# Retrieval Augmented Generation (RAG) Challenge

Javier Dastas and Jarian Del Valle

February 7, 2025

IronHack Puerto Rico

# Contents

# Abstract


## Summary:

This project explores the Retrieval-Augmented Generation (RAG) framework, a technique that enhances the quality of LLM-generated responses by grounding them in external knowledge sources. By integrating retrieval mechanisms, RAG improves accuracy, reduces hallucinations, and enables models to generate more informed and contextually relevant outputs. The goal of this project is to gain hands-on experience with the key steps involved in implementing and deploying this framework, including data retrieval, embedding generation, vector storage, and response generation. This knowledge will provide a deeper understanding of how RAG can be applied across various domains, improving AI-driven applications in fields such as customer support, research assistance, and knowledge management.


# Introduction

## Background:

With the rapid advancement of Large Language Models (LLMs), their ability to generate human-like text has significantly improved. However, these models often struggle with issues such as hallucination, where they generate inaccurate or misleading information due to their reliance on pre-trained knowledge. To address this limitation, the Retrieval-Augmented Generation (RAG) framework has emerged as a powerful technique that enhances LLM responses by integrating external knowledge retrieval into the generation process.

RAG combines retrieval-based methods with generative AI, allowing models to fetch relevant, real-time information from structured or unstructured data sources before generating responses. This approach not only improves accuracy and factual consistency but also enhances the adaptability of LLMs across various applications, such as customer support, legal research, healthcare, and enterprise knowledge management. By grounding responses in up-to-date, domain-specific data, RAG significantly expands the practical utility of AI-driven applications.

### Project Objectives:

The primary goal of this project is to gain hands-on experience with the end-to-end implementation of a Retrieval-Augmented Generation (RAG) system. This includes:

1. **Understanding the RAG Framework** – Exploring how retrieval and generation mechanisms work together to improve response quality.
2. **Data Processing & Indexing** – Preparing and embedding external knowledge sources into a vector database.
3. **Integrating Retrieval & Generation** – Implementing a pipeline that retrieves relevant information before generating responses.
4. **Model Deployment & Evaluation** – Deploying the system and assessing its effectiveness in improving LLM output.
5. **Exploring Real-World Applications** – Analyzing potential use cases and practical benefits of RAG in different industries.

## Methodology

### Technology Stack:

For this project, Jupyter Notebooks was used as the primary development environment, providing an interactive platform for experimentation and documentation. The project was implemented using Python, which is widely used in the AI and machine learning space for its extensive libraries and frameworks. The main module leveraged in this project was LangChain, which offers a suite of functions essential for data transformation and loading. Key components of LangChain used include Chroma for storing and retrieving vectorized data, TextSplitter for splitting documents into manageable chunks, and both OpenAI Embeddings and HuggingFace Embeddings to explore different embedding techniques for improved model performance.

### Data Collection and Processing:

Exploratory Data Analysis (EDA) was carried out using the Pandas module in Python to uncover valuable insights and gather the necessary information to proceed with the preprocessing and implementation of the framework. The dataset used in the project,

consisting of Puerto Rico News Articles, revealed some key characteristics during the analysis. One of the most notable observations was that several articles had substantial character lengths, with many texts exceeding 5,000 characters. This characteristic posed a challenge for text processing, particularly for the splitting and embedding processes, as it necessitated additional steps to ensure proper handling of lengthy content.

Moreover, through thorough checks, it was confirmed that there were no missing content or gaps in the dataset, ensuring the completeness of the data. Duplicate content was also not present, indicating that the dataset was clean from redundancies. The document lengths were found to vary significantly, with some articles being relatively short, while others were quite long. Descriptive statistics provided further insights, revealing that a large proportion of the content in the dataset easily surpassed the 5,000-character threshold. These findings laid the foundation for the preprocessing decisions that followed, ensuring that the dataset was prepared and ready for the subsequent implementation of the Retrieval Augmentation Generation (RAG) framework.

## Implementation Details:

In our approach, we implemented two different methods for embedding textual data: one using OpenAI's embeddings and another leveraging Hugging Face's sentence transformers. Both methods followed the same pipeline for preprocessing, batching, and storage within a ChromaDB vector database, ensuring a fair comparison of their performance. The implementation was built using the LangChain framework to facilitate seamless integration of embedding models, vector storage, and retrieval functionalities.

For the first approach, we utilized OpenAI's text-embedding-ada-002 model to generate high-quality vector representations of our dataset. These embeddings were then indexed in ChromaDB for efficient similarity searches. To ensure compatibility with OpenAI's API, we structured our data processing pipeline to handle batch requests while adhering to API rate limits. Additionally, we used OpenAI's GPT-4o Mini as the retrieval-augmented generation (RAG) model for answering user queries, retrieving the most relevant document chunks from the vector database before generating a response.

The second approach employed Hugging Face's sentence-transformers/all-MiniLM-L6-v2 model to generate embeddings. This model, which operates locally without API constraints, provides a 384-dimensional embedding space compared to OpenAI's 1536-dimensional vectors. Despite this difference, the same data preprocessing, batching, and storage procedures were applied to maintain consistency between both approaches. Using

LangChain's integration with ChromaDB, we stored and retrieved embeddings in the same manner as with OpenAI's embeddings, ensuring a uniform evaluation process.

Both implementations followed an identical pipeline: first, textual data was chunked and preprocessed, then converted into vector embeddings, and finally stored in ChromaDB. Queries were processed by retrieving the top k most similar embeddings from the database, which were then fed into their respective language models—GPT-4o Mini for OpenAI and a custom response generation approach for Hugging Face. This structured methodology allowed us to fairly compare the effectiveness of each embedding model in a RAG-based retrieval system.

# Results

## Output and Results:

Our evaluation focused on assessing the retrieval and response generation capabilities of both approaches. A key observation was that the OpenAI embedding approach successfully retrieved documents along with their source titles, while the Hugging Face model struggled to consistently associate sources with retrieved documents. This impacted on the interpretability and traceability of the results, making OpenAI's method more reliable for structured information retrieval.

To further analyze performance, we tested the OpenAI-based RAG system against a diverse set of prompts, covering areas such as hallucination detection, response formatting and structure, bias and ethical considerations, edge cases and robustness, citation and source attribution, and basic fact-checking and accuracy. The model handled most of these well, producing relevant and well-structured responses. However, we observed occasional inconsistencies in answering certain questions, particularly when dealing with edge cases or prompts requiring nuanced reasoning. We hypothesized that this could be due to the sources being older and containing information that is not current or up to date.

Overall, OpenAI's approach demonstrated stronger retrieval capabilities and a more structured response generation process, while the Hugging Face method had limitations in maintaining source integrity. These findings highlight the trade-offs between proprietary and open-source embeddings in RAG implementations, where OpenAI's model excels in structured retrieval, whereas Hugging Face offers a more flexible and cost-effective alternative but requires additional fine-tuning to match performance.

# Conclusion

## Summary of achievements:

In summary, for this RAG project we implemented and compared two Retrieval-Augmented Generation (RAG) pipelines using OpenAI and Hugging Face embeddings. Both approaches followed the same batching and vector database storage process with ChromaDB and leveraged LangChain for retrieval and response generation. The OpenAI approach utilized text-embedding-ada-002 for vectorization and GPT-4o Mini for response generation, while the Hugging Face approach used sentence-transformers/all-MiniLM-L6-v2 for embeddings.

Our evaluation revealed notable differences between the two approaches. The OpenAI model consistently retrieved documents along with their source titles, enhancing response transparency. In contrast, the Hugging Face model struggled with consistent source attribution, leading to less reliable references. When tested across various evaluation criteria—including hallucination resistance, formatting and response structure, bias and ethical considerations, edge case handling, citation accuracy, and basic fact-checking, the OpenAI-based system performed fairly well, though it exhibited occasional inconsistencies.

While both approaches demonstrate strengths, future improvements could optimize metadata handling in the Hugging Face pipeline and explore alternative embedding models to balance accuracy, explainability, and efficiency. Testing newer embeddings, such as text-embedding-3-medium or text-embedding-3-small, or fine-tuning sentence-transformers on domain-specific data could further enhance retrieval accuracy and response quality.

In conclusion, this comparison highlights the trade-offs between different embedding models in RAG-based information retrieval and underscores the importance of metadata preservation and source attribution in improving trustworthiness and usability.

## Further improvements:

Future improvements could involve enhancing metadata handling in the Hugging Face pipeline to improve source attribution and retrieval consistency. Additionally, exploring alternative embedding models that offer a balance between accuracy and explainability could lead to better performance. This includes testing other OpenAI embeddings, such as text-embedding-3-medium or text-embedding-3-small, as well as

evaluating more advanced sentence-transformer models to determine their effectiveness in document retrieval and response generation. Further refinements could also involve fine-tuning embeddings of domain-specific data to improve contextual relevance and reduce hallucinations, as well as the inclusion of more up-to-date sources.