SUBMISSION REFERENCE GUIDE

Hybrid RAG System with Automated Evaluation

Field	Value
Project Name	Hybrid RAG System with Automated Evaluation
GitHub Repository	https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation
Submission Date	February 8, 2026
Final Status	Complete - 100%

Contributors

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

TABLE OF CONTENTS

1. PROJECT OVERVIEW

System Architecture

A comprehensive Hybrid RAG system combining:

Component	Technology

Component	Technology
Dense Retrieval	ChromaDB + all-MiniLM-L6-v2 embeddings
Sparse Retrieval	BM25 + NLTK tokenization
Fusion	Reciprocal Rank Fusion (RRF) with k=60
Generation	FLAN-T5-base with confidence calibration

Key Statistics

Metric	Value
Total URLs	500 (200 fixed + 300 random)
Total Chunks	7,519 segments
Chunk Size	200-400 tokens with 50-token overlap
Questions	100 main + 30 adversarial = 130 total
Evaluation Metrics	6 comprehensive metrics
Innovation Techniques	7 advanced techniques

Interactive Dashboard Features (NEW)

The Streamlit UI now includes:

Feature	Description
Chunk Score Visualization	Interactive bar chart showing Dense, Sparse, and RRF scores
Dense vs Sparse vs Hybrid Comparison	Side-by-side tabs showing top 5 chunks from each method
Real-time Metrics	Live MRR, Recall@10, Response Time updates
Per-Question Breakdown	Last 5 queries with complete metrics

2. REPOSITORY STRUCTURE

Path	Description	GitHub Link
README.md	Main project documentation	View
config.yaml	System configuration	View
requirements.txt	Python dependencies	View
chromadb_rag_system.py	Main RAG system	View
app_chromadb.py	Streamlit UI	View
build_chromadb_system.py	ChromaDB index builder	View

Data Files

File	Description	GitHub Link
data/fixed_urls.json	200 fixed Wikipedia URLs	View
data/corpus.json	Processed corpus (7,519 chunks)	View
data/questions_100.json	100 Q&A pairs	View
data/adversarial_questions.json	30 adversarial questions	View

Source Code Modules

File	Description	GitHub Link
src/data_collection.py	Wikipedia data collector	View
src/semantic_chunker.py	Semantic chunking	View
src/rrf_fusion.py	Reciprocal Rank Fusion	View

Evaluation Framework

File	Description	GitHub Link
evaluation/metrics.py	Core metrics (MRR, BERTScore)	View
evaluation/novel_metrics.py	4 novel metrics	View
evaluation/innovative_eval.py	Innovative techniques	View
evaluation/run_evaluation.py	Main evaluation pipeline	View
evaluation/create_dataset.py	Question generation	View

Submission Folder

Folder	Description	GitHub Link
submission/01_source_code/	All source code	View
submission/02_data/	All data files	View
submission/03_vector_database/	ChromaDB database	View
submission/04_evaluation_results/	Evaluation outputs	View
submission/05_reports/	Reports (PDF, MD)	View
submission/06_documentation/	Documentation files	View
submission/07_visualizations/	Charts and diagrams	View
submission/08_screenshots/	System screenshots	View

3. DATASET REQUIREMENTS

3.1 Fixed URLs (200 URLs)

Item	Details
File	data/fixed_urls.json
Count	200 unique Wikipedia URLs
GitHub	View File

Topics Covered: Science, Technology, History, Geography, Arts, Sports, Philosophy, Literature, Mathematics, Medicine

3.2 Random URLs (300 URLs per run)

Item	Details
Implementation	src/data_collection.py
Count	300 random Wikipedia URLs
GitHub	View File

3.3 Chunking Strategy

Parameter	Value
Min Tokens	200
Max Tokens	400
Overlap	50 tokens
Tokenizer	tiktoken (cl100k_base)

3.4 Corpus Storage

Item	Details
File	data/corpus.json
Total Chunks	7,519 segments
File Size	14 MB
GitHub	View File

---

4. PART 1: HYBRID RAG SYSTEM

4.1 Dense Vector Retrieval

Component	Details

Component	Details
Embedding Model	sentence-transformers/all-MiniLM-L6-v2
Vector Database	ChromaDB with persistent storage
Similarity Metric	Cosine similarity
Implementation	chromadb_rag_system.py

4.2 Sparse Keyword Retrieval

Component	Details
Algorithm	BM25Okapi
Tokenizer	NLTK word_tokenize
Parameters	k1=1.5, b=0.75
Implementation	chromadb_rag_system.py

4.3 Reciprocal Rank Fusion (RRF)

Component	Details
Formula	RRF_score(d) = sum(1/(k + rank_i(d)))
K Value	60
Implementation	src/rrf_fusion.py

4.4 Response Generation

Component	Details
Model	google/flan-t5-base (248M parameters)
Max Length	512 tokens
Temperature	0.7
Implementation	chromadb_rag_system.py

4.5 User Interface (Enhanced)

Feature	Description
Query Input	Text box with example queries
Answer Display	Generated answer with confidence score
Chunk Score Visualization	Interactive Plotly bar chart (Dense/Sparse/RRF)
Dense vs Sparse vs Hybrid Tabs	Side-by-side comparison of top 5 chunks

Feature	Description
Real-time Metrics	MRR, Recall@10, Response Time
Per-Question Breakdown	Last 5 queries with metrics

UI File: app_chromadb.py

Launch Command:

```
streamlit run app_chromadb.py
```

---

5. PART 2.1: QUESTION GENERATION

Question Dataset (100 Q&A pairs)

Item	Details
File	data/questions_100.json
Total	100 Q&A pairs
GitHub	View File

Question Distribution

Type	Count	Percentage
Factual	59	59%
Multi-hop	15	15%
Comparative	15	15%
Inferential	11	11%

Adversarial Questions (30)

Item	Details
File	data/adversarial_questions.json
Total	30 adversarial questions
GitHub	View File

---

6. PART 2.2: EVALUATION METRICS

Mandatory Metric: MRR (URL-level)

Item	Details
Formula	MRR = (1/N) * sum(1/rank_i)
Implementation	evaluation/metrics.py
Score Range	0-1 (higher is better)

Custom Metric 1: BERTScore

Item	Details
Model	bert-base-uncased
Measures	Semantic similarity between generated and reference answers
Implementation	evaluation/metrics.py

Custom Metric 2: Recall@10

Item	Details
Formula	Recall@10 = relevant URLs in top-10 / total relevant URLs
Measures	Retrieval coverage quality
Implementation	evaluation/metrics.py

---

7. PART 2.3: INNOVATIVE EVALUATION

7 Innovation Techniques Implemented

Technique	Description	Implementation
Adversarial Testing	30 adversarial questions (ambiguous, negated, unanswerable)	adversarial_questions.json
Ablation Studies	Dense vs Sparse vs Hybrid comparison	run_evaluation.py
Error Analysis	Failure categorization (35% retrieval, 45% generation, 20% context)	ERROR_ANALYSIS.md
LLM-as-Judge	5-dimension evaluation (Accuracy, Completeness, Relevance, Coherence, Hallucination)	metrics.py
Confidence Calibration	ECE, MCE, Brier Score with calibration curves	innovative_eval.py
Novel Metrics	Entity Coverage, Answer Diversity, Hallucination Rate, Temporal Consistency	novel_metrics.py
Interactive Dashboard	Real-time metrics, chunk visualization, method comparison	app_chromadb.py

8. PART 2.4-2.5: PIPELINE AND REPORTS

Automated Pipeline

Item	Details
Script	evaluate_chromadb_fast.py
GitHub	View File

Run Command:

```
python evaluate_chromadb_fast.py
```

Report Generation

Report Type	File	GitHub Link
PDF Report	submission/05_reports/Hybrid_RAG_Evaluation_Report.pdf	Download
Markdown Report	submission/05_reports/Hybrid_RAG_Evaluation_Report.md	View

See Also

- SUBMISSION_REQUIREMENTS.md - Deliverables checklist and submission folder structure
- QUICK_ACCESS_LINKS.md - Direct links to all project files

Last Updated: February 8, 2026
Repository: https://github.com/vishalvishal099/Hybrid_RAG_System_with_Automated_Evaluation