Embedding-Free RAG

Jessica Maghakian, Raunak Sinha*, Gunkirat Kaur*, Max Schettewi*
Goldman Sachs
{jessica.maghakian, raunak.sinha, gunkirat.x.kaur, max.j.schettewi}@gs.com

Abstract

Retrieval-Augmented Generation (RAG) is the current state-of-the-art method for mitigating the shortcomings of large language models (LLMs) by incorporating external knowledge sources to provide more relevant and accurate responses to user queries. However building performant RAG systems for real use-cases typically requires heavy investment from NLP experts, such as fine-tuning embedding models for specialized domains, experimenting with text chunking strategies and other niche hyperparameter tunings. We propose *Embedding-Free RAG*, a model-agnostic approach that enables the deployment of a one-size-fits-all RAG pipeline for user-provided grounding documents. Unlike traditional RAG, which relies on embedding models for information retrieval, Embedding-Free RAG leverages the generalized reasoning abilities of LLMs in a novel algorithmic framework during the retrieval stage. Extensive experiments demonstrate that Embedding-Free RAG outperforms existing state-of-the-art methods, achieving up to 4.6x higher F1 scores and up to 2x better question answering accuracy across a wide range of challenging domains.

1 Introduction

Large Language Models (LLMs) have made remarkable strides since the release of ChatGPT in late 2022 (Minaee et al., 2024). Despite advancements like multimodality (Li et al., 2024), large context windows (Team et al., 2023), and improved reasoning (OpenAI, 2024), the newest generation of LLMs still face challenges first noted during their rise to popularity. Issues like factual hallucination (Huang et al., 2023; Bai et al., 2024), knowledge staleness (Wang et al., 2024b), limited access to proprietary data (Ahmed et al., 2024), and poor answer attribution (Li et al., 2023) continue to affect even the most advanced models.

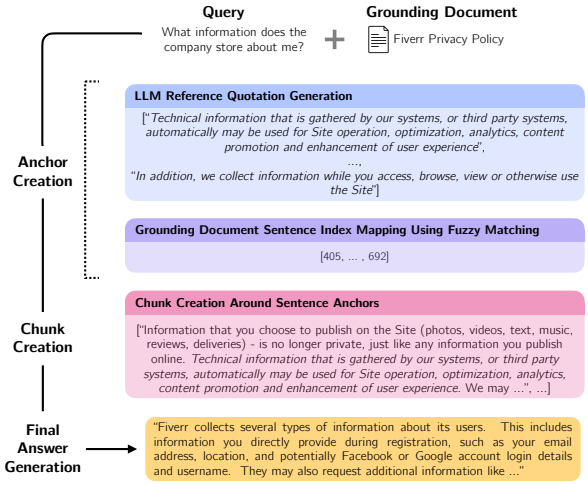

Figure 1: Overview of Embedding-Free RAG. Our framework uses LLMs in the retrieval stage to find relevant context in a user-provided grounding document.

Retrieval-Augmented Generation (RAG) has emerged as the leading technique to mitigate these shortcomings by incorporating external information into in-context learning (Lewis et al., 2020; Minaee et al., 2024). Despite the simple premise, deploying RAG systems for real-world applications requires substantial NLP expertise, including decisions on embedding models (Caspari et al., 2024), chunking strategies (Qu et al., 2024; Yepes et al., 2024), reranker models (Nogueira et al., 2019; Jin et al., 2022), and other optimizations (Ma et al., 2023; Guu et al., 2020). In specialized fields like finance, law, or medicine, custom fine-tuned embeddings are often needed (Dolphin et al., 2022; Chalkidis et al., 2020; Yuan et al., 2022), as off-the-shelf models are typically trained on general-purpose datasets and struggle with domain-specific linguistic patterns (Tang and Yang, 2024; Anderson et al., 2024).

The complex implementation of RAG systems and the inability of the framework to generalize have been a significant challenge for practitioners

*These authors contributed equally.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 24974–24985
November 4-9, 2025

(Wang et al., 2024a). Google's 2024 introduction of long-context LLMs (Pichai and Hassabis, 2024) was seen as a potential replacement, with some predicting the "death of RAG" (Dahl, 2024). However long-context LLMs have since been shown to have notable drawbacks, most prominently their susceptibility to positional bias (Tian et al., 2024). The "lost in the middle" phenomenon demonstrates that long-context LLMs struggle to effectively use relevant information located in the middle of an input during in-context learning (Liu et al., 2024). These models also face challenges with transparency and trustworthiness because their vast inputs are too large for humans to easily verify.

We introduce *Embedding-Free RAG*, a retrieval framework that replaces embedding-based search with an algorithmic approach that leverages the reasoning capabilities of LLMs. Our framework enables a unified, plug-and-play pipeline while restricting input to only relevant context, mitigating the "lost in the middle" issue in long-context LLMs. The result is a system that combines the strengths of traditional RAG and long-context models. By surfacing the exact text used to generate answers, Embedding-Free RAG improves trustworthiness and transparency — a necessity for fields where users rely on AI outputs to make sensitive decisions. The goal of our paper is specifically to address the understudied "Chat with a Document," scenario when users provide grounding documents with their query.

1.1 Motivation for "Chat with a Document"

Traditional RAG assumes external knowledge is unknown and must be retrieved from a broad corpus via search (Gao et al., 2023), typically by pre-indexing the corpus to enable scalable query-time retrieval. In contrast, real-world applications increasingly support "Chat with a Document" interactions (Microsoft, 2024; Adobe, 2024; PDF.ai, 2024), where users upload documents on-the-fly to ground AI-driven question answering. This user-provided context offers a practical path to integrating expert domain knowledge (Rane et al., 2024). While related to Document AI (Cui et al., 2021), our setting emphasizes real-time, user-driven dialogue rather than offline, structured extraction. It is also related to reading comprehension, where models answer questions from a single passage. However, unlike standard reading comprehension, "Chat with a Document" admits arbitrarily large inputs, demanding new solutions.

1.2 Key Contributions

In this work, we identify a critical gap in ad-hoc question answering over user-provided grounding documents and formalize the "Chat with a Document" problem for the first time in the academic literature. To address this setting, we propose *Embedding-Free RAG*, a novel algorithmic framework that replaces embedding-based retrieval with LLM-driven reasoning, enabling a one-size-fits-all RAG pipeline without the need for task-specific tuning or infrastructure. We demonstrate that Embedding-Free RAG delivers strong performance in complex, domain-specific question-answering tasks, validating its effectiveness in high-value, real-world use cases.

2 Embedding-Free RAG

We formally define the "Chat with a Document" Problem and introduce Embedding-Free RAG. In Section 2.2, we detail the generalized Embedding-Free RAG framework. Section 2.4 discusses some optimizations and algorithm variants, including parallelization in the retrieval stage (Section 2.4.1), use of task-specific LLMs (Section 2.4.2) and leveraging rich metadata from structured documents (Section 2.4.3).

2.1 Problem Statement

In *"Chat with a Document,"* the inputs are a user-provided document D and a query Q, with the goal of returning an answer A that accurately responds to Q using only the information in D. Although we assume a single document, the problem naturally extends to multi-document inputs.

This is an *online* problem: both D and Q are provided at inference time, and the system must return A with low latency. The low latency requirement prevents task-specific tuning, offline preprocessing or index construction on the revealed document and query. Traditional RAG pipelines that rely on pre-built dense indexes, or graph-based retrieval methods that require constructing and traversing knowledge graphs, are impractical in this setting. While D may originate from structured formats such as PDFs, we assume it is available as a raw text string, potentially sourced from diverse media (e.g., PowerPoint decks or scraped web content).

In some cases, D may be accompanied by metadata M that captures structural or positional information, such as section headers or page locations. We refer to this variant as *"Chat with a Structured*

Document," where the input is (D, M, Q) and the task remains to produce an answer A grounded in both D and M. We elaborate on this setting in Section 2.4.3.

2.2 Generalized Embedding-Free RAG

Embedding-Free RAG addresses the "Chat with a Document" problem while preserving the core retrieve-then-generate structure of traditional RAG. In traditional RAG, embedding models are used in the retrieval stage to retrieve relevant chunks of text that are then passed with the user query to an LLM to generate a final answer during the Generation stage. We mimic this structure in Embedding-Free RAG, while eliminating the need for retrieval-stage hyperparameter tuning, which is infeasible in online settings. Our approach involves two key steps:

1. *Anchor Creation:* identify text spans likely relevant to the query (Section 2.2.1).
2. *Chunk Construction:* expand anchors into context-rich chunks for answer generation (Section 2.2.2).

2.2.1 Anchor Creation

To create anchors, we first pass the document text to the LLM in a *reference quotation generation prompt*. The LLM's output should be a list of relevant reference quotations $\mathbf{r} = [r_1, \ldots, r_k]$, with each r_i corresponding to a sentence in the document that pertains to the user query. An example prompt template is provided in Appendix A.1. In the event that an empty list is returned ($\mathbf{r} = []$), we assume the document does not contain any relevant information and a grounded answer cannot be provided to the query.

Once the reference quotations $\mathbf{r}$ are obtained, they must be mapped to their corresponding positions in the original text. During experimentation, we found that LLMs may not exactly reproduce the original sentences. To identify the original source of a quotation generated by a language model (e.g., one with slight variations in spacing or spelling), we use the *Levenshtein distance* to find the most similar sentence in the original document. This allows for robust matching even when exact text comparison fails.

Definition 1. *The Levenshtein distance between two strings is the minimum number of single-character edits (insertions, deletions, or substitutions) needed to transform one into the other.*

Using sentence splitters, the input document D can be decomposed into its sentences $\mathbf{s} = [s_1, \ldots, s_n]$. For a reference quotation r_j, the index a_j of the matching original anchor sentence can be found using the Levenshtein distance:

$$a_j = \arg \min_{i \in [1,\ldots,n]} \text{lev}(r_j, s_i),$$

yielding a final list of anchors $\mathbf{a} = [a_1, \ldots, a_k]$. Numerous efficient implementations of the Levenshtein distance exist, such as the RapidFuzz library for Python (Bachmann). Experimental evaluations showing the time and space efficiency of fuzzy matching on real data are available in Section 3.1.3.

2.2.2 Chunk Construction

The anchor generation stage produces a list of sentence indices $[a_1, \ldots, a_k]$ from the original document D that the LLM identifies as relevant to the user query Q. However, these anchors alone may not provide enough context for a complete response. To enhance the answer quality and accuracy, we construct chunks around the anchor sentences to provide additional context.

A simple approach is to create chunks by selecting w sentences before and after each anchor:

$$c_j = [s_{\max\{1, a_j - w\}}, \ldots, s_{\min\{a_j + w, n\}}],$$

resulting in an initial list of chunks $\mathbf{c} = [c_1, \ldots, c_k]$. Overlapping chunks in $\mathbf{c}$ are merged to form the final list of contiguous excerpts C (see Algorithm 2). Despite its simplicity, this approach performs well in our experiments. More advanced chunking methods can be used when additional structural information about D is available (see Section 2.4.3).

2.2.3 Final Answer Generation

The final stage of Embedding-Free RAG exactly mimics the answer generation stage of traditional RAG. The list of chunks C is treated in the same way as chunks retrieved by an embedding model would be in a traditional RAG flow. Appendix A.2 shows a sample final answer prompt. Algorithm 1 shows the final generalized Embedding-Free RAG algorithm.

2.3 Comparison with Traditional RAG

Traditional RAG pipelines depend heavily on embedding-based retrieval, where documents are preprocessed into vector representations, stored in a vector database, and retrieved using similarity search. Optimizing these systems requires careful

Algorithm 1 Generalized Embedding-Free RAG

Require: query Q, document D, window size w
Ensure: answer A to Q based on D

▷ Step 1: Generate Reference Quotations
Call LLM with Reference Quotation Generation Prompt using Q and $[s_1, \ldots, s_n]$
▷ Assume the LLM returns a list of reference quotations $[r_1, \ldots, r_m]$
▷ Step 2: Compute Closest Sentence Indices
Compute $[a_1, \ldots, a_m]$ where:
for each reference quotation r_j in $[r_1, \ldots, r_m]$ **do**
    $a_j \leftarrow \arg\min_{i \in [1,\ldots,n]} \text{lev}(r_j, s_i)$ ▷ Find the index of the sentence s_i in D that has the minimum Levenshtein distance to r_j
end for
▷ Step 3: Generate Chunks Around Closest Sentences
Generate $[c_1, \ldots, c_m]$ where:
for each index a_j in $[a_1, \ldots, a_m]$ **do**
    $c_j \leftarrow [s_{\max\{1, a_j - w\}}, \ldots, s_{\min\{a_j + w, n\}}]$ ▷ Create a chunk around the sentence s_{a_j} with a window size w
end for
▷ Step 4: Consolidate Overlapping Chunks
Call Algorithm 2 with $[c_1, \ldots, c_m]$ to get consolidated chunks C
▷ Step 5: Generate Final Answer
Call LLM with Final Answer Prompt using Q and C
return Final answer A generated by LLM

design of chunking strategies, reranker models, and domain-specific fine-tuning — often with significant upfront engineering costs.

Embedding-Free RAG maintains the same high-level retrieve-then-generate flow but replaces the embedding stage entirely. Instead of static similarity search, our framework directly uses an LLM to identify relevant spans on-the-fly, eliminating the need for pre-computation or domain-specific hyperparameter tuning. This makes it uniquely suited for scenarios where documents are introduced at query time and where real-time processing is essential. By removing the dependency on embeddings and offline indexing, our approach offers a lightweight, plug-and-play alternative to traditional pipelines while preserving transparency and grounding.

2.4 Embedding-Free RAG Optimizations

We now discuss variants of Embedding-Free RAG that are optimized for different scenarios.

2.4.1 Parallelized Document Processing

Long-context LLMs are known to be susceptible to the "lost in the middle" effect for large inputs (Liu et al., 2024). This can lead to false negatives for reference quotation generation on large input documents. As a solution, we suggest parallelized document processing during quotation generation by splitting D into smaller subdocuments, which are processed individually before aggregating the anchors and proceeding with chunk generation.

Restricting the LLM's context to subdocuments, however, risks loss of global context. For instance, a query about company performance in a public filing may fail if the filing date appears in a different subdocument. To address this, we prepend a lightweight summary to each subdocument during quotation generation. This summary (2–3 sentences based on the first 5,000 words of the document) is not an extensive document representation but rather a minimal contextual scaffold to enhance and relevance across fragmented subdocuments.

Beyond improving retrieval accuracy, this approach enables substantial efficiency gains. Unlike methods dependent on monolithic structures such as memory trees (Wang et al., 2024b), Embedding-Free RAG decouples information extraction from question answering, enabling parallelization. Parallelization in turn reduces latency, as LLM response time scales linearly with output length (Chen et al., 2024) and typically fewer quotations are produced per subdocument. It also lowers cost when using vendor-hosted models with token-based pricing tiers (Google, 2024).

2.4.2 Using Task-Specific LLMs

While the Embedding-Free RAG framework is model-agnostic, it uses LLMs in two different capacities: (1) for the information extraction task and (2) for the final answer generation. The LLM used in these two steps need not be the same. As the task of identifying relevant information in a passage is easier than any reasoning and understanding of nuance required to synthesize a final answer, we recommend using a faster and less powerful LLM for quotation generation. Beyond reducing cost, using a lightweight LLM during retrieval significantly improves latency, especially when combined with the parallelization described in Section 2.4.1.

2.4.3 Structured Documents

In scenarios where users provide structured documents (e.g., PDFs with hierarchical organization), two key advantages emerge. First, instead of relying on reference quotations as anchors, an LLM can be prompted to return identifiers derived from metadata M, corresponding to specific sections or subsections. Second, the document's hierarchy can inform chunking; by prompting the LLM to generate structure-aware tags, we enable retrieval of coherent units (e.g., pages or subsections) as chunks. To support this, the document text may require preprocessing annotation with structural metadata, allowing the LLM to effectively exploit the document's organization during retrieval and response generation.

2.5 Cost of Embedding-Free RAG

Embedding-Free RAG processes a document exactly once per query during the Retrieval Phase, resulting in input token usage proportional to the document length, plus a small prompt overhead. In the question-answering paradigm, the retrieved subset of the text will be much smaller than the entire document, resulting in $O(\sum_{i=1}^{n} |s_i|)$ input tokens processed by the LLM for the entire pipeline. This is comparable to commercial "Chat with a Document" solutions that rely on large-context LLMs to process entire documents and queries in-context.

While traditional RAG approaches are cheaper per query, they incur significant upfront costs such as designing chunking strategies, tuning retrievers, fine-tuning embeddings and preparing domain-specific datasets. For instance, the CAUD legal dataset took 40 lawyers over a year and cost $2 million to create (Pipitone and Alami, 2024). In contrast, Embedding-Free RAG minimizes setup costs and enables rapid deployment, even in highly specialized or low-resource domains.

3 Empirical Evaluations

Our empirical evaluations focus on domains which prove challenging for modern RAG systems, specifically question answering on legal and financial documents. We use the LegalBench-RAG (Pipitone and Alami, 2024) and FinanceBench (Islam et al., 2023) benchmarks. The queries for both benchmarks directly reference the document to be used, narrowing the scope of RAG from a corpus-wide search to effectively the "Chat with a Document" problem. Unless otherwise specified, our evaluations use the parallelized variant of Embedding-Free RAG described in Section 2.4.1, with Gemini 1.5 Flash as the LLM for both quotation and final answer generation, subdocument size of 3000 words and chunk window size $w = 5$.

LegalBench-RAG. This benchmark prioritizes accurate retrieval by extracting concise, highly relevant text segments from legal documents for 4 datasets: Privacy Question Answering (PrivacyQA) (Ravichander et al., 2019), Contract Understanding Atticus Dataset (CUAD) (Hendrycks et al., 2021), Mergers and Acquisitions Understanding Dataset (MAUD) (Wang et al., 2023) and Contract Natural Language Inference (ContractNLI) (Koreeda and Manning, 2021).

The datasets span a variety of areas in the legal space, with PrivacyQA focusing on privacy policies of consumer apps, CUAD on private contracts, MAUD on merger and acquisition documents of public companies and ContractNLI focusing on NDA-related documents. Each of the nearly 7000 queries in the LegalBench-RAG benchmark is associated to one legal document, with ground-truth annotations created by domain-experts through a rigorous quality controlled process.

FinanceBench. This benchmark evaluates LLMs on financial question answering, covering 40 publicly traded US companies and public filings from 2015 to 2023. We used the open-source evaluation dataset that consists of 150 cases, designed for a fine-grained analysis of question answering capabilities across the following 3 categories:

- **Domain-Relevant:** Expert-written queries targeting commonly analyzed financial indicators (e.g., recent dividend payments, consistency of operating margins).
- **Novel-Generated:** Scenario-based questions crafted by financial experts, tailored to the specific company, report, and industry. Designed to be realistic, varied, and non-trivial.
- **Metrics-Generated:** Questions requiring numerical reasoning and calculations using financial statements (e.g., income statement, balance sheet, cash flow).

3.1 Retrieval Performance

We evaluate the performance of Embedding-Free RAG on the retrieval subtask using the LegalBench-RAG dataset. In the original paper, Pipitone and Alami (2024) evaluated the recall and precision

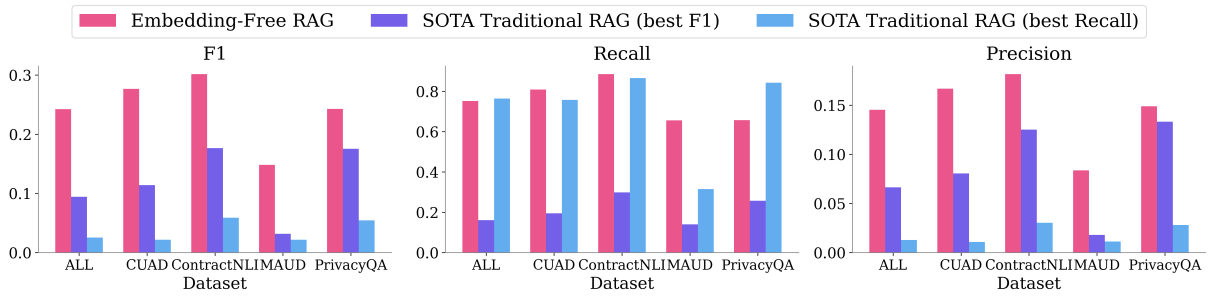

Figure 2: Embedding-Free RAG out-of-the-box significantly outperforms each of the strongest dataset-specific RAG pipelines evaluated in the LegalBench-RAG benchmark (Pipitone and Alami, 2024) on F1 Score. Each SOTA datapoint represents the best-performing configuration by metric from a pool of 28 RAG pipelines, optimized with varied chunking, reranking, and retrieval strategies per dataset. Embedding-Free RAG achieves higher F1 scores across all four legal datasets with an average improvement of 2.6× and up to 4.6× on the most challenging dataset (MAUD).

of various RAG pipelines on the task of retrieving the ground-truth annotation from a document given a query. These pipelines used OpenAI's "text-embedding-3-large" embedding model and included varied chunking strategies, reranking strategies and number of chunks retrieved. Across these variations, the paper evaluated 28 different RAG pipelines on all datasets. We refer to the best of these 28 pipelines as state-of-the-art (SOTA), looking at best overall performance across both F1 and Recall scores, as well as best performance on each of the datasets for the two metrics. No one pipeline of the 28 was the best on more than one dataset.

In our evaluations, we evaluated the precision, recall and F1 score of the chunks generated by Embedding-Free RAG on each query in the LegalBench-RAG benchmark dataset. We show the F1 scores for Embedding-Free RAG using Gemini 1.5 Flash as the reference quotation generation LLM in Figure 2. Embedding-Free RAG outperformed the best overall RAG pipeline of Pipitone and Alami (2024) on F1 score by a factor of 2.6x, as well as for each of the 4 datasets. The largest improvement on performance was with the MAUD dataset, where Embedding-Free RAG achieved an F1 score 4.6x higher than the previous best result. Of the 4 datasets, retrieval on MAUD was the most challenging in the original paper, with the highest recall RAG pipeline only achieving a recall of 0.31 and a precision of 0.01 (Pipitone and Alami, 2024). In comparison, Embedding-Free RAG achieved a recall of 0.66 and a precision of 0.08

3.1.1 Retrieval Across Multiple Documents

Although Embedding-Free RAG is intended for the "Chat with a Document" task, it can still be applied to a more traditional RAG settings where there is a corpus of documents and the correct document is not known a priori. Rather than using just one document in the quotation generation phase for Embedding-Free RAG, all documents can be passed as inputs. We evaluated the performance of Embedding-Free RAG on the Multi-Document scenario for the PrivacyQA dataset. Due to cost concerns, we evaluated the performance of the pipeline on a subset of 10 queries across the entire corpus. The results are shown in Table 1.

Input	Precision	Recall	F1 Score
Single Document	0.1489	0.6571	0.2428
Multi-Document	0.1395	0.6339	0.2287

Table 1: Although not intended for use over an entire corpus due to cost considerations, Embedding-Free RAG still maintains strong performance in the multi-document regime.

The results of Table 1 show that Embedding-Free RAG can be used in more traditional RAG settings, especially when the corpus is small and cost is not an issue.

3.1.2 Impact of Different LLMs on Retrieval

To determine the impact of LLM used for reference quotation generation in the retrieval phase, we conducted an ablation study on the entire PrivacyQA dataset. Our evaluation compared Llama 3.1 (8B and 70B), Qwen 2.5 (7B, 14B and 32B), DeepSeek R1 Distill Qwen (7B, 14B and 32B) and DeepSeek R1 Distill Llama (70B) using the same reference quotation generation prompt. Table 2 shows that on the PrivacyQA dataset Embedding-Free RAG achieves a 2x improvement in recall for 7 out of 9

models for the best-performing Traditional RAG pipeline with respect to the F1 score evaluated by Pipitone and Alami (2024). The smallest Llama 3.1 models still achieved recall twice that of the SOTA Traditional RAG pipeline, showing that smaller Qwen-based models might need custom prompts for the quotation generation task, unlike the out-of-the-box prompt (Appendix A.1) which was used to evaluate all models in the ablation study.

LLM	Precision	Recall	F1 Score
Llama 3.1 (8B)	0.0991	0.6315	0.1714
Llama 3.1 (70B)	0.1709	0.6611	0.2717
Qwen 2.5 (7B)	0.1135	0.3231	0.1680
Qwen 2.5 (14B)	0.1787	0.6684	0.2821
Qwen 2.5 (32B)	0.1473	0.5094	0.2286
DeepSeek R1 Distill Qwen (7B)	0.0670	0.1853	0.0984
DeepSeek R1 Distill Qwen (14B)	0.1136	0.5720	0.1895
DeepSeek R1 Distill Qwen (32B)	0.1397	0.5998	0.2266
DeepSeek R1 Distill Llama (70B)	0.1525	0.7456	0.2533

Table 2: Varying the LLM used for reference quotation generation on the retrieval subtask for the PrivacyQA dataset (Ravichander et al., 2019) demonstrates that the strong performance of the Embedding-Free RAG framework generalizes across different models.

3.1.3 Fuzzy Matching Performance

We conducted a similar study on the entire PrivacyQA dataset to quantify the performance of the fuzzy matching used during the anchor creation stage. Using the RapidFuzz Python library on a 16GB RAM machine, we found the computational cost of fuzzy matching to be negligible. The average time to match one quotation to the corresponding sentence index was 0.005 ± 0.009 seconds, while average space utilization was 34.64 ± 14.01 KB, with peak space utilization 34.83 ± 14.01 KB.

3.2 Comparison with Long-Context LLMs

We compared the performance of Embedding-Free RAG vs long-context LLMs on question answering for MAUD, the most challenging dataset in the LegalBench-RAG benchmark. MAUD presents unique difficulties due to its extensive context length and highly technical content, which requires domain expertise in mergers and acquisitions. Since LegalBench-RAG primarily evaluates the retrieval subtask, it does not provide ground-truth answers for queries. To address this, we generate ground-truth answers with an LLM by supplying the ground-truth context from the dataset.

In our experiment, for each query we generate answers using two methods: (1) long-context LLM answer generation, where we provide the entire document along with the question, and (2) Embedding-Free RAG answer generation, where we pass our retrieved chunks for response generation. For all Embedding-Free RAG variants, we use Gemini 1.5 Flash in the reference quotation generation phase.

To evaluate performance, we employ an LLM oracle (GPT-4o) to compare the generated answers against the ground truth and return either a "Correct", "Incorrect" or "Decline to Answer" label, where the final label corresponds to the scenario where the generator LLM responds that it cannot answer the user query based on the provided context. The reliability of the LLM oracle is validated on a subset of the data, ensuring its assessments align with human-verified ground-truth labels. For answer generation, we experiment with Gemini 1.5 Flash and GPT-4o for final answer generation for both the long-context LLM and Embedding-Free RAG setting.

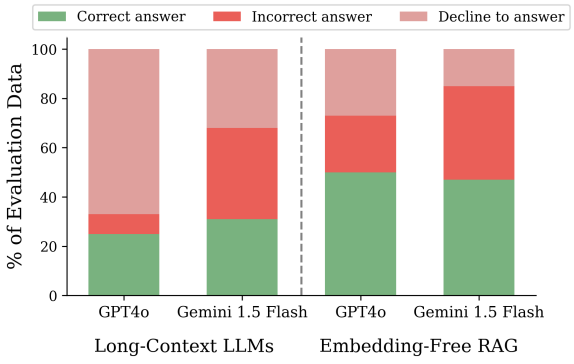

Figure 3: Using the same LLM for answer generation in the Embedding-Free RAG framework compared to passing the entire document into the model's context for question answering improves accuracy up to 2x on the MAUD dataset (Wang et al., 2023). The high rate of "failure to answer" responses from the Long-Context LLMs provides evidence of the models' susceptibilities to positional bias and the "lost in the middle" effect.

Figure 3 shows that Embedding Free-RAG gives a higher percentage of correct answers compared to long-context LLMs for both Gemini and GPT models. Using Embedding-Free RAG provides a 52% and 100% improvement over the long-context

model alone, for Gemini 1.5 Flash and GPT-4o, respectively. Additionally, the long-context LLMs often fail to give an answer, stating that the query cannot be answered with the available information. The failure to recognize that the query can be answered using the text is indicative of positional bias on these large inputs (on average 350,000 characters per document for the MAUD dataset).

3.3 Structured Documents

We next evaluate the performance of Embedding-Free RAG on FinanceBench, specifically using the structured Embedding-Free RAG variant (see Section 2.4.3). In the original paper, Islam et al. (2023) bench-marked sixteen distinct model configurations across five setups and two prompt orders. These configurations tested various approaches to financial question answering, including a naive "closed book" setting (no additional information provided), an unrealistic "oracle" setting (providing an LLM with the correct page from the source document), and three more realistic settings: a single vector store per document, a shared vector store across all documents, and a long-context window approach using the full public filing. We compare our results against the best results from the three pipelines that fit within the "Chat with a Document" paradigm: (1) the single vector store per document, (2) the long-context LLM with access to the entire document, and (3) the oracle setting, which represents RAG with perfect retrieval.

To incorporate metadata about the structured public filing documents into Embedding-Free RAG, we used pages as our atomic unit. When parsing each PDF with standard Python libraries, the text can be easily annotated on the fly with where pages begin and end. During the retrieval phase, we prompted the LLM to return a list of page numbers which contain relevant information. The returned list of pages function as anchors, and we retrieve each specified page and pass all of them to the LLM for the final generation prompt. While Gemini 1.5 Flash was used for reference quotation generation, we used GPT-4 Turbo for the final answer generation LLM, as that was the best-performing model available and evaluated when the benchmark was published. This allows us to directly compare the performance of Embedding-Free RAG with the results of Islam et al. (2023).

Table 3 shows the performance of Embedding-Free RAG on the FinanceBench benchmark, with the best performing methods evaluated by Islam et al. (2023) shown (annotated with asterisks). Embedding-Free RAG excels on this challenging benchmark, outperforming both the best long-context LLM result as well as best traditional RAG result. The performance of the Oracle shows that the performance of GPT-4 Turbo, even with access to the ground-truth retrieved information, still only achieves 85% accuracy. This result demonstrates the limitations of GPT-4 Turbo itself in answering challenging financial questions and contextualizes the 83% accuracy of Embedding-Free RAG.

Technique	Correct Answer	Incorrect Answer	Failed to Answer
Traditional RAG*	50%	11%	39%
Long-Context LLM*	79%	17%	4%
Embedding-Free RAG	82%	14%	4%
Oracle*	85%	15%	0%

Table 3: Embedding-Free RAG outperforms the best traditional RAG and long-context LLM results on the FinanceBench benchmark (Islam et al., 2023), achieving an accuracy close to that of the Oracle (an LLM with access to ground-truth context).

Since LLMs struggle with tabular data (Zhang et al., 2023), we wanted to compare the performance of Embedding-Free RAG on text extracted using a standard Python PDF parsing library with content extracted by Optical Character Recognition (OCR). Extracting text from a document using OCR provides the LLM with more structural information about tables and other objects in the filings. For our OCR engine, we use Amazon Textract (AWS, 2024), which is able to return tabular data within a document in markdown format. The pages are then annotated in the same manner and the same reference quotation generation prompt.

Question Type	Accuracy (w/o OCR)	Accuracy (with OCR)
Domain Relevant	70%	64%
Novel Generated	84%	84%
Metrics Generated	92%	78%

Table 4: Surprisingly, Embedding-Free RAG on content extracted with OCR performed worse than on content extracted without OCR across all question types on the FinanceBench benchmark (Islam et al., 2023).

Table 4 shows the surprising results that reference quotation generation is actually *worse* with tabular information displayed in markdown format. The biggest drop in performance is for the met-

rics generated question type, which heavily relies on parsing and utilizing information in tables. Although our results are only for the performance of Gemini 1.5 Flash on OCR outputs and may be a model-related artifact, evidence in the literature suggests that markdown may not be an optimal representation of tabular data, compared to other formats such as HTML (Sui et al., 2024).

4 Conclusion

Embedding-Free RAG advances RAG by replacing traditional embedding models with an LLM-based, model-agnostic approach. Our one-size-fits-all pipeline removes the need for extensive hyperparameter tuning, cutting deployment costs and complexity. Extensive empirical evaluations on challenging legal and financial datasets demonstrate Embedding-Free RAG's superior performance on retrieval tasks question-answering accuracy. Scalability is enabled through parallelization, supporting efficient processing of large documents. Embedding-Free RAG offers a powerful and adaptable solution for real-time, user-driven question answering on user-provided documents, particularly in specialized domains where domain-specific knowledge is crucial.

5 Limitations

While Embedding-Free RAG performs well on challenging QA tasks, it is not suitable for all use cases. It is ill-suited for summarization, where reference quoting is undefined or intractable, and often requires passing the full document to the LLM—undermining the efficiency benefits. For large-scale applications over fixed document sets, traditional RAG remains more cost-effective long-term due to its lower inference overhead, despite higher initial setup costs. In such settings, alternatives like DocAI and programmatic extraction may be more appropriate. Finally, the additional cost of Embedding-Free RAG might not be necessary for less-challenging, non-technical documents, where long-context LLM inference alone can suffice.

Acknowledgments

We thank Koustuv Dasgupta for his valuable feedback and guidance in refining and positioning this work. We are also grateful to Bing Xiang and Rahul Sharma for their support in bringing this publication to fruition.

References

Adobe. 2024. Adobe document upload. Website.

Toufique Ahmed, Christian Bird, Premkumar Devanbu, and Saikat Chakraborty. 2024. Studying llm performance on closed-and open-source data. *arXiv preprint arXiv:2402.15100*.

Peter Anderson, Mano Vikash Janardhanan, Jason He, Wei Cheng, and Charlie Flanagan. 2024. Greenback bears and fiscal hawks: Finance is a jungle and text embeddings must adapt. *arXiv preprint arXiv:2411.07142*.

AWS. 2024. Amazon textract. Website.

Max Bachmann. Rapidfuzz 3.11.0. Website.

Zechen Bai, Pichao Wang, Tianjun Xiao, Tong He, Zongbo Han, Zheng Zhang, and Mike Zheng Shou. 2024. Hallucination of multimodal large language models: A survey. *arXiv preprint arXiv:2404.18930*.

Laura Caspari, Kanishka Ghosh Dastidar, Saber Zerhoudi, Jelena Mitrovic, and Michael Granitzer. 2024. Beyond benchmarks: Evaluating embedding model similarity for retrieval augmented generation systems. *arXiv preprint arXiv:2407.08275*.

Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

Yanxi Chen, Yaliang Li, Bolin Ding, and Jingren Zhou. 2024. On the design and analysis of llm-based algorithms. *arXiv preprint arXiv:2407.14788*.

Lei Cui, Yiheng Xu, Tengchao Lv, and Furu Wei. 2021. Document ai: Benchmarks, models and applications. *arXiv preprint arXiv:2111.08609*.

Dawid Dahl. 2024. The death of rag: What a 10m token breakthrough means for developers. Website.

Rian Dolphin, Barry Smyth, and Ruihai Dong. 2022. Stock embeddings: Learning distributed representations for financial assets. *arXiv preprint arXiv:2202.08968*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Google. 2024. Pricing models. Website.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review. *arXiv preprint arXiv:2103.06268*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. 2023. Financebench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*.

Qiao Jin, Chuanqi Tan, Mosha Chen, Ming Yan, Ningyu Zhang, Songfang Huang, Xiaozhong Liu, et al. 2022. State-of-the-art evidence retriever for precision medicine: algorithm development and validation. *JMIR Medical Informatics*, 10(12):e40743.

Yuta Koreeda and Christopher D Manning. 2021. Contractnli: A dataset for document-level natural language inference for contracts. *arXiv preprint arXiv:2110.01799*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Dongfang Li, Zetian Sun, Xinshuo Hu, Zhenyu Liu, Ziyang Chen, Baotian Hu, Aiguo Wu, and Min Zhang. 2023. A survey of large language models attribution. *arXiv preprint arXiv:2311.03731*.

Ming Li, Keyu Chen, Ziqian Bi, Ming Liu, Benji Peng, Qian Niu, Junyu Liu, Jinlang Wang, Sen Zhang, Xuanhe Pan, et al. 2024. Surveying the mllm landscape: A meta-review of current surveys. *arXiv preprint arXiv:2409.18991*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*.

Microsoft. 2024. Microsoft document upload. Website.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.

OpenAI. 2024. Introducing openai o1. Website.

PDF.ai. 2024. Pdf.ai document upload. Website.

Sundar Pichai and Demis Hassabis. 2024. Our next-generation model: Gemini 1.5. Website.

Nicholas Pipitone and Ghita Houir Alami. 2024. Legalbench-rag: A benchmark for retrieval-augmented generation in the legal domain. *arXiv preprint arXiv:2408.10343*.

Renyi Qu, Ruixuan Tu, and Forrest Bao. 2024. Is semantic chunking worth the computational cost? *arXiv preprint arXiv:2410.13070*.

NL Rane, M Paramesha, J Rane, and O Kaya. 2024. Emerging trends and future research opportunities in artificial intelligence, machine learning, and deep learning. *Artificial Intelligence and Industry in Society*, 5:2–96.

Abhilasha Ravichander, Alan W Black, Shomir Wilson, Thomas Norton, and Norman Sadeh. 2019. Question answering for privacy policies: Combining computational and legal perspectives. *arXiv preprint arXiv:1911.00841*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Yixuan Tang and Yi Yang. 2024. Do we need domain-specific embedding models? an empirical investigation. *arXiv preprint arXiv:2409.18511*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Runchu Tian, Yanghao Li, Yuepeng Fu, Siyang Deng, Qinyu Luo, Cheng Qian, Shuo Wang, Xin Cong, Zhong Zhang, Yesai Wu, et al. 2024. Distance between relevant information pieces causes bias in long-context llms. *arXiv preprint arXiv:2410.14641*.

Steven H Wang, Antoine Scardigli, Leonard Tang, Wei Chen, Dimitry Levkin, Anya Chen, Spencer Ball, Thomas Woodside, Oliver Zhang, and Dan Hendrycks. 2023. Maud: An expert-annotated legal nlp dataset for merger agreement understanding. *arXiv preprint arXiv:2301.00876*.

Xiaohua Wang, Zhenghua Wang, Xuan Gao, Feiran Zhang, Yixin Wu, Zhibo Xu, Tianyuan Shi, Zhengyuan Wang, Shizheng Li, Qi Qian, et al. 2024a. Searching for best practices in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17716–17736.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li,

Xian Li, Bing Yin, et al. 2024b. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*.

Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Renyu Li. 2024. Financial report chunking for effective retrieval augmented generation. *arXiv preprint arXiv:2402.05131*.

Zheng Yuan, Zhengyun Zhao, Haixia Sun, Jiao Li, Fei Wang, and Sheng Yu. 2022. Coder: Knowledge-infused cross-lingual medical term embedding for term normalization. *Journal of biomedical informatics*, 126:103983.

Han Zhang, Xumeng Wen, Shun Zheng, Wei Xu, and Jiang Bian. 2023. Towards foundation models for learning on tabular data. *arXiv preprint arXiv:2310.07338*.

A Auxiliary Algorithms

A.1 Reference Quotation Generation

```
Your task is to find information related to the Query based on the Document Chunk.
Here is a description of the Document: {doc_desc}

Your response must be a list of relevant quote strings.
The quotes must be exact quotes from the Document that are directly related to the query.
Each exact quote must be a word-for-word quote from the Document.
If there are no related quotes, return an empty list.

Reminder, this is what your response should look like:
["<quote_1>", "<quote_2>", ... , "<quote_n>"]

If there are no relevant quotes, your answer should be an empty list.
Do not return an empty string or None, instead return exactly the following:
'[]'

Query: {query}
Document Chunk: {doc}
Format: {format_instructions}
Output List:
```

A.2 Final Answer Generation

```
Read the following Text and answer the Question directly in 10-15 words.
Please be sure to directly answer the question.
Say "Sorry, the information is not available." if you don't know the answer based on the Text.
Question: {query}
Here is a description of the Document: {doc_desc}
Text: {doc}
Answer:
```

A.3 Merging Overlapping Chunks

Algorithm 2 Merge Overlapping Chunks

Require: $\mathbf{c} = [c_1, c_2, \ldots, c_j]$ where $c_i = [s_{\min\{a_i - w, 1\}}, \ldots, s_{\max\{a_i + w, n\}}]$

$C \leftarrow []$ ▷ Initialize the merged list
for each c_i in **c do**
  if C is empty **then**
    $C \leftarrow [c_i]$
  else
    last_chunk $\leftarrow C[-1]$
    if c_i overlaps with last_chunk **then**
      merged_chunk $\leftarrow [s_{\min(\min\{a_i - w, 1\}, \min\{a_{\text{last}} - w, 1\})}, \ldots, s_{\max(\max\{a_i + w, n\}, \max\{a_{\text{last}} + w, n\})}]$
      $C[-1] \leftarrow$ merged_chunk
    else
      $C \leftarrow C \cup [c_i]$
    end if
  end if
end for
return C