Teal) clusters in the low-latency zone.

Retrieval Recall@k Scaling



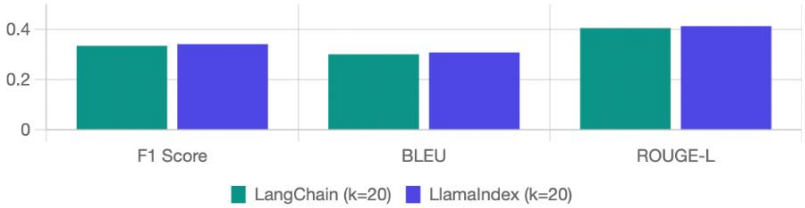
Notice: LangChain plateaus at k=20, whereas LlamaIndex continues to benefit from deeper retrieval (up to k=100).

Agent Tool Orchestration (Avg Calls/Query)



Notice: Both agents favor `rag_qa` (~90%). LlamaIndex invokes tools slightly more frequently as context deepens.

Semantic Answer Quality (at k=20)



Notice: While F1 is similar, LlamaIndex leads in ROUGE-L and BLEU, indicating better structural answer alignment.

Data Source: CUAD-RAG Experiments (Table I, II, III)

6. Lessons Learned

Implications for Legal RAG Design & Evaluation

Key Takeaways from CUAD-RAG Study

Framework Choice Matters

Orchestration is not merely an implementation detail. It fundamentally dictates **retrieval depth**, **latency profiles**, and **tool behavior** beyond simple model accuracy.

Architectural Trade-offs

Retrieval ≠ Generation

High recall (LlamaIndex) does not guarantee better F1 scores, but it significantly improves **semantic alignment** (BARTScore) and answer fidelity.

Quality vs. Token Match

Tool Orchestration Critical

Tool efficiency is a first-class metric. Superior tool selection correlates with better semantic quality, proving that **agent design** directly impacts output reliability.

Minimize Waste Calls

Multi-Metric Evaluation is Essential

Single metrics mislead. A robust evaluation must triangulate **F1/BLEU** (text match), **Recall@k** (retrieval), **Latency** (UX), and **Tool Efficiency** (system health) to capture the full picture of RAG performance.

Holistic Benchmarking

Deep Instrumentation Pays Off

Granular tracing (e.g., WandB) is non-negotiable for RAG. It reveals hidden behaviors like **latency tails** at depth k=100 and precise **agent decision paths** that aggregate metrics obscure.

Visibility = Reliability

7. Future Directions

Roadmap for Legal RAG Research

Next Steps

Multi-hop Reasoning Evaluation

Extend benchmark to queries requiring synthesis across multiple contract sections (cross-document reasoning), addressing current single-hop limitations.

Hybrid Architectures

Combine **LlamaIndex's** superior retrieval backend with **LangChain's** low-latency orchestration engine to optimize the recall-latency frontier.

Advanced Agentic Tools

Move beyond keyword heuristics to LLM-based tool selection. Implement complex tools for clause comparison and risk scoring.

Domain Extension

Validate findings on high-stakes domains like **Medical (EHR)** and **Finance (10-K)**, varying chunking strategies and embedding models.

Links and references

Slides beautified with Genspark
Github Link: Link
Wanb Link: Link
Wand Report Link: Link
Report Link: Link

Thank you