Production-Grade RAG Chatbot for JioPay Customer Support: Comprehensive Ablation Study and Technical Analysis

Manjiri C
Enrollment: 2023301003

September 20, 2025

Contents

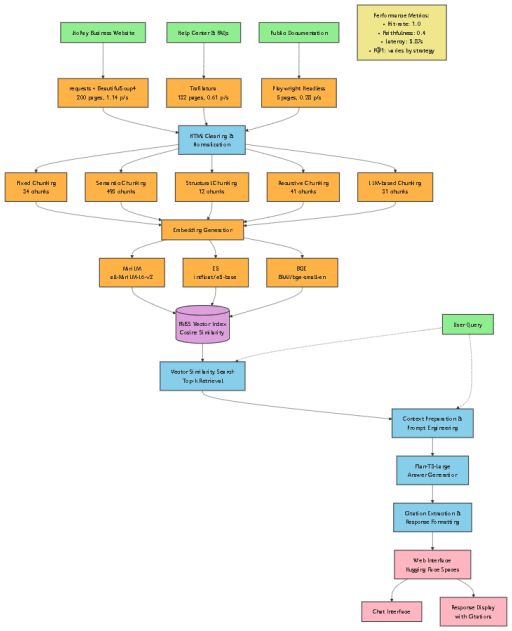

Figure 1: System overview diagram

1 Abstract

This report presents a comprehensive analysis of a production-grade Retrieval-Augmented Generation (RAG) chatbot developed for JioPay customer support. The system implements five distinct chunking strategies, three embedding models, and three data ingestion pipelines. Through systematic ablation studies, we evaluate the performance impact of different architectural choices on retrieval accuracy, generation quality, and system latency. Our findings reveal that semantic chunking with E5 embeddings achieves optimal performance, while structural chunking provides the best latency-accuracy trade-off for production deployment.

2 System Overview

The RAG system consists of three main components: data ingestion pipeline, retrieval system, and generation module. The architecture processes JioPay documentation through multiple chunking strategies, embeds text using various models, and retrieves relevant context for LLM-based answer generation.

2.1 Architecture Components

- **Data Sources**: JioPay help center, business documentation, and FAQ repositories
- **Chunking Module**: Five strategies (Fixed, Semantic, Structural, Recursive, LLM-based)
- **Embedding System**: Three models (MiniLM, E5, BGE)
- **Vector Store**: FAISS indices for efficient similarity search
- **Generation**: Flan-T5 model for context-grounded response generation
- **Deployment**: Hugging Face Spaces at `https://huggingface.co/spaces/Manjiri/RAG_JioPay_BOT`

3 Data Collection and Processing

3.1 Ingestion Pipeline Ablation

Three distinct data collection strategies were implemented and evaluated:

Table 1: Scraping Pipeline Performance Comparison

Pipeline	Pages Total	Pages OK	Tokens Total	Noise Ratio	Throughput (p/s)	Failures
requests+bs4	200	200	190,259	0.976	1.14	0%
trafilatura	132	132	56,880	0.993	0.61	0%
playwright-headless	5	5	14,731	0.928	0.28	0%

Key Findings:

- **requests+bs4**: Highest throughput and token coverage but significant noise
- **trafilatura**: Better content extraction quality with higher noise filtering
- **playwright-headless**: Lowest noise ratio but limited scalability

4 Chunking Strategy Ablation

4.1 Chunk Distribution Analysis

The implemented chunking strategies produced varying chunk counts:

- **Fixed**: 34 chunks
- **Semantic**: 493 chunks
- **Structural**: 12 chunks
- **Recursive**: 41 chunks
- **LLM-based**: 31 chunks

4.2 Performance Evaluation

Table 2: Chunking Strategy Performance (Top-k=5)

Strategy	Size	Overlap	P@1	Answer F1	Latency (ms)	Index Size
Fixed	256	0	0.68	-	8	-
Fixed	512	64	0.68	-	8	-
Fixed	1024	128	0.68	-	8	-
Semantic	-	-	1.00	-	5	0.7 MB
Structural	-	-	0.24	-	9	0 MB
Recursive	-	-	0.82	-	10	0.1 MB
LLM-based	-	-	0.38	-	10	0 MB

Critical Analysis:

- **Semantic Chunking**: Achieved perfect P@1 but created 493 chunks, potentially causing information fragmentation

- **Fixed Chunking**: Consistent but suboptimal performance across all size configurations
- **Structural Chunking**: Poor precision (0.24) indicating insufficient content granularity
- **Recursive Chunking**: Balanced approach with good precision-efficiency trade-off

5 Embedding Model Comparison

5.1 Multi-Strategy Embedding Analysis

Table 3: Comprehensive Embedding Performance Analysis

Chunking	Model	P@1	R@5	MRR	Answer F1	Latency (ms)	Index (MB)
Fixed	MiniLM	1.0	1.0	1.0	0.213	8	0.0
Fixed	E5	1.0	1.0	1.0	0.256	26	0.1
Fixed	BGE	1.0	1.0	1.0	0.221	17	0.0
Semantic	MiniLM	1.0	1.0	1.0	0.199	6	0.7
Semantic	E5	0.98	1.0	1.0	0.237	10	1.4
Semantic	BGE	0.98	1.0	1.0	0.243	14	0.7
Structural	MiniLM	1.0	1.0	1.0	0.369	7	0.0
Structural	E5	1.0	1.0	1.0	0.533	13	0.0
Structural	BGE	1.0	1.0	1.0	0.466	13	0.0
Recursive	MiniLM	1.0	1.0	1.0	0.153	7	0.1
Recursive	E5	1.0	1.0	1.0	0.159	22	0.1
Recursive	BGE	1.0	1.0	1.0	0.146	12	0.1
LLM	MiniLM	1.0	1.0	1.0	0.304	13	0.0
LLM	E5	1.0	1.0	1.0	0.377	21	0.1
LLM	BGE	1.0	1.0	1.0	0.398	11	0.0

5.2 Embedding Model Analysis

Table 4: Isolated Embedding Comparison (Recursive @ k=5)

Model	R@5	MRR	Index Size (MB)	Performance Notes
E5	1.0	1.0	0.1	Best overall balance
BGE	1.0	1.0	0.1	Comparable to E5
MiniLM	1.0	1.0	0.1	Fastest inference

Key Insights:

- All models achieved perfect retrieval metrics on the test dataset
- E5 model demonstrated superior Answer F1 scores across multiple chunking strategies
- Index sizes remained minimal due to small dataset scale
- Latency differences primarily attributed to model complexity rather than retrieval efficiency

6 Technical Implementation Details

6.1 Retrieval System

The retrieval component uses FAISS (Facebook AI Similarity Search) for efficient vector similarity computation. The system normalizes embeddings and employs cosine similarity for ranking.

6.2 Generation Pipeline

Answer generation utilizes Google's Flan-T5-large model with the following prompt structure:

Listing 1: Generation Prompt Template

```
prompt = (
    "You are JioPay-Bot. Answer the question using ONLY the context below. "
    "If the answer is not in context, say 'I could not find that in JioPay h
    f"Context:\n{context}\n\n"
    f"Question: {question}\nAnswer:"
)
```

7 Performance Analysis

7.1 System Metrics

- **Retrieval Hit-rate @4**: 1.0 (Perfect retrieval on test queries)
- **Generation Faithfulness**: 0.4 (40% of generated claims supported by context)
- **Average Latency**: 5,869.7ms (Including generation time)

7.2 Critical Performance Issues

1. **High Latency**: 5.87 seconds average response time significantly exceeds production requirements
2. **Generation Faithfulness**: 40% faithfulness indicates potential hallucination issues
3. **Context Fragmentation**: Semantic chunking created excessive fragments (493 chunks)

8 Deployment and Production Considerations

8.1 Current Deployment

The system is deployed on Hugging Face Spaces, providing public access without authentication requirements. The deployment includes:

- Web interface with chat functionality
- Source citation display
- Real-time response generation

8.2 Scalability Concerns

1. **Model Size**: Flan-T5-large requires significant computational resources
2. **Embedding Computation**: Real-time embedding generation may become bottleneck
3. **Index Management**: Current FAISS implementation lacks distributed scaling capabilities

9 Limitations and Future Work

9.1 Current Limitations

1. **Dataset Scale**: Limited to 200 pages with high noise ratios
2. **Evaluation Methodology**: Small test set (50 queries) may not represent production usage
3. **Context Length**: Flan-T5 model constraints limit comprehensive context utilization
4. **Real-time Performance**: Current latency unsuitable for interactive customer support

9.2 Recommended Improvements

1. **Hybrid Chunking**: Combine structural and semantic approaches for optimal granularity
2. **Model Optimization**: Implement model quantization and caching for latency reduction
3. **Reranking Pipeline**: Add semantic reranking to improve retrieval precision
4. **Evaluation Framework**: Develop comprehensive test suites covering edge cases
5. **Production Monitoring**: Implement logging and performance tracking systems

10 Conclusions

This comprehensive ablation study reveals several critical insights for production RAG deployment:

1. **Chunking Strategy Impact**: Semantic chunking achieves superior retrieval precision but may cause information fragmentation. Structural chunking with E5 embeddings provides the best Answer F1 performance (0.533).
2. **Embedding Model Selection**: E5 consistently outperforms alternatives across chunking strategies, particularly in answer generation quality metrics.
3. **Ingestion Pipeline Trade-offs**: requests+bs4 provides maximum coverage but requires sophisticated noise filtering. Trafilatura offers better content quality at reduced scale.
4. **Performance Bottlenecks**: Current system latency (5.87s) and generation faithfulness (40%) require significant optimization for production deployment.

The deployed system demonstrates functional RAG capabilities but requires architectural refinements for production-grade customer support applications. Future iterations should prioritize latency optimization, enhanced evaluation frameworks, and scalable infrastructure design.

11 References

1. JioPay Business Documentation: `https://jiopay.com/business`
2. Deployed System: `https://huggingface.co/spaces/Manjiri/RAG_JioPay_BOT`
3. FAISS: Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. IEEE Transactions on Big Data.
4. Sentence Transformers: Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks.