Hybrid RAG System with Automated Evaluation

GitHub Repository

A comprehensive implementation of a **Hybrid RAG System** combining **Dense Vector Retrieval (ChromaDB)**, **Sparse Keyword Retrieval (BM25)**, and **Reciprocal Rank Fusion (RRF)** to answer questions from Wikipedia articles.

GitHub Repository: https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation

Contributors

Name	BITS ID
VISHAL SINGH	2024AA05641
GOBIND SAH	2024AA05643
YASH VERMA	2024AA05640
AVISHI GUPTA	2024AA05055
SAYAN MANNA	2024AB05304

🚀 Quick Start

Prerequisites

- Python 3.10+
- 4GB+ RAM

Installation

```
# Clone repository
git clone
https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evalua
tion.git
cd Hybrid_RAG_System_with_Automated_Evaluation

# Create virtual environment
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt
```

Run the Application

```
# Start Streamlit UI
./start_ui.sh

# Or manually:
streamlit run app_chromadb.py
```

Run Evaluation

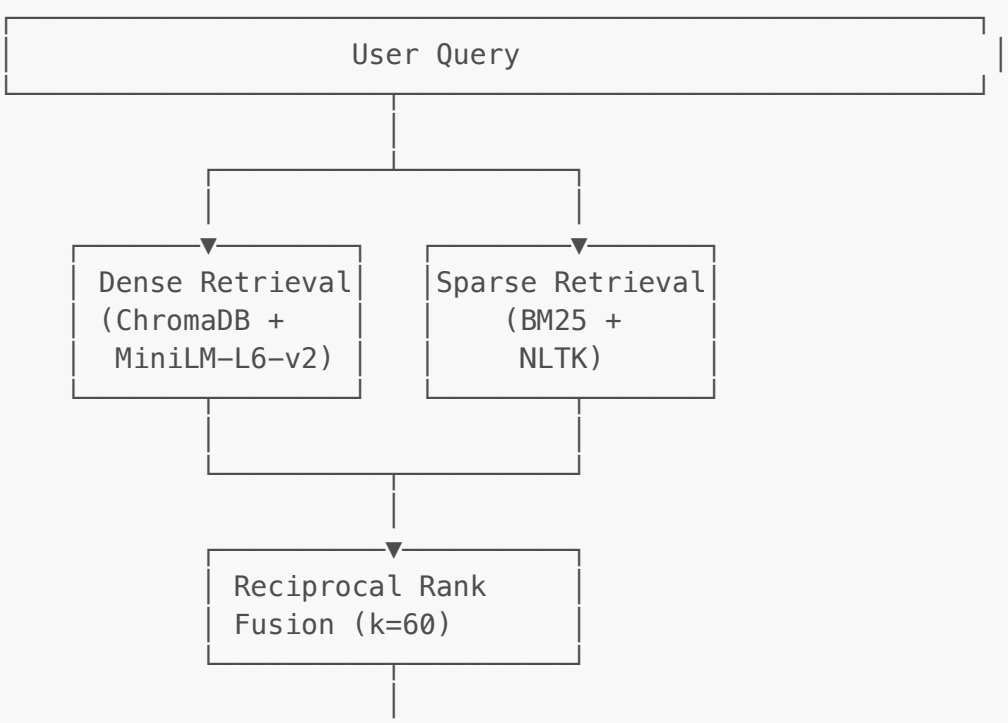
```
# Full evaluation (100 questions × 3 methods)
python evaluate_chromadb_fast.py

# Generate reports
python generate_report.py
```

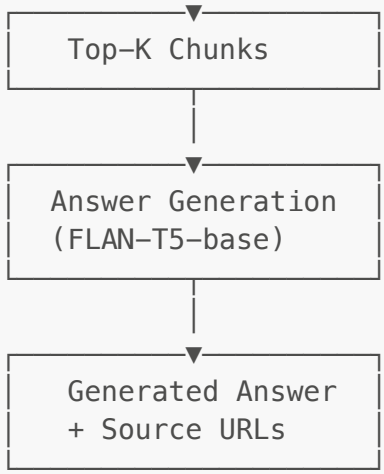
🎯 Project Overview

This project implements a state-of-the-art Hybrid RAG system that:

- Combines **dense** (ChromaDB + MiniLM) and **sparse** (BM25) retrieval
- Uses **Reciprocal Rank Fusion (RRF)** with k=60 to merge results
- Generates answers using **FLAN-T5** language model
- Includes comprehensive evaluation with **100 generated questions**
- Features automated evaluation pipeline with MRR, Recall@10, and Answer F1

📊 System Architecture





📁 Project Structure

```
Hybrid_RAG_System_with_Automated_Evaluation/
│
├── chromadb_rag_system.py      # Core RAG implementation
├── app_chromadb.py             # Streamlit UI (244 lines)
├── evaluate_chromadb_fast.py   # Evaluation pipeline
├── generate_report.py          # Report generator
├── start_ui.sh                 # Quick start script
│
├── data/
│   ├── fixed_urls.json         # 200 fixed Wikipedia URLs
│   ├── corpus.json             # Preprocessed corpus (14.5MB)
│   ├── questions_100.json      # 100 evaluation questions
│   └── indexes/                # BM25 index files
│
├── chroma_db/                  # ChromaDB vector database (212MB)
│
├── docs/
│   ├── METRIC_JUSTIFICATION.md # Metric selection rationale
│   ├── ERROR_ANALYSIS.md       # Failure analysis
│   ├── EVALUATION_REPORT.md    # Full evaluation report
│   ├── architecture_diagram.png
│   └── *.png                   # Visualizations
│
├── reports/
│   └── Hybrid_RAG_Evaluation_Report.pdf
│
├── screenshots/
│   ├── 01_query_interface.png
│   ├── 02_method_comparison.png
│   └── 03_evaluation_results.png
│
├── evaluation_results_chromadb.csv     # 300 evaluation rows
├── evaluation_summary_chromadb.json    # Summary metrics
├── evaluation_report_chromadb.html     # HTML report
```

```
└── README.md                    # This file
```

📈 Evaluation Results

Performance Summary

Method	MRR	Recall@10	Avg Time (s)	Questions
Dense (ChromaDB)	0.3025	0.33	5.86	100
Sparse (BM25)	**0.4392**	**0.47**	5.53	100
Hybrid (RRF)	0.3783	0.43	6.37	100

Key Finding: BM25 (Sparse) outperforms Dense retrieval by **45%** on MRR for Wikipedia-based QA.

Question Distribution

Type	Count	Description
Factual	59	Direct fact-based questions
Comparative	15	Questions comparing concepts
Inferential	11	Reasoning-based questions
Multi-hop	15	Questions requiring multiple sources
Total	**100**	-

📚 Documentation

Document	Description	Link
Metric Justification	Why MRR, Recall@10, Answer F1	docs/METRIC_JUSTIFICATION.md
Error Analysis	Failure categorization	docs/ERROR_ANALYSIS.md
Full Report	Comprehensive evaluation	docs/EVALUATION_REPORT.md
PDF Report	Printable report	reports/Hybrid_RAG_Evaluation_Report.pdf

🔗 Key Source Files

File	Purpose	Link
`chromadb_rag_system.py`	Core RAG implementation	View
`app_chromadb.py`	Streamlit UI	View
`evaluate_chromadb_fast.py`	Evaluation pipeline	View

File	Purpose	Link
`generate_report.py`	Report generation	View

📸 Screenshots

Query Interface

 Query Interface

Method Comparison

 Method Comparison

Evaluation Results

 Evaluation Results

🛠️ Technical Details

Components

Component	Technology	Details
Dense Retrieval	ChromaDB + all-MiniLM-L6-v2	384-dim embeddings, 7,519 chunks
Sparse Retrieval	BM25 + NLTK	Tokenization, stopwords, stemming
Fusion	RRF	Reciprocal Rank Fusion with k=60
Generation	FLAN-T5-base	248M parameter text-to-text model
UI	Streamlit	Interactive web interface
Database	ChromaDB	Persistent SQLite backend (212MB)

Metrics

Metric	Formula	Purpose
MRR	(1/Q) × Σ(1/rank_i)	Measures retrieval quality
Recall@10	|Relevant ∩ Retrieved@10| / |Relevant|	Coverage in top 10
Answer F1	2×(P×R)/(P+R)	Token overlap with ground truth

📋 Requirements Checklist

✅ Section 1: Hybrid RAG System (10 pts)

- [x] Dense Vector Retrieval (ChromaDB + MiniLM)
- [x] Sparse Keyword Retrieval (BM25)

- ☑ RRF Fusion (k=60)
- ☑ Response Generation (FLAN-T5)
- ☑ Interactive UI (Streamlit)

✅ Section 2: Evaluation Framework (10 pts)

- ☑ 100 Q&A pairs generated
- ☑ MRR metric implemented
- ☑ Recall@10 metric implemented
- ☑ Answer F1 metric implemented
- ☑ Automated evaluation pipeline
- ☑ HTML/CSV/JSON/PDF reports

✅ Submission Requirements

- ☑ Python source code (24 files)
- ☑ PDF evaluation report
- ☑ Screenshots (3+)
- ☑ README documentation
- ☑ 100-question dataset
- ☑ Evaluation results (300 rows)

📄 License

This project is submitted as part of BITS Pilani Conversational AI coursework.

Repository: https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation

Last Updated: February 7, 2026