HybridRAG: A Practical LLM-based Chatbot Framework based on Pre-Generated Q&A over Raw Unstructured Documents

Sungmoon Kim[2], Hyuna Jeon[2], Dahye Kim[2], Mingyu Kim[2], Dong-Kyu Chae[2,*] and Jiwoong Kim[1,*]

[1]*Makebot Incorporated, Seoul, South Korea*
[2]*Hanyang University, Seoul, South Korea*

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a powerful approach for grounding Large Language Model (LLM)-based chatbot responses on external knowledge. However, existing RAG studies typically assume well-structured textual sources (*e.g.*, Wikipedia or curated datasets) and perform retrieval and generation at query time, which can limit their applicability in real-world chatbot scenarios. In this paper, we present **HybridRAG**, a novel and practical RAG framework towards more accurate and faster chatbot responses. First, HybridRAG ingests raw, unstructured PDF documents containing complex layouts (text, tables, figures) via Optical Character Recognition (OCR) and layout analysis, and converts them into hierarchical text chunks. Then, it pre-generates a plausible question-answer (QA) knowledge base from the organized chunks using an LLM. At query time, user questions are matched against this QA bank to retrieve *immediate answers* when possible, and only if no suitable QA match is found does our framework fall back to an on-the-fly response generation. Experiments on OHRBench demonstrate that our HybridRAG provides higher answer quality and lower latency compared to a standard RAG baseline. We believe that HybridRAG could be a practical solution for real-world chatbot applications that must handle large volumes of unstructured documents and many users under limited computational resources.

Keywords

Retrieval-augmented generation, QA pre-generation, LLM-based Chatbot

1. Introduction

Large Language Model (LLM)-based chatbots are increasingly being adopted to provide users with convenient, on-demand access to information and services [1]. By interpreting user queries and generating precise answers, AI chatbots can operate 24/7 and reduce the need for manual information lookup or human customer service. A prominent strategy in recent chatbot systems is *Retrieval-Augmented Generation* (RAG), which equips an LLM with an external knowledge base so that it can retrieve relevant context and ground its responses in up-to-date information [2]. RAG has proven effective at reducing factual errors (hallucinations) by letting the model consult factual sources during generation, especially in domain-specific or knowledge-intensive tasks [3].

Despite this progress, deploying RAG in real-world scenarios still faces important challenges. One challenge is that real-world documents are often unstructured and complex: In practice, enterprise knowledge bases include raw, scanned PDFs with figures or graphics, and forms with tables. Such content is not plain text and requires pre-processing like Optical Character Recognition (OCR) to be usable [4]. However, most studies on RAG typically assume that external knowledge is available as clean, structured, textual data, typically curated from well-organized datasets such as Wikipedia or other textual corpora [2]. Another challenge is that current RAG-based chatbots tend to be runtime-intensive: they perform retrieval and LLM inference for each user query. Such on-the-fly generation can lead to significant latency, especially problematic for enterprises operating under resource constraints and

IR-RAG @ SIGIR '25: The Second Edition of the Workshop on Information Retrieval's Role in RAG Systems, July 17, 2025, Padua, Italy
*Corresponding authors.
smk980510@hanyang.ac.kr (S. Kim); younghyuna12@hanyang.ac.kr (H. Jeon); dahye99@hanyang.ac.kr (D. Kim); mingyu0519@hanyang.ac.kr (M. Kim); dongkyu@hanyang.ac.kr (D. Chae); creative@makebot.ai (J. Kim)

needing to serve high volumes of queries promptly. In this case, relying on the LLM alone to generate answers to numerous user queries would increase the response latency, ultimately leading to customer dissatisfaction and potential churn from the service.

In this work, we start with the following question: "*What if we could pre-generate plausible question-answer (QA) pairs from these raw document PDFs and utilize them at query time? And how much do they contribute to improving chatbot response quality and latency?*" To answer this, we propose **HybridRAG**, a practical framework that enjoys the strengths of a pre-generated QA base to enhance the response time and quality. Our framework starts with an offline QA pre-generation from PDFs with an LLM. In order to convert a raw document (i.e., a scanned PDF of documents with non-selectable texts) into structured textual representations suitable for QA generation, we conduct MinerU-based layout analysis [5] and OCR to extract text; for the detected non-textual elements such as tables and figures, we prompt an LLM (*e.g.*, GPT-4o-mini) to generate appropriate textual descriptions for them. Inspired by RAPTOR [6], we then apply a hierarchical chunking to organize the text representations into a tree of chunks ranging from fine-grained (paragraph or section) to coarse-grained (document summary). Finally, an LLM generates a comprehensive and diverse set of plausible QA pairs from these hierarchical chunks. We design prompts with chain-of-thought [7] reasoning and enforce that questions are answerable only using the chunk's content, with no extraneous hallucination. In addition, we extract keywords to generate relevant QA pairs for each chunk node; here, we assign more keywords to higher layers of the hierarchy, which aggregate larger chunks of information, and fewer keywords to lower layers. Finally, these pre-generated QA pairs are indexed by their question embeddings.

At query time, HybridRAG first attempts to retrieve a matching question from the index and directly returns the corresponding answer if a close match is found (above a similarity threshold), without invoking the LLM generation. If no stored QA is a good match for the user's query, HybridRAG uses the LLM to generate an answer on the fly based on the retrieved contexts. In this way, we expect that common or predictable questions can be efficiently and reliably answered, while still retaining the flexibility to handle unexpected queries.

To validate our HybridRAG, we conduct extensive experiments using OHRBench [4], consisting of 1,261 real-world unstructured PDF documents (in total 8,561 pages) spanning 7 domains (*textbook*, *law*, *finance*, *newspaper*, etc.) and 8,498 ground truth QA pairs for benchmarking RAG systems. Our experiments show that HybridRAG achieves much lower average latency (and thus improved user experience) compared to a standard RAG pipeline. At the same time, HybridRAG achieves higher response quality than the baseline with the same LLM, demonstrating that the pre-generated knowledge base effectively aids answer correctness.

2. Related Works

2.1. Unstructured Document Understanding

Most previous RAG studies assume scenarios where refined textual data is directly available [8, 9, 10], which may struggle to effectively handle questions on unstructured documents [3]. Several recent studies try to address such complex documents in the context of OCR [11, 12, 13], but simple OCR-based text extraction cannot sufficiently capture the characteristics of multimodal documents [4]. Consequently, further research has aimed to improve document parsing performance [14, 5, 4]. More recently, several multimodal methods have been proposed to understand complex documents via integration with Vision-Language Models [15, 16, 17, 18]. Our HybridRAG is orthogonal to these approaches; they can be adopted to enhance the quality of our pre-generated QA pairs.

2.2. QA Generation

Early studies on QA generation [19, 20] have the limitation of focusing primarily on QA pairs including short, factoid-style answers. Several methods [21, 22] leverage a Frequently Asked Questions (FAQ) database to swiftly respond to frequent user queries, but the coverage of questions in such FAQ bases is

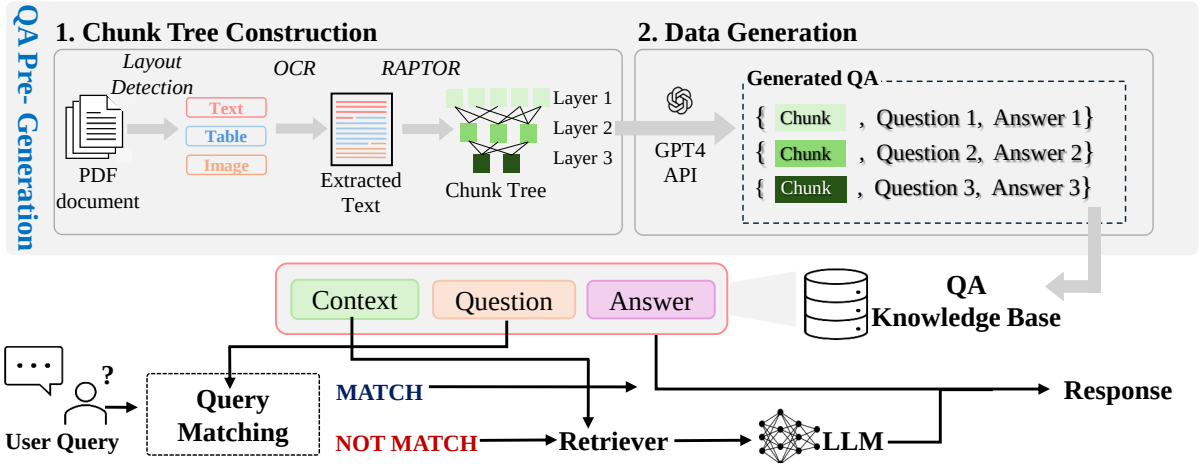


Figure 1: The overview of our HybridRAG.

very limited. Some work [23] fine-tuned LLMs with an intensively curated QA dataset, but both QA data curation and fine-tuning LLMs may not be feasible under resource-limited situations. Different from the prior works, HybridRAG aims to generate numerous high-quality QA pairs by integrating layout analysis, OCR techniques, and hierarchical chunking to manage raw, unstructured PDF documents.

3. Method

The proposed framework consists of an offline QA pre-generation phase and an online query-time phase. Figure 1 illustrates the overview of our framework.

3.1. QA Pre-Generation

3.1.1. Document Preprocessing

Each input PDF document is first processed by the MinerU [5], which segments each page into layouts of type text, table, and image, and outputs their bounding box coordinates in a machine-readable format (*e.g.*, Markdown or JSON). Text blocks within the detected regions are then extracted using PaddleOCR[1]. Non-textual layouts (tables and figures) are converted to textual descriptions via a dedicated LLM. Here, we use GPT-4o-mini with the following prompt to generate descriptive text:

> *You are an image description expert who accurately outputs chunk text based on the information contained in the image. You must analyze the content without omission, avoid duplication or misinterpretation, and write the result as a coherent paragraph without any markdown formatting or external commentary.*

The extracted texts via OCR and generated ones with GPT-4o-mini are then consolidated and structured into chunks for later use. Finally, to maintain contextual coherence among various document elements, we also store their metadata such as page number, coordinates, and sequential order.

3.1.2. Hierarchical Chunking

Inspired by RAPTOR [6], we partitioned the entire document chunks into a hierarchical tree structure; we place compressed and summarized information about the whole document at the top-level node, and progressively include more specific and detailed content in lower-level nodes. Structuring the document into a tree in this way allows effective and flexible responses to user queries, ranging from general context to specific details, thereby enabling comprehensive QA generation.

[1]https://github.com/PaddlePaddle/PaddleOCR

3.1.3. Keyword Extraction

To generate QA pairs that are as relevant as possible, we extract core keywords for each node via GPT-4o-mini. Reflecting the characteristic of the chunk tree structure (higher layers aggregate content from multiple chunks and thus contain richer information), we map a larger number of keywords to upper layers and fewer keywords to lower layers.

3.1.4. QA Generation

To generate QA pairs reflective of realistic user queries, we apply a chain-of-thought-based prompting approach [7]. The prompt is designed to guide the LLM (GPT-4o-mini) through a structured, five-step reasoning process in order to ensure that questions were generated exclusively from explicitly stated information in the text. Additionally, constraints were included to prevent redundancy with previously generated QA pairs. To enhance QA quality, the prompt contained detailed examples comprising the original text, provided keywords, explicit reasoning steps, and resulting QA pairs. Furthermore, the number of QA pairs generated per chunk was set equal to the number of keywords extracted from that chunk, ensuring comprehensive coverage of diverse key information. The detailed prompt is as follows:

<Prompt used for QA generation>

You are an AI specialized in generating QAs from documents. Your mission is to analyze the document, follow the instructions, and generate RAG-style question-answer pairs based on the document. RAG-style refers to a question that needs to be answered by retrieving relevant context from an external document based on the question, so the question MUST obey the following criteria:

1. Question should represent a plausible inquiry that a person (who has not seen the page) might ask about the information uniquely presented on this page. The questions should not reference this specific page directly (by page number, pointing to a specific paragraph or figure, and never refer to the document using phrases like 'in the document'), nor should they quote the text verbatim. They should use natural language reflecting how someone might inquire about the page's content without direct access.
2. Question must contain all information and context/background necessary to answer without the document. Do not include phrases like "according to the document" in the question.
3. Question must not contain any ambiguous references, such as 'he', 'she', 'it', 'the report', 'the paper', and 'the document'. You MUST use their complete names.

<System prompt used for QA generation>

- Instruction:
1. Analyze the text above and the given keywords.
2. Create new, meaningful question–answer pairs that a user might naturally ask about this text.
3. Do not duplicate any previously generated Q&A.
4. Only generate questions that can be answered explicitly by the text.
5. Provide concise, direct answers without extra elaboration.

The answer form should be as diverse as possible, including [Yes/No, Numeric, String, List].

- Output format:
Question:
Answer:

<Few-shot example>

text:
"With direct access to human-written reference as memory, retrieval-augmented generation has achieved much progress in various text generation tasks. In this framework, better memory typically leads to better generation (the 'primal problem'). The proposed Selfmem approach leverages an iterative memory to improve performance."
keywords: ["retrieval-augmented generation", "primal problem", "Selfmem"]
(Reasoning steps)
1. Identify main content: retrieval-augmented approach, concept of 'better memory => better generation'
2. Form a question about what tasks or idea the snippet highlights
3. The answer must be directly found in the snippet
Question: What core idea does the Selfmem framework build upon?
Answer: It uses an iterative memory mechanism, where better generation leads to better memory, improving overall performance.

3.2. HybridRAG

We embed the questions from the pre-generated QAs using the BGE-M3 (dense retrieval) [24]. At query time, a given user query is embedded using the same model, and similarity scores are calculated via the inner product between embeddings. The top-3 QA pairs are then retrieved, and if the highest similarity score exceeds a predefined threshold (*e.g.*, 0.9), the answer from the most similar QA pair is directly returned. However, if the similarity score is below the threshold, the chunks corresponding to the retrieved top-3 QA pairs are aggregated and provided along with the user query as input to an LLM to generate the final response. We used Llama3.2-3B-Instruct [25] and Qwen2.5-3B-Instruct [26] as the generative LLMs for generating on-the-fly responses.

<User prompt used for RAG>

You are an expert, you have been provided with a question and documents retrieved based on that question. Your task is to search the content and answer these questions using both the retrieved information.
*You **MUST** answer the questions briefly with one or two words or very short sentences, devoid of additional elaborations.*
Write the answers within <response></response>. If you cannot find answer from retrieved Documents, say: "Not answerable".

4. Experiments

4.1. Experimental Settings

4.1.1. Benchmark

We employed OHRBench curated by the authors of [4], including 1,261 PDFs spanning 8,561 pages. These PDFs were obtained from seven real-world RAG application domains, including Finance, Administration, Law, etc. Furthermore, this benchmark provides a diverse set of ground truth QA pairs derived from multimodal elements in those documents. Table 1 shows the statistics of OHRBench.

4.1.2. Metrics

In terms of response quality, we measure how well the answers generated by a RAG system match the ground truth answers provided in OHRBench (the QA pairs), using the following three metrics: **F1-score** measures the overlap between the model response and the ground truth answer based on word tokens; **ROUGE-L** evaluates the Longest Common Subsequence (LCS) between the model's answer

Table 1
Benchmark statistics.

Domains	# PDFs	# Pages	# Q&A pairs
Law	95	1,187	1,142
Finance	65	2,133	1,367
Textbook	504	678	1,125
Manual	87	1,724	1,151
Newspaper	279	487	1,202
Academic	85	1,011	1,179
Administration	146	1,341	1,332
Total	1,261	8,561	8,498

and the ground truth answer; and **BERTScore** utilizes contextual embeddings from BERT to measure semantic similarity between the generated answer and the ground truth answer. For response **latency**, we measure the average time interval from receiving a user query to completing the response. We used a server equipped with an NVIDIA RTX 3090 GPU.

4.1.3. Baselines

In order to highlight the effectiveness of our method, we implement two baselines: (1) **Standard RAG** follows the original RAG pipeline without any pre-generated QAs. It only adopts MinerU-based OCR [5] to recognize texts in the given PDFs, without considering figures or tables. The chunk size is set to 1024 tokens without overlap, following [4]. Top-3 most similar text chunks are retrieved to aid LLM response generation. (2) **Simplified HybridRAG** involves pre-generating QA pairs, but it only considers plain text extracted by MinerU-based OCR (identical to the Standard RAG setup). Other aspects, such as embedding models or LLMs, are identical across all three compared models.

4.2. Response Quality and Latency

Table 2 compares the performance of a standard RAG, HybridRAG and its simplified version in terms of response quality and latency. Overall, the simplified HybridRAG outperforms Standard RAG, which demonstrates that the pre-generated QA pairs (even though they were based solely on texts in the documents) have a positive effect on both latency and response quality. With Llama3.2, latency is especially improved, and with Qwen2.5, there is a noticeable answer quality gain. In addition, HybridRAG (Ours)—which incorporates layout detection, OCR, LLM-driven table/figure description, and hierarchical chunking—further boosts answer quality over Simplified HybridRAG. Especially, the domains with a high proportion of chart/table–based queries in OHRBench (Administration: 562, Finance: 1,038 QA pairs) show the largest F1 gains from our HybridRAG: The Administration domain improves by 19% with Qwen2.5, and Finance by 22% compared to Simplified HybridRAG.

4.3. Quality of Pre-generated QAs

This section aims to evaluate the quality of our pre-generated QA pairs. Since there are no human-crafted references or ground truth, we adopt the LLM-judge approach, specifically G-Eval [27]. We assess the quality of QA pairs across the following four perspectives proposed by QGEval [28]: **Context-Question-Answer Relevance (CQAR)** assesses how well the generated question and answer align contextually with the source document; **Answerability** measures the extent to which the generated question is answerable given the provided context; **Clarity** evaluates the preciseness and unambiguity of the generated QAs; **Fluency** reflects the grammatical correctness and naturalness of the generated text. For comparison, the QA pairs provided by OHRBench, which were oriented from the same PDF sources as ours, are also evaluated in the same way. Approximately 5,000 QA pairs were sampled from each set and evaluated via G-Eval.

Table 2
Performance comparison w.r.t. response quality and latency. Best performance is in **bold**; second-best is underlined.

		Standard RAG				HybridRAG (simplified)			
LLM	Domain	F1(↑)	BERT(↑)	ROUGE(↑)	Latency(↓)	F1(↑)	BERT(↑)	ROUGE(↑)	Latency(↓)
Llama3.2	Academic	21.27	0.7521	0.2671	0.899s	20.10	0.7442	0.2510	0.699s
	Administ.	25.09	0.7678	0.2894	1.046s	24.56	0.7608	0.2774	0.747s
	Finance	15.26	0.7133	0.1795	1.185s	14.82	0.7106	0.1780	0.774s
	Law	31.62	0.7805	0.3572	0.889s	30.99	0.7776	0.3370	0.572s
	Manual	31.07	0.7896	0.3476	0.996s	27.82	0.7761	0.3135	0.694s
	Newspaper	20.91	0.8126	0.1938	5.021s	27.59	0.8346	0.2201	1.236s
	Textbook	17.50	0.7561	0.2662	1.760s	19.34	0.7692	0.2825	0.772s
	Avg.	23.25	0.7674	**0.2715**	1.685s	<u>23.60</u>	<u>0.7676</u>	0.2656	**0.785s**
Qwen2.5	Academic	14.14	0.7069	0.1704	0.403s	16.10	0.7137	0.1887	0.362s
	Administ.	14.86	0.7054	0.1723	0.409s	18.25	0.7203	0.2049	0.372s
	Finance	7.71	0.6584	0.0816	0.331s	10.08	0.6748	0.1174	0.426s
	Law	19.42	0.7394	0.1985	0.325s	23.03	0.7490	0.2415	0.287s
	Manual	21.53	0.7396	0.2295	0.396s	22.80	0.7438	0.2448	0.318s
	Newspaper	17.25	0.7133	0.1288	0.490s	25.22	0.7658	0.1844	0.495s
	Textbook	12.37	0.6947	0.1571	0.458s	15.96	0.7114	0.1957	0.354s
	Avg.	15.33	0.7082	0.1626	0.402s	<u>18.78</u>	**0.7255**	<u>0.1968</u>	**0.373s**

		HybridRAG (Ours)			
LLM	Domain	F1(↑)	BERT(↑)	ROUGE(↑)	Latency(↓)
Llama3.2	Academic	20.87	0.7417	0.2459	0.843s
	Administ.	26.31	0.7640	0.2954	0.842s
	Finance	17.51	0.7134	0.2034	1.085s
	Law	31.14	0.7793	0.3471	0.562s
	Manual	28.48	0.7747	0.3201	0.985s
	Newspaper	26.09	0.8310	0.2108	1.338s
	Textbook	20.12	0.7527	0.2596	0.866s
	Avg.	**24.36**	**0.7692**	<u>0.2689</u>	<u>0.931s</u>
Qwen2.5	Academic	16.85	0.7127	0.1894	0.390s
	Administ.	21.80	0.7307	0.2359	0.412s
	Finance	12.31	0.6826	0.1344	0.403s
	Law	25.34	0.7553	0.2628	0.315s
	Manual	23.15	0.7462	0.2526	0.372s
	Newspaper	22.78	0.7557	0.1728	0.455s
	Textbook	13.70	0.6934	0.1395	0.295s
	Avg.	**19.42**	<u>0.7252</u>	**0.1982**	<u>0.377s</u>

Figure 2 shows the results. Compared to the QA pairs provided by OHRBench, those generated by our method demonstrated superior quality w.r.t. all four dimensions: CQAR (+0.24), answerability (+0.31), clarity (+0.50), and fluency (+0.76).

4.4. Threshold Analysis

Lowering the threshold increases the proportion of responses directly returned from the pre-generated QA base. Specifically, at a threshold of 0.9, stored responses handle 13% of the total test queries; however, at a threshold of 0.7, this proportion rises significantly to 80%. Consequently, as shown in Figure 3, while this reduces the average latency further, it introduces a trade-off by simultaneously decreasing the quality of responses.

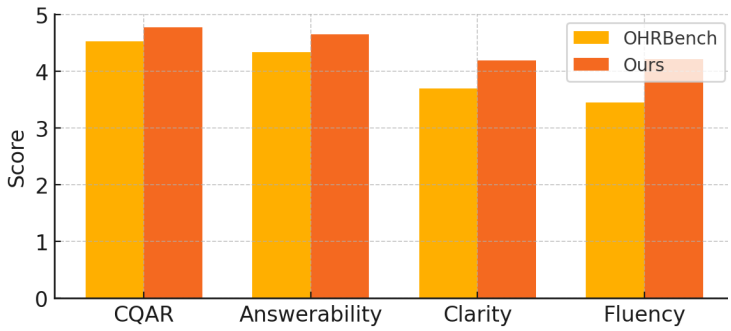
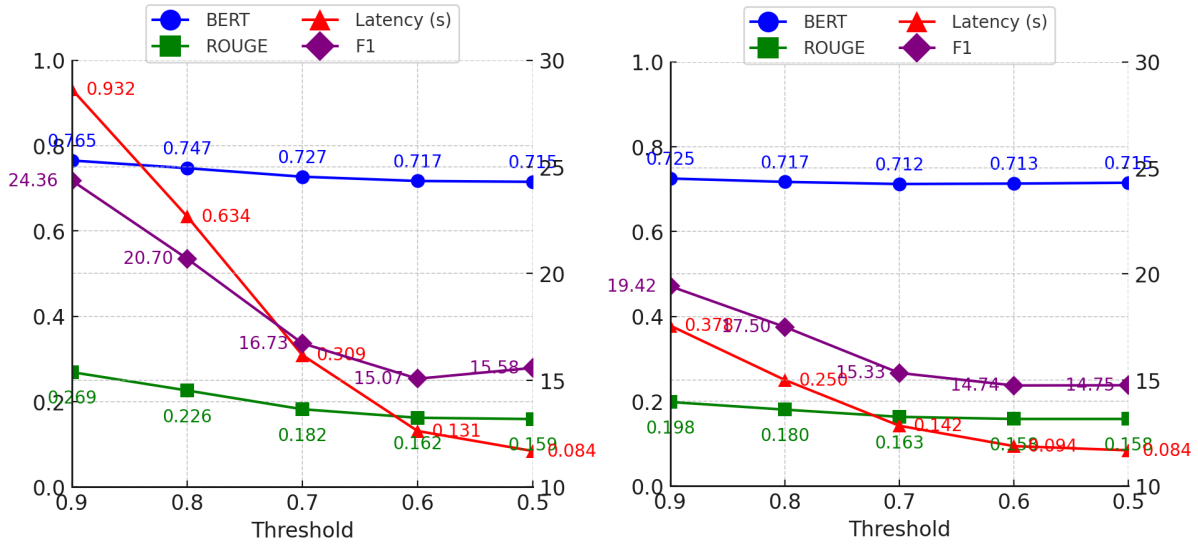


Figure 2: Quality evaluation of the pre-generated QAs.



(a) Llama3.2-3B-Instruct

(b) QWEN2.5-3B-Instruct

Figure 3: Performance depending on different thresholds.

5. Limitation

Despite its benefits, HybridRAG incurs one-time offline overhead costs during the QA pre-generation. Generating the QA base of approximately 130,000 pairs demands computational resources and API expenses summarized in Table 3. However, these upfront costs can be justified given the substantial gains in query time performance and response quality. We believe these gains would be more valuable in real-world chatbot use cases which need to address large amounts of unstructured documents and numerous users with limited GPU resources.

6. Conclusion

We introduced HybridRAG, a practical RAG framework optimized for real-world chatbot scenarios that need to provide low-latency and address unstructured scanned PDF documents. HybridRAG analyzes PDF documents via layout detection, OCR, hierarchical chunking, and dedicated LLM-based description generation for visual elements, and then generates a rich repository of plausible QA pairs. At query time, user queries are matched against the pre-generated QA pairs via embedding similarity; if a sufficiently similar question is found, the corresponding answer is directly returned; if not, an LLM generates responses based on the retrieved contexts. Validated through extensive experiments on OHRBench, HybridRAG achieves higher answer quality (with improved F1, ROUGE-L, and BERTScore) and lower latency compared to the standard RAG baseline.

Table 3
Time (min) and cost ($) requirements for QA generation.

	# of QA	Layout analysis	Description generation	Chunk hierarchy	QA generation	API cost
Law	16,134	237m	258m	159m	460m	$2.91
Finance	40,226	2,191m	644m	565m	1,151m	$7.74
Textbook	5,886	326m	247m	26m	1,104m	$0.96
Manual	17,477	531m	280m	204m	1,884m	$3.16
Newspaper	24,931	220m	431m	118m	486m	$4.25
Academic	14,991	480m	245m	152m	448m	$2.72
Administ.	15,568	616m	249m	162m	905m	$2.76

Acknowledgments

This work was fully supported by **Makebot Incorporated**. This research was conducted based on the company's original ideas and patented technologies, with financial support provided by the company.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT in order to (1) check grammar and spelling and (2) generate the graphs and charts for Figures 2 and 3. After using ChatGPT, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. K. Dam, C. S. Hong, Y. Qiao, C. Zhang, A complete survey on llm-based ai chatbots, arXiv:2406.16937 (2024).

[2] J. Sánchez Cuadrado, S. Pérez-Soler, E. Guerra, J. De Lara, Automating the development of task-oriented llm-based chatbots, in: Proceedings of the 6th ACM Conference on Conversational User Interfaces, 2024, pp. 1–10.

[3] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, arXiv:2312.10997 (2023).

[4] J. Zhang, Q. Zhang, B. Wang, L. Ouyang, Z. Wen, Y. Li, K.-H. Chow, C. He, W. Zhang, Ocr hinders rag: Evaluating the cascading impact of ocr on retrieval-augmented generation, in: ICCV, 2025, pp. 17443–17453.

[5] B. Wang, C. Xu, X. Zhao, L. Ouyang, F. Wu, Z. Zhao, R. Xu, K. Liu, Y. Qu, F. Shang, et al., Mineru: An open-source solution for precise document content extraction, arXiv:2409.18839 (2024).

[6] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, C. D. Manning, RAPTOR: Recursive abstractive processing for tree-organized retrieval, in: The Twelfth International Conference on Learning Representations, 2024.

[7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, in: Advances in neural information processing systems, 2022, pp. 24824–24837.

[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in neural information processing systems 33 (2020) 9459–9474.

[9] S. Siriwardhana, R. Weerasekera, E. Wen, S. Nanayakkara, Fine-tune the entire rag architecture (including dpr retriever) for question-answering, arXiv:2106.11517 (2021).

[10] A. Misrahi, N. Chirkova, M. Louis, V. Nikoulina, Adapting large language models for multi-domain retrieval-augmented-generation, arXiv:2504.02411 (2025).

[11] L. Blecher, G. Cucurull, T. Scialom, R. Stojnic, Nougat: Neural optical understanding for academic documents, in: The Twelfth International Conference on Learning Representations, 2024.
[12] C. Liu, H. Wei, J. Chen, L. Kong, Z. Ge, Z. Zhu, L. Zhao, J. Sun, C. Han, X. Zhang, Focus anywhere for fine-grained multi-page document understanding, arXiv:2405.14295 (2024).
[13] H. Wei, C. Liu, J. Chen, J. Wang, L. Kong, Y. Xu, Z. Ge, L. Zhao, J. Sun, Y. Peng, et al., General ocr theory: Towards ocr-2.0 via a unified end-to-end model, arXiv:2409.01704 (2024).
[14] H. Chao, J. Fan, Layout and content extraction for pdf documents, in: Document Analysis Systems VI: 6th International Workshop, DAS 2004, Florence, Italy, September 8-10, 2004. Proceedings 6, 2004, pp. 213–224.
[15] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, et al., Qwen2. 5-vl technical report, arXiv:2502.13923 (2025).
[16] Z. Chen, W. Wang, Y. Cao, Y. Liu, Z. Gao, E. Cui, J. Zhu, S. Ye, H. Tian, Z. Liu, et al., Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling, arXiv:2412.05271 (2024).
[17] Z. Chen, W. Wang, H. Tian, S. Ye, Z. Gao, E. Cui, W. Tong, K. Hu, J. Luo, Z. Ma, et al., How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites, Science China Information Sciences 67 (2024) 220101.
[18] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, et al., Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution, arXiv:2409.12191 (2024).
[19] P. Lewis, Y. Wu, L. Liu, P. Minervini, H. Küttler, A. Piktus, P. Stenetorp, S. Riedel, Paq: 65 million probably-asked questions and what you can do with them, Transactions of the Association for Computational Linguistics 9 (2021) 1098–1115.
[20] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, Journal of machine learning research 21 (2020) 1–67.
[21] L. S. Nguyen, T. T. Quan, Urag: Implementing a unified hybrid rag for precise answers in university admission chatbots–a case study at hcmut, in: International Symposium on Information and Communication Technology, 2024, pp. 82–93.
[22] G. Agrawal, S. Gummuluri, C. Spera, Beyond-rag: Question identification and answer generation in real-time conversations, arXiv:2410.10136 (2024).
[23] A. Afzal, A. Kowsik, R. Fani, F. Matthes, Towards optimizing and evaluating a retrieval augmented qa chatbot using llms with human-in-the-loop, in: Proceedings of the Fifth Workshop on Data Science with Human-in-the-Loop (DaSH 2024), 2024, pp. 4–16.
[24] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, Z. Liu, M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, in: Findings of the Association for Computational Linguistics ACL 2024, 2024, pp. 2318–2335.
[25] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al., The llama 3 herd of models, arXiv:2407.21783 (2024).
[26] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, et al., Qwen2. 5 technical report, arXiv:2412.15115 (2024).
[27] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, C. Zhu, G-eval: Nlg evaluation using gpt-4 with better human alignment, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023, pp. 2511–2522.
[28] W. Fu, B. Wei, J. Hu, Z. Cai, J. Liu, Qgeval: Benchmarking multi-dimensional evaluation for question generation, in: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2024, pp. 11783–11803.