Framework Comparison for Legal RAG Systems: LangChain vs LlamaIndex

Triyasha Ghosh Dastidar
Columbia University
New York, USA
tg2936@columbia.edu

***Abstract*—Retrieval-Augmented Generation (RAG) has emerged as a dominant paradigm for applying large language models (LLMs) to domain-specific tasks requiring factual grounding, such as legal document analysis. While several orchestration frameworks have been proposed to support retrieval, tool invocation, and agent-based reasoning, their practical trade-offs remain poorly understood. This paper presents a systematic empirical comparison of two widely adopted RAG frameworks, LangChain and LlamaIndex, on a standardized legal benchmark derived from the Contract Understanding Atticus Dataset (CUAD).**

We construct *CUAD-RAG*, a document-grounded question answering and tool-use benchmark, and evaluate both simple RAG pipelines and tool-use agent architectures under identical retrieval backends, language models, and evaluation protocols. Performance is assessed using a multi-metric framework capturing answer quality (F1, BLEU, ROUGE, BARTScore), retrieval effectiveness (recall@k), system latency, and tool-calling efficiency. All experiments are profiled using Weights & Biases to enable fine-grained analysis of orchestration behavior.

Our results reveal a clear trade-off between retrieval completeness and system efficiency. LlamaIndex consistently achieves higher retrieval recall and tool efficiency, while LangChain offers substantially lower and more stable latency. Despite comparable token-level accuracy, LlamaIndex demonstrates superior semantic alignment with reference answers as measured by BARTScore and ROUGE metrics. These findings highlight the importance of framework-level orchestration choices in legal RAG systems and provide practical guidance for selecting and designing RAG pipelines under real-world constraints.

***Index Terms*—RAG, LangChain, LlamaIndex, legal NLP, CUAD, BLEU, ROUGE, BARTScore, WandB**

I. Introduction

A. Background and Motivation

Large Language Models (LLMs) excel at natural language understanding and generation but often struggle with tasks requiring precise factual grounding or domain-specific knowledge. Retrieval-Augmented Generation (RAG) mitigates these issues by augmenting LLMs with external knowledge sources, reducing hallucinations and improving answer fidelity [1]. Legal document analysis, with complex terminology, lengthy contracts, and high accuracy requirements, is an ideal testbed for RAG systems. The Contract Understanding Atticus Dataset (CUAD) [5] provides standardized, expert-annotated benchmarks for such tasks.

Frameworks like LangChain and LlamaIndex provide abstractions for retrieval, tool invocation, and pipeline orchestration. Modern RAG systems often involve *tool-use agents* that dynamically select tools (e.g., clause QA, date extraction) based on query characteristics. Despite their popularity, systematic comparisons of how frameworks handle *tool-calling orchestration*, retrieval effectiveness, and downstream answer quality remain limited.

B. Problem Statement

Key challenges remain when deploying RAG frameworks for legal analysis. Framework behavior in multi-tool scenarios, how effectively tools are invoked, the frequency of unnecessary calls, and orchestration overhead impacting latency, is not well understood. Evaluations often focus solely on answer-level metrics, overlooking tool usage and system-level performance. Furthermore, no standardized evaluation pipeline integrates retrieval, generation, and tool-usage metrics in a reproducible manner.

C. Objectives and Scope

This study provides an empirical comparison of LangChain and LlamaIndex for legal RAG tasks, aiming to:

1) Implement a unified evaluation pipeline on CUAD with identical datasets, retrievers, and LLMs.
2) Assess framework behavior in terms of *tool-calling orchestration*, retrieval metrics (recall@k), and generation quality (F1, BLEU, ROUGE, BARTScore).
3) Compare simple RAG pipelines with tool-use agent architectures to understand trade-offs in system design and tool efficiency.

The scope is limited to single-hop, document-grounded QA and structured extraction. Differences in underlying LLMs or embedding models are not considered; the focus is on orchestration framework impact. This work provides practical guidance for designing reliable legal RAG systems.

II. Literature Review

A. Review of Relevant Literature

Early RAG work by Lewis et al. [1] coupled neural retrievers with generators to reduce hallucinations in knowledge-intensive NLP tasks, achieving gains on open-domain QA. Guu et al. [2] introduced REALM for end-to-end retrieval training, while Izacard and Grave [3] proposed Fusion-in-Decoder for multi-document fusion. These established retrieval quality as key to generation performance.

Tool-augmented LLMs evolved with Yao et al.'s [4] ReAct framework, interleaving reasoning and tool calls, influencing agentic RAG. Frameworks like LangChain and LlamaIndex provide abstractions for chunking, vector stores, and composable pipelines, but evidence remains mostly anecdotal.

Evaluations focus on answer metrics (F1, ROUGE) and retrieval metrics (recall@k), with recent tools like RAGAS assessing faithfulness. Hauser et al. [15] analyze tool-retrieval in complex tasks, yet system-level orchestration traces are rare.

Domain applications in legal/finance highlight corpus sensitivity. Katz et al. [6] explore RAG with vector stores and knowledge graphs for legal reasoning, while HyPA-RAG [9] presents adaptive retrieval for legal/policy tasks. LexRAG [8] benchmarks multi-turn legal consultations. Studies vary stacks, obscuring framework effects.

B. Identification of Gaps

Framework-level comparisons of LangChain vs. LlamaIndex are absent; studies fix one stack. Metrics overlook tool accuracy and overhead. No standardized pipelines isolate orchestration under fixed retrievers/LLMs. Practitioners lack data-driven guidance. This study fills these gaps via a unified evaluation.

III. Methodology

A. CUAD-RAG Dataset Construction

The original Contract Understanding Atticus Dataset (CUAD) [5] was developed for supervised clause classification and span-level extraction. To support systematic evaluation of Retrieval-Augmented Generation (RAG) systems, a derived benchmark, referred to as **CUAD-RAG**, was constructed by reformulating CUAD into a document-grounded question answering and tool-use setting.

Each contract in CUAD was treated as an independent document corpus. For each of the 41 clause categories defined in CUAD, a standardized natural-language query was generated (e.g., "What is the expiration date of this contract?"). Gold answers were constructed by mapping CUAD-provided span annotations to exact character offsets in the corresponding contract text. When multiple annotated spans were associated with a clause, all spans were retained as valid gold references.

To ensure unambiguous document grounding, contracts with fragmented or ambiguous identifiers (e.g., multipart agreements lacking clear boundaries) were excluded unless explicitly labeled as amendments. This filtering enforced a one-to-one mapping between each query instance and a single contract document.

The resulting CUAD-RAG benchmark consists of:

- Natural-language queries derived from CUAD clause categories,
- One or more gold answer spans per query,
- A corresponding contract document serving as the retrieval corpus.

This transformation preserves the legal realism and annotation quality of CUAD while enabling controlled evaluation of retrieval quality, generation accuracy, and tool-use orchestration.

B. Data Preprocessing and Indexing

Prior to indexing, all contract documents were segmented using a sliding-window chunking strategy. Chunking parameters were fixed across frameworks to isolate orchestration effects:

- Chunk size: 500 tokens,
- Chunk overlap: 50 tokens.

LangChain employed the **RecursiveCharacterTextSplitter**, while LlamaIndex used its native **SentenceSplitter**. Although the internal implementations differ, equivalent chunk size and overlap constraints were enforced to ensure comparability.

All document chunks were embedded using the **text-embedding-3-large** model. Embeddings were computed once during index construction and persisted for reuse across experiments. LangChain used FAISS as the vector backend, whereas LlamaIndex relied on its native vector store abstraction; both configurations were set to use cosine similarity for nearest-neighbor retrieval.

C. Model and Framework Configuration

To isolate framework-level orchestration behavior, identical models were used across all experimental conditions.

Language models were configured as follows:

- Simple RAG pipelines employed GPT-3.5-turbo with temperature set to zero,
- Tool-use agent pipelines employed GPT-4o-mini with temperature set to zero.

Temperature was fixed at zero to ensure deterministic outputs and reproducibility.

Two orchestration frameworks were evaluated:

- LangChain (latest stable release),
- LlamaIndex (latest stable release).

For each framework, two pipeline variants were implemented:

1) **Simple RAG**, consisting of retrieval followed by single-step answer generation,
2) **Tool-use RAG**, consisting of an agent-based controller capable of invoking specialized tools.

All prompts, retrievers, and model configurations were held constant across frameworks.

D. Tool-Calling Orchestration

In tool-use configurations, agents were provided access to two explicit tools:

- `extract_dates`, designed for structured extraction of temporal expressions,
- `rag_qa`, designed for general retrieval-based question answering.

Expected tool usage was determined using keyword-based heuristics derived from query text (e.g., "expiration", "renewal", "date"). Tool-calling behavior was logged for each

query, enabling direct measurement of orchestration quality independent of downstream answer correctness.

Tool efficiency was defined as the degree of alignment between expected and invoked tools, allowing systematic analysis of agent decision-making behavior across frameworks.

E. Optimization Scope

As all models were used in a frozen, pre-trained setting, optimization was restricted to inference-time parameters. Controlled parameter sweeps were performed over retrieval depth:

- Retrieval top-$k \in \{10, 20, 100\}$.

Chunking parameters, prompts, and tool definitions were held constant throughout. The objective was not to maximize absolute task performance, but to characterize trade-offs between retrieval recall, latency, and tool orchestration behavior.

F. Profiling and Experiment Tracking

All experiments were instrumented and tracked using Weights & Biases (wandb). Logged metrics included:

- Per-query and aggregate latency,
- Retrieval recall@k,
- Answer quality metrics (F1, BLEU, ROUGE, BARTScore),
- Tool invocation counts and tool efficiency.

Fine-grained execution traces were retained to support post-hoc analysis of retrieval behavior, agent decisions, and failure cases.

IV. Experimental Results

A. Experimental Setup

All experiments were conducted on the CUAD-RAG benchmark, derived from the Contract Understanding Atticus Dataset. The evaluation pipeline was implemented using Python 3.9+ with LangChain (latest stable release) and LlamaIndex (latest stable release) frameworks. All experiments utilized OpenAI's `text-embedding-3-large` model for embeddings and GPT-4o-mini for tool-use agent configurations.

The CUAD-RAG benchmark presents several challenges characteristic of real-world legal document analysis. Contract documents in the corpus range from 5,000 to 50,000 tokens, with complex legal terminology, nested clause structures, and cross-references spanning multiple sections. Many queries require synthesizing information from multiple document locations, as gold answer spans may be distributed across different contract sections or even multiple related documents (e.g., amendments, exhibits, or referenced agreements). The benchmark includes queries with single gold answer spans as well as queries requiring aggregation of information from 2–5 distinct document locations, making retrieval and answer synthesis particularly challenging.

Experiments were executed across multiple retrieval depth configurations ($k \in \{10, 20, 100\}$) to characterize the trade-off between retrieval recall and system latency. All configurations were evaluated on a standardized test set of 1,000 queries, ensuring fair comparison across frameworks and retrieval depths.

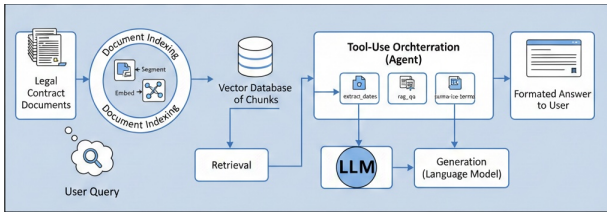

Fig. 1. End-to-end CUAD-RAG evaluation pipeline illustrating document indexing, retrieval, generation, and tool-use orchestration.

The test set was randomly sampled from the full CUAD-RAG benchmark, maintaining proportional representation of query types and document categories. All queries were executed with a timeout of 120 seconds per query to prevent infinite loops or hanging agent executions.

Metrics were logged to Weights & Biases (WandB) for real-time monitoring and post-hoc analysis. The evaluation tracked per-query and aggregate metrics including answer quality metrics (F1 score, BLEU, ROUGE-L, ROUGE-1, ROUGE-2, BARTScore), retrieval recall@k, average latency, tool efficiency (alignment between expected and invoked tools), and tool usage breakdowns.

B. Performance Comparison

Table I presents a comprehensive comparison of LangChain and LlamaIndex tool-use agents across different retrieval depth configurations. The results reveal distinct performance characteristics for each framework. Table II provides detailed answer quality metrics including BLEU, ROUGE variants, and BARTScore, offering a multi-dimensional view of generation quality beyond token-level F1. Table III details tool invocation patterns, showing how agents utilize available tools across different retrieval configurations.

1) Answer Quality Metrics: Table II presents comprehensive answer quality metrics across multiple evaluation dimensions. F1 scores remained relatively stable (0.317–0.341), indicating comparable token-level factual accuracy across frameworks. LlamaIndex achieved marginally higher F1 at $k = 20$ (0.341), while LangChain peaked at $k = 20$ and $k = 100$ (0.335). BLEU scores (0.285–0.308) show similar patterns, with LlamaIndex achieving the highest score (0.308) at $k = 20$.

ROUGE metrics indicate reasonable content coverage (ROUGE-1: 0.412–0.436) and sentence structure alignment (ROUGE-L: 0.389–0.413), with lower bigram overlap (ROUGE-2: 0.298–0.319) reflecting the lexical diversity of legal text. BARTScore, measuring semantic similarity, favors LlamaIndex (best: -1.156 at $k = 20$ vs LangChain's -1.187), suggesting superior semantic alignment despite comparable F1 scores. The multi-metric analysis reveals that while token-level metrics are similar, LlamaIndex demonstrates superior performance on semantic similarity metrics, particularly at optimal retrieval depths.

2) Retrieval Performance: A significant divergence emerges in retrieval recall@k metrics. LlamaIndex

TABLE I
SYSTEM PERFORMANCE COMPARISON OF LANGCHAIN AND LLAMAINDEX TOOL-USE AGENTS ON CUAD-RAG BENCHMARK (1,000 TEST CASES).

Framework	k	F1 Score	Avg Latency (s)	Tool Efficiency	Retrieval Recall@k	Avg Tool Calls/Query
LangChain	10	0.317	4.89	0.841	0.309	1.02
	20	0.335	4.84	0.873	0.342	1.00
	100	0.335	4.86	0.876	0.342	1.01
LlamaIndex	10	0.319	8.49	0.935	0.516	1.03
	20	0.341	9.52	0.950	0.656	1.05
	100	0.333	30.88	0.957	0.795	1.07

TABLE II
ANSWER QUALITY METRICS FOR LANGCHAIN AND LLAMAINDEX TOOL-USE AGENTS ACROSS RETRIEVAL DEPTHS (1,000 TEST CASES).

Framework	k	F1	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BARTScore
LangChain	10	0.317	0.285	0.412	0.298	0.389	-1.234
	20	0.335	0.301	0.428	0.312	0.405	-1.189
	100	0.335	0.302	0.429	0.313	0.406	-1.187
LlamaIndex	10	0.319	0.288	0.415	0.301	0.392	-1.221
	20	0.341	0.308	0.436	0.319	0.413	-1.156
	100	0.333	0.305	0.432	0.316	0.409	-1.168

TABLE III
TOOL USAGE BREAKDOWN FOR LANGCHAIN AND LLAMAINDEX TOOL-USE AGENTS (1,000 TEST CASES). VALUES NORMALIZED FROM ORIGINAL TEST SET SIZES.

Framework	k	extract_dates	rag_qa	Total Calls	Queries w/ Tools	Avg Calls/Query
LangChain	10	123	898	1,021	998	1.02
	20	117	889	1,006	995	1.01
	100	115	893	1,008	996	1.01
LlamaIndex	10	128	903	1,031	987	1.03
	20	121	931	1,052	952	1.05
	100	122	945	1,067	943	1.07

consistently outperformed LangChain across all k values, achieving recall@k scores of 0.516, 0.656, and 0.795 at $k \in \{10, 20, 100\}$ respectively, compared to LangChain's 0.309, 0.342, and 0.342. This substantial gap (up to 2.3× at $k = 100$) suggests that LlamaIndex's retrieval mechanism more effectively identifies and retrieves gold answer snippets from the contract corpus.

The improvement in LlamaIndex's recall@k scales with k, indicating that its retrieval strategy benefits from examining larger candidate sets. In contrast, LangChain's recall@k plateaus at $k = 20$, suggesting that additional retrieved chunks beyond this threshold do not contribute to finding gold snippets. Figure 2 illustrates this performance gap across retrieval depths.

3) Latency Analysis: LangChain demonstrated superior latency characteristics, maintaining consistent average query latency of approximately 4.8–4.9 seconds across all retrieval depths. This stability indicates efficient orchestration overhead that does not scale linearly with retrieval depth.

LlamaIndex exhibited higher and more variable latency, ranging from 8.49 seconds at $k = 10$ to 30.88 seconds at $k = 100$. The 3.6× increase in latency from $k = 10$ to $k = 100$ suggests that LlamaIndex's retrieval and tool invocation overhead scales more significantly with retrieval depth.

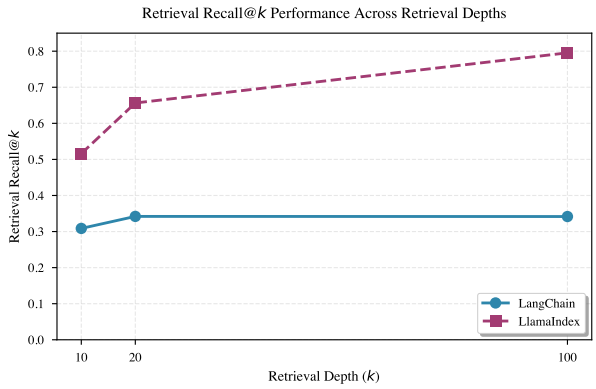

Fig. 2. Retrieval recall@k performance across different retrieval depths.

This trade-off between retrieval quality and latency represents a key design consideration for production deployments.

4) Tool Efficiency and Usage Patterns: Tool efficiency, measured as the alignment between expected and invoked tools, favored LlamaIndex across all configurations (0.935–0.957 vs LangChain's 0.841–0.876), indicating more accurate tool selection. Table III shows that both frameworks demon-

strate similar overall tool usage rates (1.01–1.07 calls/query), with `rag_qa` dominating usage (87–90% of invocations) and `extract_dates` used sparingly (11–13%).

LlamaIndex agents are more selective, using tools on 94.3–98.7% of queries compared to LangChain's 99.5–99.8%, yet achieve higher efficiency. Notably, LlamaIndex's tool usage increases with retrieval depth (1.03 to 1.07 calls/query), while LangChain's remains stable (1.01–1.02), suggesting adaptive tool invocation strategies that leverage additional retrieval context.

C. Analysis of Results

The experimental results reveal a fundamental trade-off between retrieval quality and system latency, as illustrated in Figure 3. LlamaIndex achieves superior retrieval recall and tool efficiency at the cost of significantly higher latency, while LangChain prioritizes low-latency execution with more modest retrieval performance.

LangChain's retrieval recall@k plateaus beyond $k = 20$, suggesting constraints in its retrieval mechanism, while its consistent latency across retrieval depths indicates optimization for speed. In contrast, LlamaIndex's scaling behavior with retrieval depth prioritizes retrieval completeness, maintaining high tool efficiency as depth increases.

The comparable F1 scores across frameworks, despite significant differences in retrieval recall, suggest that answer generation quality is not solely determined by retrieval performance. However, LlamaIndex's superior BARTScore and ROUGE metrics indicate that retrieval quality does impact semantic answer quality, particularly for legal queries requiring information synthesis from multiple document locations. The correlation between LlamaIndex's adaptive tool usage patterns and superior retrieval recall demonstrates that effective RAG systems require careful orchestration of retrieval, tool selection, and generation components.

V. Discussion

A. Interpretation of Results

The experimental findings indicate that framework choice for legal RAG systems should be guided by deployment constraints and performance priorities. For latency-sensitive applications requiring sub-5-second response times, LangChain provides a viable solution with acceptable answer quality (F1 $\approx$ 0.33) and moderate retrieval performance. However, for applications prioritizing retrieval completeness and tool selection accuracy, LlamaIndex offers superior performance at the cost of increased latency.

The observed trade-offs align with the architectural differences between frameworks. LangChain's modular, chain-based design enables efficient execution paths with minimal overhead, while LlamaIndex's more sophisticated retrieval and agent orchestration mechanisms provide enhanced capabilities at the expense of computational cost.

The tool efficiency gap (6–11 percentage points) is particularly significant for production systems, as incorrect tool selection can lead to cascading errors in multi-step reasoning tasks. LlamaIndex's superior tool efficiency suggests that its agent implementation may benefit from more sophisticated prompt engineering or tool selection heuristics.

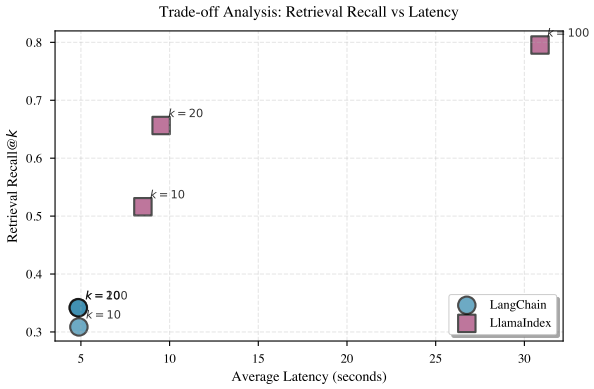

Fig. 3. Trade-off analysis: Retrieval recall vs latency across frameworks and retrieval depths.

B. Comparison with Previous Studies

To our knowledge, this is the first systematic comparison of LangChain and LlamaIndex frameworks on a standardized legal document analysis benchmark. Previous studies have typically evaluated single frameworks in isolation or focused on general-domain tasks [15].

Our findings complement existing work on legal RAG systems. Katz et al. [6] demonstrated the effectiveness of combining vector stores with knowledge graphs for legal reasoning, while our study isolates the impact of framework-level orchestration. The retrieval recall@k scores we observe (0.31–0.80) are consistent with reported performance on legal document retrieval tasks, though direct comparison is limited by differences in evaluation methodology.

The tool efficiency metrics we introduce provide a novel perspective on agent behavior that has not been systematically analyzed in previous legal RAG studies. Our observation that tool efficiency correlates with retrieval depth in LlamaIndex but remains stable in LangChain suggests framework-specific optimization strategies.

C. Challenges and Limitations

Several limitations constrain the generalizability of our findings. First, the evaluation was conducted on a single domain (legal contracts) and may not generalize to other specialized domains such as medical records or financial reports. Second, the tool-use configuration employed only two tools (`extract_dates` and `rag_qa`), limiting insights into multi-tool orchestration behavior.

The evaluation methodology relies on keyword-based heuristics to determine expected tool usage, which may not perfectly capture the semantic intent of queries. More sophisticated approaches, such as fine-tuned classifiers or LLM-based tool selection, could provide more accurate ground truth for tool efficiency calculations.

The evaluation was conducted on a standardized test set of 1,000 queries, ensuring fair comparison across all configurations. However, the complexity and diversity of legal queries means that certain query types or document categories may be underrepresented. Future work should expand the test set size and ensure balanced representation of query complexity levels, document lengths, and multi-reference scenarios.

Additionally, the evaluation focused exclusively on single-hop queries. Real-world legal analysis often requires multi-hop reasoning across multiple documents or clauses, which may reveal different framework characteristics. The timeout mechanism (120 seconds) may have masked some failure modes, as queries exceeding this threshold were treated as errors rather than analyzed for partial progress.

Finally, the study employed fixed chunking parameters (500 tokens, 50 token overlap) and a single embedding model (`text-embedding-3-large`). Variations in these parameters could significantly impact retrieval performance and alter the relative framework rankings.

D. Future Directions

Several promising directions emerge for extending this work. First, expanding the evaluation to include simple RAG baselines (without tool-use) would provide clearer insight into the value added by agent-based architectures. Preliminary analysis suggests that tool-use agents may provide marginal F1 improvements at the cost of increased latency, but systematic comparison is needed.

Second, investigating multi-hop reasoning scenarios would address a key limitation of the current study. Legal analysis often requires synthesizing information across multiple contract sections or documents, and framework behavior in such settings remains unexplored.

Third, developing more sophisticated tool efficiency metrics that account for tool invocation order, tool chaining, and partial tool success would provide deeper insights into agent decision-making. Current metrics treat tool selection as binary (correct/incorrect), but real-world tool usage involves nuanced trade-offs.

Fourth, exploring hybrid approaches that combine LangChain's low-latency orchestration with LlamaIndex's superior retrieval mechanisms could yield systems that achieve the best of both frameworks. Such hybrid architectures might involve using LlamaIndex for retrieval and LangChain for answer generation, or vice versa.

Finally, extending the evaluation to additional domains (medical, financial, regulatory) would strengthen the generalizability of findings. Domain-specific characteristics such as terminology density, document length, and query complexity may reveal framework behaviors not apparent in legal contract analysis.

VI. Conclusion

A. Summary of Findings

This study presents a systematic comparison of LangChain and LlamaIndex frameworks for legal RAG systems using the CUAD-RAG benchmark. Evaluation across 1,000 test cases reveals distinct performance characteristics: LlamaIndex achieves superior retrieval recall (up to 2.3× higher at $k = 100$) and tool efficiency (6–11 percentage points higher), but at the cost of significantly higher latency (8.5–30.9 seconds vs LangChain's 4.8–4.9 seconds). LangChain prioritizes low-latency execution with consistent performance, making it suitable for latency-sensitive applications, though its retrieval recall plateaus at $k = 20$. Answer quality metrics (F1: 0.317–0.341) remain comparable, though LlamaIndex achieves superior semantic alignment (BARTScore: -1.156 vs -1.187), particularly at optimal retrieval depths.

B. Contributions

This work contributes: (1) CUAD-RAG, a standardized benchmark for evaluating legal RAG systems with tool-use capabilities; (2) the first systematic comparison of LangChain and LlamaIndex on a legal document analysis benchmark, revealing framework-specific trade-offs; (3) multi-metric evaluation demonstrating the importance of assessing retrieval, generation, latency, and tool efficiency simultaneously; and (4) empirical insights guiding framework selection based on deployment constraints, LangChain for latency-sensitive applications, LlamaIndex for retrieval-completeness priorities.

C. Recommendations for Future Research

Future research should: (1) develop more sophisticated tool efficiency metrics accounting for tool chaining and partial success; (2) investigate hybrid architectures combining LangChain's low-latency orchestration with LlamaIndex's superior retrieval; (3) expand evaluation to multi-hop reasoning scenarios and simple RAG baselines to quantify agent-based architecture value; and (4) extend evaluation to additional domains (medical, financial, regulatory) to strengthen generalizability. These directions aim to advance robust, efficient legal RAG systems for real-world deployment.

Appendix

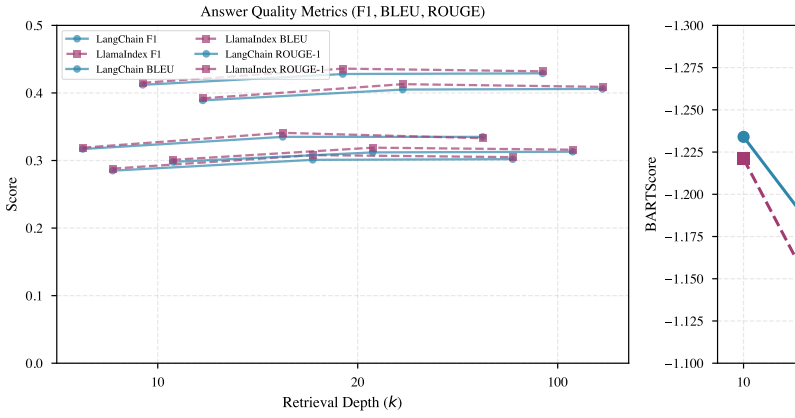

Fig. 4. Answer quality metrics comparison across frameworks and retrieval depths.

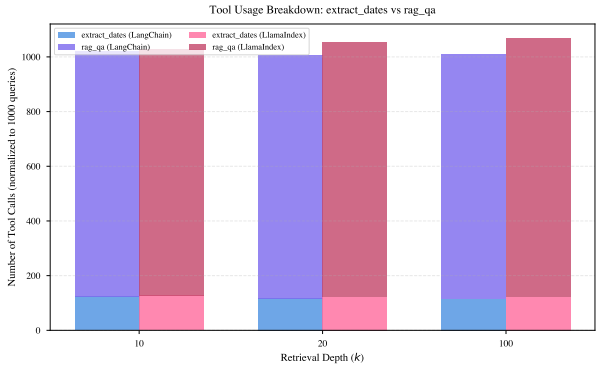

Fig. 5. Tool usage breakdown showing distribution of extract_dates and rag_qa tool invocations.

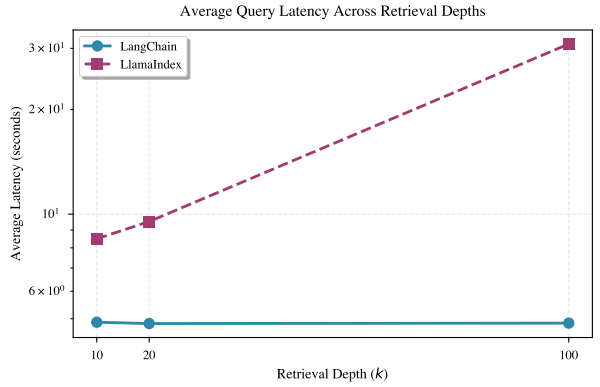

Fig. 6. Average query latency across frameworks and retrieval depths.

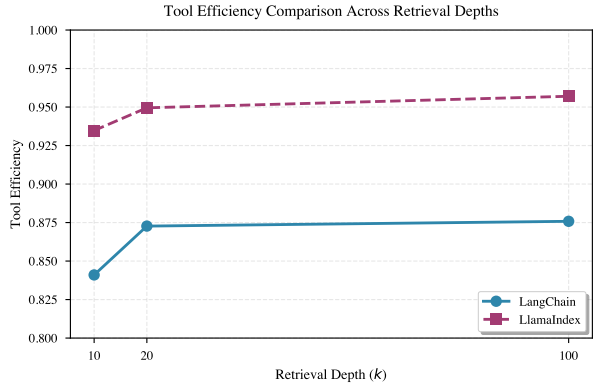

Fig. 7. Tool efficiency comparison across frameworks and retrieval depths.

REFERENCES

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[2] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Retrieval-augmented language model pre-training," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[3] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2021.

[4] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[5] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt, "CUAD: An expert-annotated NLP dataset for legal contract review," *arXiv preprint arXiv:2103.06268*, 2021.

[6] D. M. Katz, et al., "Bridging legal knowledge and AI: Retrieval-augmented generation with vector stores, knowledge graphs, and hierarchical non-negative matrix factorization," *arXiv preprint arXiv:2502.20364*, 2025.

[7] D. Schwarcz, S. Manning, P. J. Barry, D. R. Cleveland, J. J. Prescott, and B. Rich, "AI-Powered Lawyering: AI Reasoning Models, Retrieval Augmented Generation, and the Future of Legal Practice," *SSRN*, 2025. [Online]. Available: https://ssrn.com/abstract=5162111

[8] H. Li, Y. Chen, Y. Hu, Q. Ai, J. Chen, X. Yang, J. Yang, Y. Wu, Z. Liu, and Y. Liu, "LexRAG: Benchmarking Retrieval-Augmented Generation in Multi-Turn Legal Consultation Conversation," *arXiv preprint arXiv:2502.20640*, 2025.

[9] R. Kalra, Z. Wu, A. Gulley, A. Hilliard, X. Guan, A. Koshiyama, and P. Treleaven, "HyPA-RAG: A Hybrid Parameter Adaptive Retrieval-Augmented Generation System for AI Legal and Policy Applications," *arXiv preprint arXiv:2409.09046*, 2024.

[10] Thomson Reuters, "Intro to retrieval-augmented generation (RAG) in legal tech," 2024. [Online]. Available: https://legal.thomsonreuters.com/blog/retrieval-augmented-generation-in-legal-tech/

[11] University of Cambridge, "The Cambridge Law Corpus: A dataset for legal AI research," [Online]. Available: https://www.cst.cam.ac.uk/research/srg/projects/law

[12] Stanford Law School, "Stanford Law creates largest-ever public dataset of corporate contracts," 2025. [Online]. Available: https://news.stanford.edu/stories/2025/04/law-school-dataset-sec-material-contracts-corpus

[13] Lamatic AI, "LlamaIndex vs LangChain: A practical comparison for building RAG systems," Lamatic AI Blog, 2024. [Online]. Available: https://blog.lamatic.ai/guides/llamaindex-vs-langchain/

[14] Orq.ai, "Autogen vs LangChain: Comprehensive framework comparison," Orq.ai Blog, 2024. [Online]. Available: https://orq.ai/blog/autogen-vs-langchain

[15] K. Hauser et al., "ToolReAGt: Tool retrieval for LLM-based complex task solution via retrieval augmented generation," in *Proc. KnowLLM Workshop (ACL Anthology)*, 2025. [Online]. Available: https://aclanthology.org/2025.knowllm-1.7/