

Linux Command Assistant (RAG task)

1. Objective

The objective of this project was to design and implement a Retrieval-Augmented Generation (RAG) system for Linux commands and related documentation.

The system enables semantic search and contextual answering over a structured dataset containing Linux commands, installation guides, fundamentals, troubleshooting steps, common scenarios, and package managers.

The project evolved from a pure embedding-based semantic search (Phase-1) into a fully developed RAG pipeline using a vector database (Phase-2).

2. Dataset Description

The dataset is stored in a structured JSON file named “**linux.json**”, containing the following sections:

- i. commands
- ii. installation_guides
- iii. linux_fundamentals
- iv. troubleshooting_guides
- v. common_scenarios
- vi. package_managers

3. Chunking Strategy

Each logical unit is converted into self-contained text chunks, including:

- Command core information (syntax, description, explanation)
- Command options
- Examples
- Permission modes
- Installation guide sections (overview, steps, troubleshooting, security)
- Linux fundamentals (directories, permissions, processes, operators)
- Troubleshooting workflows
- Common scenarios
- Package manager commands

Chunking example:

```
{'text': 'Command: ls\n\nSecurity considerations:\nRegular users can only see files they have\npermissions for. Use with caution when examining system directories.', 'metadata': {'type': 'command_security', 'command': 'ls'}}}
```

Outcome:

- i. Chunking ensures contextual completeness
- ii. Improves retrieval precision
- iii. Enables multi-section answers in RAG

4. Embedding Model Selection

Models Evaluated	Model Characteristics
BAAI/bge-m3	High-quality, multilingual, best semantic understanding
all-MiniLM-L6-v2	Lightweight, fast, lower semantic depth
all-MiniLM-L12-v2	Balanced speed and quality
all-mpnet-base-v2	Strong semantic similarity, moderate size

Final Choice: BAAI/bge-m3

Reason for choosing BAAI/bge-m3 as embedding model:

- i. Best semantic alignment for technical documentation
- ii. Performs well for both short and complex queries
- iii. Suitable for RAG systems despite higher initial download cost

5. Embedding Generation and Caching**Phase-1 Approach**

- i. All chunk embeddings were generated once
- ii. Stored locally as `embeddings.npy`
- iii. Avoided recomputation during query time

Phase-2 Improvement

- i. Migration to ChromaDB vector store
- ii. Embeddings are now:
 - Generated once
 - Persisted inside the vector database
 - Automatically loaded on subsequent runs

Key Change:

Locally stored embeddings as `embeddings.npy` is no longer required, Vector DB persistence replaces manual embedding storage.

6. ChromaDB Integration (Vector database)

Reason why chroma database was chosen as vector database :

- i. Persistent storage
- ii. Fast similarity search
- iii. Metadata filtering
- iv. Easy to create collections
- v. Scales beyond in-memory embeddings

Implementation Details

- i. Chunks are embedded using SentenceTransformers “BAAI/bge-m3” model
- ii. Stored in a persistent Chroma collection linuxDB
- iii. Collection is reused across runs
- iv. Embeddings are not recomputed if data already exists

Benefits

- i. Faster startup after first run
- ii. Clean separation of concerns
- iii. Production-ready retrieval layer

7. Retrieval Process

- i. **Query Encoding:** User queries are encoded using BGE-M3 embeddings, optimized for multilingual retrieval with enhanced performance
- ii. **Vector Search:** Query embeddings are searched against ChromaDB vector store using cosine similarity
- iii. **Top-K Retrieval:** Top 3 most relevant chunks are retrieved based on semantic similarity
- iv. **Context Assembly:** Retrieved chunks are assembled into a unified context for LLM processing
- v. **Response Generation:** Context is passed to Groq LLM for grounded response generation

8. LLM Integration in RAG pipeline

- **Provider:** Groq (Cloud API)
- **Model:** llama-3.3-70b-versatile

Why Groq for This Implementation:

- i. Extreme Low Latency: Sub-second response times for real-time Linux assistance
- ii. High-Quality Instruction Following: Excellent at adhering to strict context usage rules
- iii. 70B Parameter Model: Sufficient capacity for detailed technical explanations
- iv. Cost-Effective: Competitive pricing for production deployment
- v. Reliable API: High uptime and consistent performance

LLM Configuration:

- i. Temperature: 0.2 (Low for factual accuracy)
- ii. Max Tokens: 1024 (Adequate for comprehensive Linux explanations)
- iii. Retries: 2 automatic retries on failure
- iv. Timeout: 20 seconds per request

LLM Role in RAG Pipeline:

- i. Context-Only Responses: Answers derived exclusively from retrieved documents
- ii. No External Knowledge: Prevents information leakage beyond documentation
- iii. Structured Responses: Maintains technical accuracy with proper formatting
- iv. Error Handling: Graceful fallback when context is insufficient

Conversation Memory:

- i. Implementation: ConversationBufferMemory from LangChain
- ii. Memory Key: "chat_history" tracks complete conversation flow
- iii. Context Retention: Enables follow-up questions and multi-turn dialogues
- iv. Session Management: Memory persists throughout user session

9. Prompt Engineering

Key Prompt Rules:

- Use ONLY provided context
- If information is missing say: "I couldn't find relevant information in the knowledge base."
- Do not use external knowledge
- Do not explain beyond what is documented

Impact

- Prevents hallucinations
- Ensures answers reflect the dataset only
- Makes system behavior predictable and auditable

10. User Interface & Deployment Architecture

FastAPI with integrated HTML/CSS/JavaScript

API endpoints:

- GET /
- POST /ask
-

12. Results and Observations

- i. Vector database make the system performance efficient
- ii. User queries returned expected answer, no hallucinations at all
- iii. Fast API user interface was working smoothly

13. Conclusion

This project successfully demonstrates a production ready RAG system specialized for Linux documentation. The implementation achieves the critical balance between retrieval accuracy, response quality, and system efficiency.

By combining:

- Structured chunking
- High-quality embeddings
- Persistent vector storage
- Controlled answer generation

The system delivers accurate, contextual, and trustworthy responses while remaining efficient and extensible.

OUTPUT:

The screenshot shows a web browser window titled "Linux RAG Assistant" at the URL "127.0.0.1:8000". The interface includes a sidebar with "Linux Assistant" branding and links for "Common Commands", "File Permissions", and "Install Software". The main area displays a conversation history:

- You: Explain Linux file permissions
- Assistant: Assistant: Linux file permissions consist of user, group, and others permissions with read, write, and execute bits.
The permission bits are:
 - r (read) (4): File → View file contents, Directory → List directory contents
 - w (write) (2): File → Modify file contents, Directory → Create/delete files in directory
 - x (execute) (1): File → Execute as program, Directory → Access directory contentsThere are also special permissions:
 - setuid (s): File executes with owner's privileges
 - setgid (s): File executes with group's privileges, directories inherit group
 - sticky bit (t): Only file owner can delete files in directory
- You: syntax of grep command
- Assistant: Assistant: The syntax of the grep command is: grep [options] pattern [file...]
- You: how do i install linux on my virtual machine
- Assistant: Assistant: To install Linux on your virtual machine, follow these steps:
 1. Download and install virtualization software
 2. Create new virtual machine
 3. Allocate resources (RAM, CPU, disk)
 4. Attach Linux ISO as virtual CD/DVD
 5. Start VM and follow installation wizard

A text input field at the bottom contains the placeholder "Ask about Linux..." and a "Send" button.

The screenshot shows a continuation of the web interface from the previous screenshot:

- You: how do i install linux on my virtual machine
- Assistant: Assistant: To install Linux on your virtual machine, follow these steps:
 1. Download and install virtualization software
 2. Create new virtual machine
 3. Allocate resources (RAM, CPU, disk)
 4. Attach Linux ISO as virtual CD/DVD
 5. Start VM and follow installation wizard
 6. Install guest additions/tools
 7. Configure network settingsMake sure you have the following prerequisites:
 - VirtualBox, VMware, or Hyper-V installed
 - Linux ISO file
 - At least 20GB free disk space
 - 4GB RAM available for VM
- You: what did i asked you at the starting of our conversation
- Assistant: Assistant: You asked me to "Explain Linux file permissions" at the starting of our conversation.
- A text input field at the bottom contains the placeholder "Ask about Linux..." and a "Send" button.