Enhancing Access to Government Documents: A Retrieval-Augmented Generation Framework for Quezon City Ordinances and Resolutions

Alister Marc B. Domilies
Adviser: Miguel Francisco Remolona PhD

1 Introduction

Large language models (LLMs) excel at a wide range of language tasks, but they have inherent limitations when it comes to recalling factual, up-to-date information. Their knowledge is fixed at the time of training, and they can sometimes generate convincing but incorrect or "hallucinated" answers. Retrieval-Augmented Generation (RAG) addresses these challenges by combining an LLM with an external knowledge base. Instead of relying solely on the model's internal knowledge, RAG first retrieves relevant documents based on the user query and then feeds this information as context to the LLM, improving answer accuracy and providing clear source citations.

In this project, we apply the RAG framework specifically to Quezon City government documents. Residents and government staff often need to reference detailed criteria or regulations contained within official ordinances and resolutions. Searching through large PDF archives manually is slow and inefficient, while keyword search is limited and prone to missing relevant information without exact terms. Our goal is to create a chatbot that understands natural language questions and returns precise, reliable answers grounded directly in official documents.

Through this work, we demonstrate how RAG can enhance public access to government information by combining the strengths of retrieval and generation. The system also offers modularity—components such as the document corpus or language model can be updated independently, allowing for ongoing improvements without retraining the entire system.

2 System Architecture

The system is built on a modular Retrieval-Augmented Generation (RAG) architecture, enabling users to interact with Quezon City's legal documents using natural language. Queries are processed through a pipeline of interconnected modules that classify, transform, retrieve, generate, and validate responses. This structure ensures each answer is both context-aware and grounded in authoritative sources.

When a user submits a query via the frontend, it is sent to the Django backend through the POST /api/chat/ endpoint. The backend then passes the query through the following five key modules:

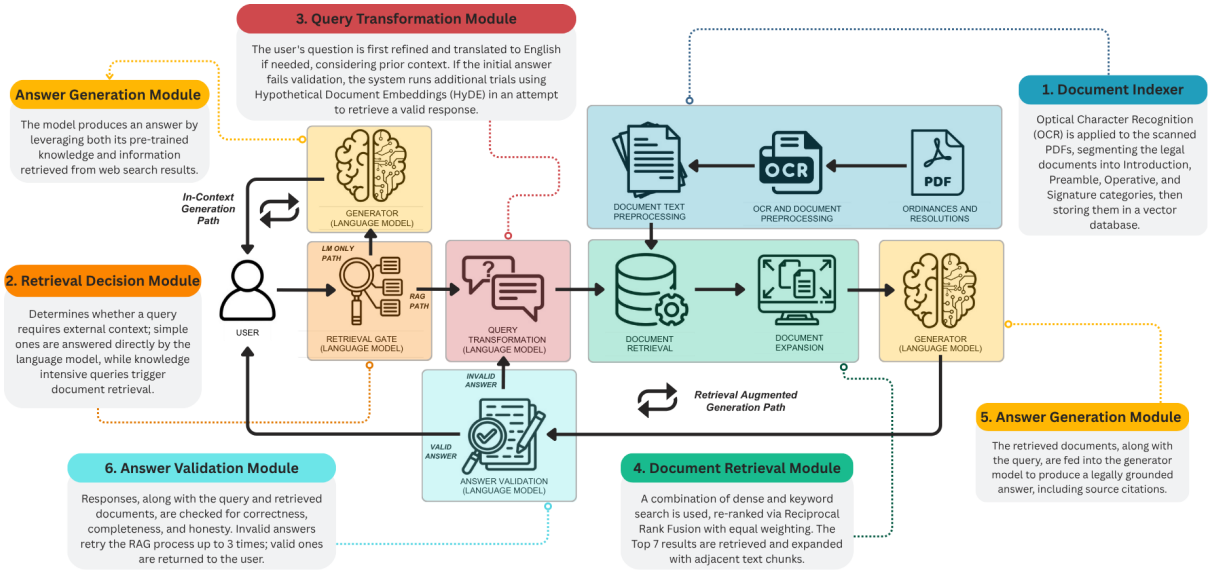

Figure 1: Architecture of RAG System

2.1 Retrieval Decision Module

This module determines whether a query requires document retrieval. For simple or conversational inputs—like greetings or clarification questions—the system may bypass retrieval and directly use the language model to generate an answer. However, for knowledge-intensive queries involving legal concepts or ordinances, retrieval is triggered to ensure grounded and accurate responses.

2.2 Query Transformation Module

The user's input is refined here to improve search relevance. The module may rephrase vague or informal questions, translate them into English when needed, and incorporate recent chat context for clarity. If the system fails to generate a valid answer on the first attempt, it initiates additional retrieval trials using Hypothetical Document Embeddings (HyDE). This technique creates pseudo-documents from the query itself to expand the semantic space and improve retrieval outcomes.

2.3 Document Retrieval Module

Once a refined query is available, this module connects to the Weaviate vector store to perform hybrid retrieval. It uses a combination of dense semantic search (via vector embeddings) and keyword search, re-ranked using Reciprocal Rank Fusion (RRF) with equal weights. The top 7 relevant passages are selected, and each result is expanded by including neighboring chunks from the same document to preserve context.

2.4 Answer Generation Module

The retrieved passages, along with the query and recent chat history, are compiled into a prompt and sent to the OpenAI API (using GPT-4o-mini). The language model generates a legally grounded response, explicitly citing the source documents it relied on. The output is structured as a JSON object containing the answer and a list of referenced titles (e.g., "sources": ["SP-3174, S-2023"]).

2.5 Answer Validation Module

Before returning the response to the user, this module performs a thorough validation. It checks whether the generated answer is factually correct, complete based on the retrieved content, and free from hallucination. If the answer is flagged as invalid, the system retries the RAG pipeline—up to three times—using different query transformations and retrieval variations. Only validated answers are returned to the user to ensure transparency and reliability.

2.6 Data Ingestion and Indexing

A vital component of the system is its searchable knowledge base, built from Quezon City ordinances and resolutions. These documents are ingested through the POST /api/upload/ endpoint, which accepts scanned PDFs. Once uploaded, files are stored in an S3 bucket and processed asynchronously using AWS Textract. Textract extracts both raw text and layout data, effectively handling lists and table-heavy formats.

After extraction, the backend preprocesses the document sections—specifically the Introduction, Preamble, Operative, and Signature—breaking them down further into smaller fragments, each up to 512 tokens as needed. Each fragment is then converted into a high-dimensional vector using the BAAI/bge-small-en embedding model. These vectors are stored in Weaviate, a vector database that supports both cosine similarity semantic search and traditional keyword search. To enhance retrieval accuracy, the results are subsequently re-ranked using Reciprocal Rank Fusion (RRF).

2.7 Technology Stack and Tools

The backend is developed in Python 3, using Django as the primary framework and Django REST Framework to expose endpoints for chat and upload. AWS Textract is integrated via the Boto3 SDK, while Weaviate serves as the vector database for document search. The embedding model is downloaded from Hugging Face and used locally for efficiency.

For response generation, the system uses OpenAI's GPT-4o-mini via the chat completion API. The frontend is built in Streamlit, providing a simple and intuitive chat interface that maintains conversation history and communicates with the backend APIs in real time. All required libraries and dependencies are listed in requirements.txt, and deployment is managed via Docker Compose. This includes services for Django, Weaviate, and optional volume mounting for preloaded vector data.

3 Results and Discussion

To evaluate our RAG chatbot system, we first generated a set of 150 synthetic question-answer pairs in English using GPT-4o. These pairs were also translated into Tagalog to enable evaluation across both English and Filipino queries, reflecting the bilingual context of Quezon City.

3.1 Embedding Model Evaluation

For the retrieval component, we compared three embedding models known for strong performance in legal domains, selected from the MTEB benchmark on Hugging Face. These were: `bge-base-en-v1.5` (109 million parameters), `snowflake-arctic-embed-m` (109 million parameters), and `multilingual-e5-large-instruct` (560 million parameters). Among these, the `bge-base-en-v1.5` model consistently yielded the best retrieval accuracy, demonstrating that a well-tuned English embedding model outperforms larger multilingual models for our domain.

Table 1: Embedding model performance across different top-k values.

Model	k	Recall	Precision	MRR
bge-base-en-v1.5	3	88.0%	30.7%	70.8%
	5	92.7%	19.9%	71.8%
	7	95.3%	14.6%	72.2%
snowflake-arctic-embed-m	3	71.3%	24.0%	46.6%
	5	80.7%	16.3%	48.7%
	7	88.0%	12.9%	49.8%
multilingual-e5-large-instruct	3	84.0%	29.1%	66.8%
	5	90.0%	18.9%	68.1%
	7	92.7%	13.9%	68.6%

As shown in Table 1, bge-base-en-v1.5 outperformed the other two models in all metrics across top-k values.

We further investigated how to handle Tagalog queries. We compared two approaches: directly using the multilingual-e5-large-instruct embedding model versus translating Tagalog queries into English and then using the bge-base-en-v1.5 model.

Table 2: Comparison of Tagalog query processing approaches.

Approach	Precision	Recall	MRR
Translation + bge-base-en-v1.5	94.0%	14.2%	70.1%
Multilingual-e5-large-instruct	72.7%	10.9%	46.2%

As seen in Table 2, the translation approach outperforms the direct multilingual embedding strategy in all three metrics.

3.2 Hybrid Search Optimization

The retrieval system combines semantic vector search and traditional keyword search, balanced by an alpha parameter. We tested alpha values of 0.3, 0.5, and 0.7.

Table 3: Retrieval performance by alpha parameter value.

Alpha	Recall	Precision	MRR
0.3	94.0%	14.5%	71.3%
0.5	95.3%	14.6%	72.2%
0.7	95.3%	14.6%	70.3%

Table 3 shows that an equal weighting (alpha = 0.5) achieves the highest MRR, recall and precision.

3.3 Query Transformation

We evaluated GPT-4o-mini, GPT-4o, and Llama 3.3 70B for query refinement and HyDE.

Table 4: Performance of different language models with query transformation techniques.

Model Used	Strategy	Recall	Precision	MRR
gpt-4o-mini	Refined Query	94.7%	14.2%	71.5%
	HyDE	94.0%	14.1%	69.4%
gpt-4o	Refined Query	94.7%	14.1%	71.5%
	HyDE	94.7%	14.1%	71.5%
Llama 3.3 70B	Refined Query	91.3%	13.1%	70.1%
	HyDE	91.0%	13.0%	70.9%

Table 4 shows that gpt-4o-mini offers performance comparable to the full gpt-4o model.

3.4 Impact of the RAG Loop

We implemented a validation loop that retries answer generation up to three times if initial outputs fail consistency checks.

Table 5: Validation success rate across retries in the RAG loop.

Retry Attempt	gpt-4o (eng)	gpt-4o-mini (eng)	gpt-4o (fil)	gpt-4o-mini (fil)
1	92.0	95.0	90.0	89.0
2	95.3	99.0	95.3	95.0
3	97.3	99.0	97.3	97.0

As seen in Table 5, the success rate improves with each retry, confirming the effectiveness of our looped validation mechanism.

3.5 Final Answer Quality Evaluation

We used RAGAS metrics to assess answer faithfulness, correctness, relevance, and similarity.

Table 6: Final answer quality evaluation using RAGAS metrics for different languages and models.

Language	Model (Setup)	Faithfulness	Answer Relevancy	Answer Similarity	Answer Correctness
English (Direct LM Answers)	gpt-4o-mini	–	–	0.932	0.602
	gpt-4o-mini (web search)	–	–	0.930	0.589
English (Our RAG System)	gpt-4o-mini	0.892	0.945	0.958	0.786
	gpt-4o	0.922	0.953	0.955	0.779
Tagalog (Our RAG System)	gpt-4o-mini	0.865	0.937	0.951	0.741
	gpt-4o	0.921	0.952	0.956	0.766

Table 6 demonstrates that RAG-enhanced answers outperform direct LLM generations in both English and Tagalog on all quality metrics.

4 Conclusion

This project demonstrates how Retrieval-Augmented Generation (RAG) can be effectively applied to improve access to Quezon City government documents. By combining document retrieval with large language models, we built a chatbot that provides accurate, source-grounded answers to user queries in both English and Filipino. Through careful design choices, module

evaluation, and iterative validation, the system achieves reliable performance and shows strong potential for real-world deployment in government services.

References

[1] Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q., Yin, R., Lv, C., Zheng, X., & Huang, X. (2024). Searching for Best Practices in Retrieval-Augmented Generation. *arXiv*.

[2] Gao, L., Ma, X., Lin, J., & Callan, J. (2022). Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*. https://arxiv.org/abs/2212.10496

[3] Manjee, S. (2024, June 11). Intelligent RAG data chunking: Fetch surrounding chunks. *Elastic Search Labs*. https://www.elastic.co/search-labs/blog/advanced-chunking-fetch-surrounding-chunks

[4] Kabir, M. R., Sultan, R. M., Rahman, F., Amin, M. R., Momen, S., Mohammed, N., & Rahman, S. (2025). LegalRAG: A hybrid RAG system for multilingual legal information retrieval. *arXiv*. https://arxiv.org/abs/2504.16121

[5] Wang, Y., Garcia Hernandez, A., Kyslyi, R., & Kersting, N. (2024). Evaluating quality of answers for retrieval-augmented generation: A strong LLM is all you need. *arXiv*. https://arxiv.org/abs/2406.18064

[6] Quezon City Government. Public Notices. https://quezoncity.gov.ph/public-notices/

[7] Capstone Poster. https://www.canva.com/design/DAGoVfAhxeY/4YqdcQ7H-Cv0xubdRkJGAA/view?utm_content=DAGoVfAhxeY&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utlId=h75e60b8482

[8] Capstone Presentation. https://www.canva.com/design/DAGoa6AiVrc/gvWyIb2bF2qqFr6R3Gd0gA/view?utm_content=DAGoa6AiVrc&utm_campaign=designshare&utm_medium=link2&utm_source=uniquelinks&utlId=hd17869486a

[9] Github and Documentation. https://github.com/alistermarc/Capstone-Project