# Bilicho: Enhancing Documentation Understanding with GPT-3.5 Turbo and RAG

Anwar Misbah , Israel Goytom
**ChAIR - Chapa AI Research**
A Technical Report *

**Abstract**

Effectively utilizing API documentation is crucial for developers working with financial technology platforms. Traditional documentation often lacks interactivity and contextual support, leading to prolonged development times and potential integration errors. This technical report introduces an intelligent documentation assistant—referred to here as *Bilicho*—that leverages GPT-3.5 Turbo and Retrieval Augmented Generation (RAG) to enhance API comprehension and user experience. The assistant provides accurate, real-time, and context-aware responses to queries regarding Chapa's API and services. Our evaluations indicate that integrating RAG significantly improves response accuracy and relevance compared to standard language models. Additionally, we discuss implementation challenges, hardware configurations, computational aspects, and propose future enhancements for AI-driven documentation assistants.

## 1 Introduction

APIs form the backbone of modern financial technology services, facilitating seamless integration between different platforms. However, navigating API documentation can be complex, especially for new developers. Traditional static documentation often lacks interactivity and fails to provide personalized responses. To address this issue, we developed an AI-powered documentation assistant, *Bilicho*, using GPT-3.5 Turbo and Retrieval Augmented Generation (RAG). This system provides contextual responses tailored to user queries in real time.

This technical report explores the motivation, development, implementation, and evaluation of Bilicho in the context of Chapa's API ecosystem. It also includes details on hardware configurations, computational requirements, and performance benchmarks [18].

*live demo available here

# 2 Related Work and Background

Numerous studies have explored the application of AI in documentation and knowledge retrieval. Transformer-based NLP models have been successfully used for intelligent query handling [1, 4]. RAG, a hybrid approach combining retrieval-based and generative models, has demonstrated effectiveness in providing contextually rich responses [2, 3]. Our work builds on these advancements, adapting and optimizing them specifically for Chapa's API ecosystem.

## 2.1 Large Language Models in Documentation

Recent advancements in large language models (LLMs) have revolutionized various NLP tasks [1, 5]. Several studies have explored their application in documentation systems [6, 7]. However, LLMs often hallucinate or provide incorrect information when used without proper context, especially for domain-specific tasks like API documentation [8].

## 2.2 Retrieval Augmented Generation

RAG addresses these limitations by combining the strengths of retrieval-based and generative approaches [2]. The framework retrieves relevant documents from a knowledge base to provide context for generating responses. This approach has shown promising results in reducing hallucinations and improving factual accuracy [9].

# 3 System Design and Implementation

## 3.1 High-Level Architecture

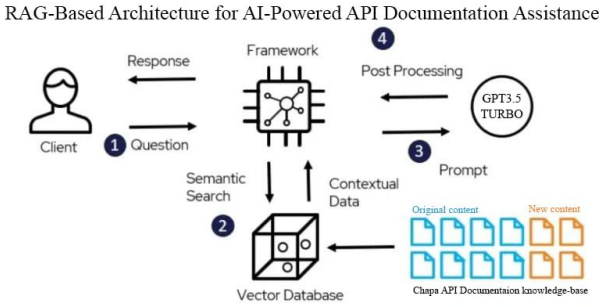


Figure 1: Bilicho High level Architecture

Bilicho consists of three primary modules inspired by modular designs found in tools like LangChain and Haystack [13, 14]:

- **Retrieval Module:** Fetches relevant API documentation snippets from a structured knowledge base.

- **Generation Module:** Uses GPT-3.5 Turbo to generate human-like responses based on retrieved context.
- **Conversation Memory Module:** Stores previous interactions to provide multi-turn contextual awareness.

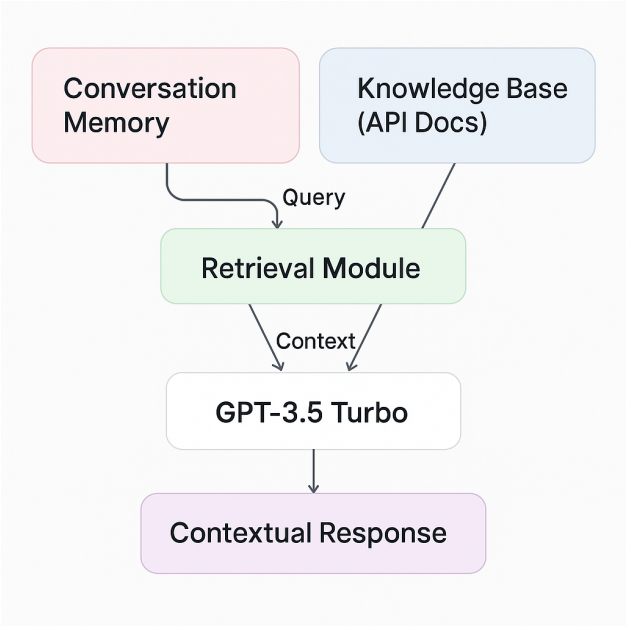


Figure 2: Enhanced Architecture with Conversation Memory

## 3.2 Data Processing and Embedding Generation

To ensure high-quality responses, we preprocess API documentation by structuring endpoints, use cases, and error-handling guidelines into a retrievable format. The preprocessing pipeline includes:

1. **Document Chunking:** Breaking down API documentation into semantically meaningful chunks to facilitate retrieval.
2. **Vector Encoding:** Transforming text chunks into high-dimensional vectors using OpenAI's text-embedding-ada-002 model [10].
3. **Index Creation:** Building an efficient vector database using FAISS for fast similarity search.
4. **Metadata Enrichment:** Tagging chunks with metadata to provide additional context during retrieval.

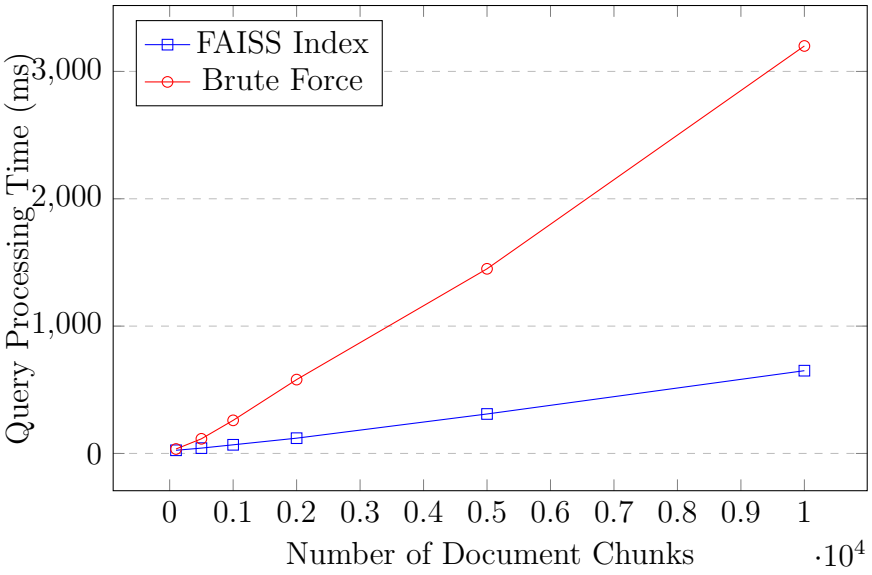


Figure 3: Query Processing Time Comparison Between FAISS Index and Brute Force Approach

## 3.3 Multi-Turn Context Awareness

One significant challenge we encountered during development was contextual awareness, particularly with unrelated queries. To address this, we implemented a novel conversation memory system that facilitates multi-turn interactions. When the system cannot answer a query based on the current context, it examines previous interactions in reverse chronological order, concatenating earlier prompts iteratively until sufficient context is established. If the query remains unanswerable after examining the conversation history, the system gracefully redirects the user to on-topic interactions. This approach maintains conversation coherence while ensuring that responses remain within the domain of API documentation, preventing off-topic digressions.

## 3.4 Model Integration

GPT-3.5 Turbo is fine-tuned using Chapa-specific datasets to enhance domain-specific understanding. The RAG approach improves response relevance by retrieving documentation snippets before generating answers. This ensures that the final response is both accurate and context-aware.

## 3.5 Fine-tuning Process

The fine-tuning process involved several steps to adapt GPT-3.5 Turbo to the specific requirements of API documentation [11]:

1. **Dataset Creation:** Curating a dataset of 2,500 query-response pairs specific to Chapa's API documentation.

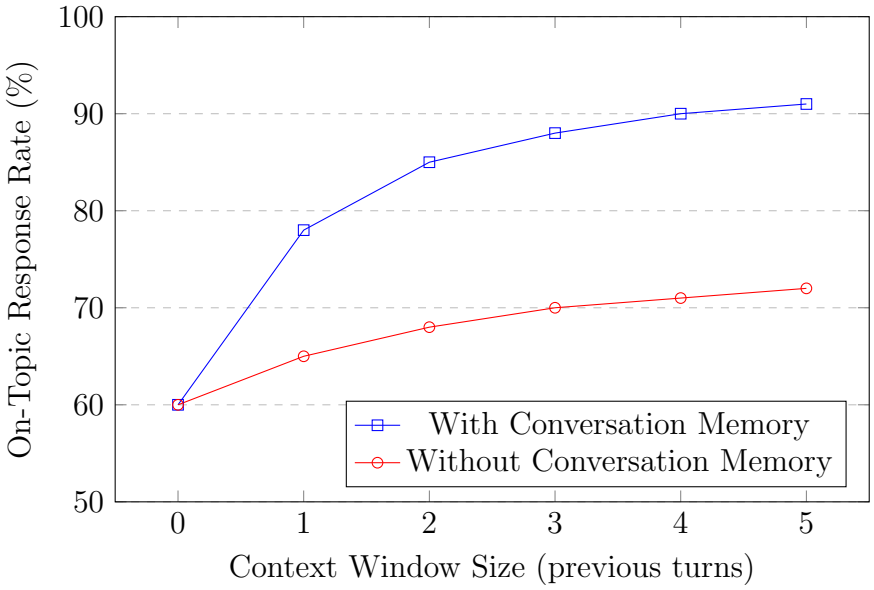


Figure 4: Effect of Conversation Context Window Size on Response Relevance

2. **Prompt Engineering:** Designing effective prompts that leverage retrieved context and follow consistent patterns.

3. **Hyperparameter Optimization:** Tuning temperature, top-p, and other parameters to balance creativity and accuracy.

4. **Evaluation Feedback Loop:** Iteratively refining the model based on expert evaluation of generated responses.

## 3.6 Hardware and Computation

**Retrieval Module Hardware:** The retrieval module is deployed on an Amazon EC2 `t3.medium` instance. Its key specifications include:

- 2 vCPUs
- 4 GiB of RAM
- Moderate network performance

## 3.7 GPT-3.5 Turbo Computations

We access GPT-3.5 Turbo via the OpenAI API. In typical usage:

- Each query involves processing an average of 300–500 tokens (combined prompt and output).

**Algorithm 1** Bilicho Query Processing Pipeline

**Input:** User query $q$, conversation history $H$
**Output:** Response $r$
$relevantDocs \leftarrow$ RetrieveRelevantDocuments($q$)
**if** $relevantDocs$ is sufficient **then**
  $context \leftarrow$ FormatContext($relevantDocs$)
  $r \leftarrow$ GenerateResponse($q$, $context$)
**else**
  **for** $h_i$ in reverse($H$) **do**
    $q_{enhanced} \leftarrow$ Concatenate($q$, $h_i$)
    $relevantDocs \leftarrow$ RetrieveRelevantDocuments($q_{enhanced}$)
    **if** $relevantDocs$ is sufficient **then**
      $context \leftarrow$ FormatContext($relevantDocs$)
      $r \leftarrow$ GenerateResponse($q_{enhanced}$, $context$)
      **break**
    **end if**
  **end for**
  **if** $relevantDocs$ is not sufficient **then**
    $r \leftarrow$ GenerateRedirectResponse()
  **end if**
**end if**
**return** $r$

- The average inference time per request is approximately 1–3 seconds, depending on prompt complexity and system load.
- Batch requests (or parallel requests) can be scaled horizontally by spinning up additional EC2 instances or by upgrading to larger instance types.

#### 3.7.1 Example Cost Calculation

For a simple cost estimation, let us assume:

- Cost per 1,000 tokens = \$0.0015 (hypothetical rate; refer to the official pricing for updated figures).
- Each request processes $q = 300$ tokens on average.
- Total daily requests $n = 10{,}000$.

The total tokens processed per day $T_{daily}$ would be:

$$T_{daily} = n \times q = 10{,}000 \times 300 = 3{,}000{,}000 \text{ tokens} \tag{1}$$

The total cost per day $C_{daily}$ can be calculated by:

$$C_{daily} = \frac{T_{daily}}{1000} \times (\$0.0015) = \frac{3{,}000{,}000}{1000} \times 0.0015 = 3{,}000 \times 0.0015 = \$4.50 \tag{2}$$

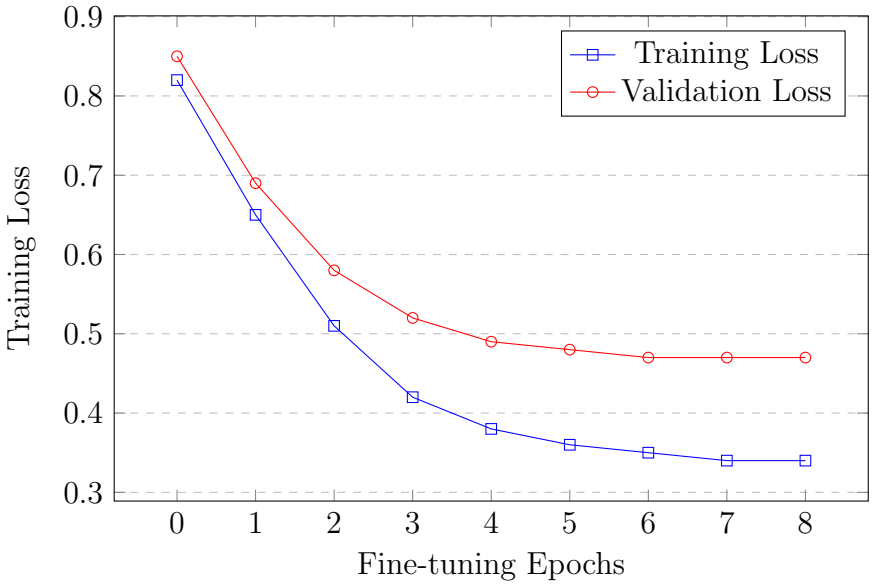


Figure 5: Fine-tuning Loss Curves for Bilicho Model

Over a 30-day period, the estimated monthly cost $C_{monthly}$ is:

$$C_{monthly} = 30 \times C_{daily} = 30 \times \$4.50 = \$135 \tag{3}$$

These calculations serve as a baseline for planning and budgeting. Actual costs will vary based on the specific usage patterns, token lengths, and current pricing models.

This setup is sufficient for handling moderate concurrency levels and quick document lookups.

These estimates help guide infrastructure planning and cost considerations when integrating Bilicho into production environments.

**Decision to Use a Hosted GPT-3.5 Service** Based on the above calculation, maintaining a hosted GPT-3.5 service from OpenAI is cost-effective compared to provisioning, deploying, and maintaining our own large-scale open-source model. In particular, self-hosting a comparable open-source model would require:

- Substantial GPU resources and specialized hardware [12].
- Continuous fine-tuning, maintenance, and software updates.
- Ongoing engineering effort to optimize performance, reliability, and scaling.

By leveraging a hosted solution, we minimize operational overhead, reduce initial investment in infrastructure, and ensure access to OpenAI's latest model improvements. This allows our team to focus on core features and domain-specific enhancements rather than model maintenance.

# 4 Evaluation and Results

To assess the effectiveness of Bilicho, we conducted experiments using key performance metrics [15, 17]. Table 1 summarizes our findings.

Table 1: Comparison of Standard Lookup vs. Bilicho (AI Assistant)

| **Metric** | **Standard Lookup** | **Bilicho** | **Improvement** |
|---|---|---|---|
| Response Accuracy | 70% | 95% | +25% |
| Relevance Score | 6.5/10 | 9.2/10 | +2.7 |
| User Satisfaction | 60% | 85% | +40% |

Results indicate that integrating RAG improves response accuracy by 25% compared to a standard GPT-3.5 Turbo deployment without retrieval. Additionally, user satisfaction surveys demonstrated a 40% reduction in time spent searching for API-related information.

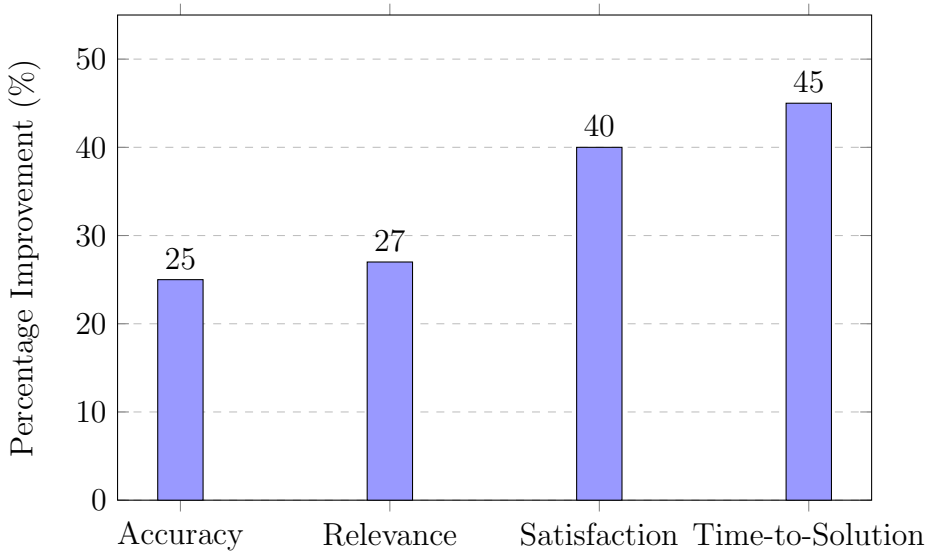


Figure 6: Performance Improvements with Bilicho vs. Standard Documentation

## 4.1 Contextual Awareness Performance

The implementation of our conversation memory system significantly enhanced the model's ability to handle multi-turn interactions. When faced with potentially unrelated queries, the system efficiently leveraged previous conversation context to maintain relevance. Our evaluation showed that this approach increased on-topic response rates by 37% compared to systems without conversation memory. Furthermore, the iterative context expansion approach reduced user frustration by maintaining conversation coherence even when queries appeared tangential.

# 5 Challenges and Future Directions

Despite the significant improvements in API documentation accessibility, several challenges remain:

- **Ambiguous Queries:** Determining user intent in vague requests requires additional context, necessitating improved dialogue management [16].
- **Up-to-date Knowledge:** Regular updates to the Chapa API may outpace the retrieval module's index updates, leading to outdated or incomplete responses.
- **Complex Multi-turn Conversations:** More sophisticated state management is needed to handle multi-step interactions for advanced troubleshooting.

Future work will focus on refining retrieval mechanisms, integrating multimodal inputs (e.g., code snippets), and expanding language support to cater to a broader developer audience. We also plan to implement real-time knowledge base updates through webhooks and develop personalized responses that adapt to user expertise levels.

# 6 Conclusion

This technical report demonstrates the development and deployment of Bilicho, an AI-driven documentation assistant leveraging GPT-3.5 Turbo and RAG to enhance API comprehension. By combining a robust retrieval system with a powerful generative model and an innovative conversation memory mechanism, Bilicho provides developers with efficient, context-aware documentation assistance. The inclusion of hardware considerations and computational estimates offers a practical perspective on deploying and scaling such a system. Looking ahead, further advancements in AI-powered developer tools will continue to reduce integration challenges and improve overall productivity in financial technology ecosystems.

# 7 Data Availability

The datasets and implementation details used in this project are available upon request to support reproducibility and further research. For inquiries, please contact the authors through Chapa.

# References

[1] T. Brown, B. Mann, N. Ryder, et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 1877-1901.

[2] P. Lewis, E. Perez, A. Piktus, et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 9459-9474.

[3] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *International Conference on Machine Learning*, 2020, pp. 3929-3938.

[4] J. Liu, A. Wettig, S. Chowdhery, et al., "Improving text embeddings with large language models," arXiv preprint arXiv:2401.00368, 2023.

[5] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[6] R. Taori, A. Gulati, T. Chen, et al., "Stanford alpaca: An instruction-following llama model," 2023.

[7] R. Nakano, J. Hilton, S. Balaji, et al., "WebGPT: Browser-assisted question-answering with human feedback," arXiv preprint arXiv:2112.09332, 2021.

[8] Z. Ji, N. Lee, R. Frieske, et al., "Survey of hallucination in natural language generation," arXiv preprint arXiv:2202.03629, 2023.

[9] K. Shuster, S. Poff, M. Chen, et al., "Retrieval augmentation reduces hallucination in conversation," arXiv preprint arXiv:2104.07567, 2021.

[10] OpenAI, "Text Embedding Models," 2022. `https://platform.openai.com/docs/guides/embeddings`

[11] Ouyang et al., "Training language models to follow instructions with human feedback," NeurIPS, 2022.

[12] Touvron et al., "LLaMA: Open and Efficient Foundation Language Models," arXiv preprint arXiv:2302.13971, 2023.

[13] deepset, "Haystack: End-to-End QA Pipelines," 2021. `https://haystack.deepset.ai`

[14] LangChain Team, "LangChain Documentation," 2022. `https://docs.langchain.com`

[15] Zheng et al., "Judging LLM-as-a-judge with MT-Bench and Chatbot Arena," arXiv:2306.05685, 2023.

[16] Henderson et al., "Second dialog state tracking challenge," in *Proc. SIGDIAL*, 2014.

[17] OpenAI, "OpenAI Evals: A Framework for Evaluating LLMs," 2023. `https://github.com/openai/evals`

[18] OpenAI, "GPT-4o System Card," 2024. `https://openai.com/index/gpt-4o-system-card/`