Implementation of an Agentic RAG Framework for Comparative Analysis of Classical Islamic Corpora

Hossein Shafaei
Independent Researcher & AI Systems Developer
Mashhad, Iran
Email: Shafaeihossein0@gmail.com
GitHub: https://github.com/shafaeihossin-hub/Classical-Shia-AI-RAG

***Abstract*—This research introduces a localized Retrieval-Augmented Generation (RAG) system specialized for theological discourse. Using an Agentic Query Expansion mechanism with DeepSeek-R1, we developed a system capable of cross-referencing the Holy Quran, Al-Kafi, Nahj al-Balagha, and Musnad Ahmad ibn Hanbal. The architecture ensures data privacy via Docker and Qdrant.**

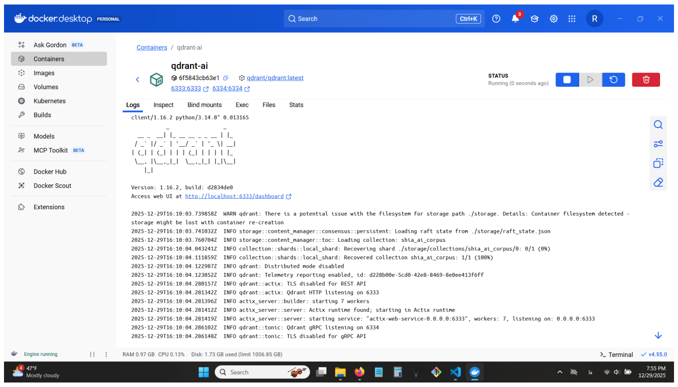
I. Introduction

Navigating classical theological texts requires high precision. Traditional LLMs often suffer from hallucination. This project implements a private RAG pipeline to ensure strict adherence to verified local contexts.

II. System Architecture

A. Infrastructure and Storage

The vector storage is handled by Qdrant, running within a Docker container to ensure environment isolation.



```
client/1.16.2 python/3.14.0" 0.013165
                   _                 _
  __ _  __| |_ __ __ _ _ __ | |_
 / _` |/ _` | '__/ _` | '_ \| __|
| (_| | (_| | | | (_| | | | | |_
 \__, |\__,_|_|  \__,_|_| |_|\__|
    |_|

Version: 1.16.2, build: d2834de0
Access web UI at http://localhost:6333/dashboard

2025-12-29T16:10:03.739858Z  WARN qdrant: There is a potential issue with the filesystem for storage path ./storage. Details: Container filesystem detected -
storage might be lost with container re-creation
2025-12-29T16:10:03.741032Z  INFO storage::content_manager::consensus::persistent: Loading raft state from ./storage/raft_state.json
2025-12-29T16:10:03.760704Z  INFO storage::content_manager::toc: Loading collection: shia_ai_corpus
2025-12-29T16:10:04.043241Z  INFO collection::shards::local_shard: Recovering shard ./storage/collections/shia_ai_corpus/0: 0/1 (0%)
2025-12-29T16:10:04.111859Z  INFO collection::shards::local_shard: Recovered collection shia_ai_corpus: 1/1 (100%)
2025-12-29T16:10:04.122987Z  INFO qdrant: Distributed mode disabled
2025-12-29T16:10:04.123052Z  INFO qdrant: Telemetry reporting enabled, id: d228b00e-5cd0-42e8-8469-8e0ee413f6ff
2025-12-29T16:10:04.280157Z  INFO qdrant::actix: TLS disabled for REST API
2025-12-29T16:10:04.281342Z  INFO qdrant::actix: Qdrant HTTP listening on 6333
2025-12-29T16:10:04.281396Z  INFO actix_server::builder: starting 7 workers
2025-12-29T16:10:04.281412Z  INFO actix_server::server: Actix runtime found; starting in Actix runtime
2025-12-29T16:10:04.281419Z  INFO actix_server::server: starting service: "actix-web-service-0.0.0.0:6333", workers: 7, listening on: 0.0.0.0:6333
2025-12-29T16:10:04.286102Z  INFO qdrant::tonic: Qdrant gRPC listening on 6334
2025-12-29T16:10:04.286148Z  INFO qdrant::tonic: TLS disabled for gRPC API
```

Fig. 1. Containerized Qdrant-AI environment in Docker.

B. Agentic Search Logic

The system uses an "Agentic" approach, generating additional search keywords to improve retrieval hit rate before querying the database.

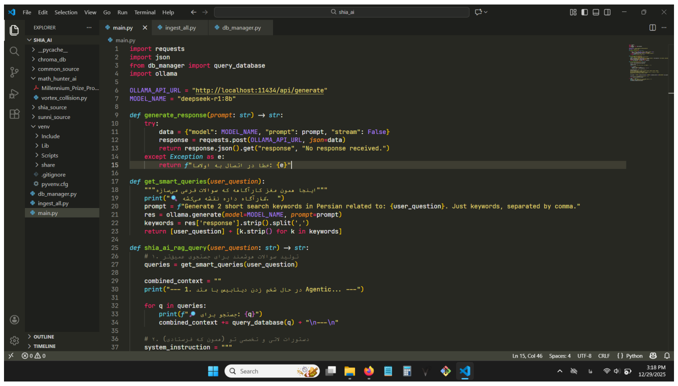
III. Implementation Details

Listing 1 demonstrates the core logic for query expansion and the RAG loop.

```python
def get_smart_queries(user_question):
    prompt = f"Generate 2 short search keywords in Persian related to: {
      user_question}."
    res = ollama.generate(model=MODEL_NAME, prompt=prompt)
    keywords = res['response'].strip().split(',')
    return [user_question] + [k.strip() for k in keywords]

def shia_ai_rag_query(user_question: str):
    queries = get_smart_queries(user_question)
    combined_context = ""
    for q in queries:
        combined_context += query_database(q) + "\n---\n"
    return generate_response(full_prompt)
```

Listing 1. Agentic Query Expansion Logic



```python
import requests
import json
from db_manager import query_database
import ollama

OLLAMA_API_URL = "http://localhost:11434/api/generate"
MODEL_NAME = "deepseek-r1:8b"

def generate_response(prompt: str) -> str:
    try:
        data = {"model": MODEL_NAME, "prompt": prompt, "stream": False}
        response = requests.post(OLLAMA_API_URL, json=data)
        return response.json().get("response", "No response received.")
    except Exception as e:
        return f"خطا در اتصال به اولاما: {e}"

def get_smart_queries(user_question):
    """اینجا همون مغز کارآگاهه که سوالات فرعی می‌سازه"""
    print("🔍 کارآگاه داره نقشه می‌کشه.  ")
    prompt = f"Generate 2 short search keywords in Persian related to: {user_question}. Just keywords, separated by comma."
    res = ollama.generate(model=MODEL_NAME, prompt=prompt)
    keywords = res['response'].strip().split(',')
    return [user_question] + [k.strip() for k in keywords]

def shia_ai_rag_query(user_question: str) -> str:
    # ۱. تولید سوالات هوشمند برای جستجوی عمیق‌تر
    queries = get_smart_queries(user_question)

    combined_context = ""
    print("--- 1. در حال شخم زدن دیتابیس با متد Agentic... ---")

    for q in queries:
        print(f"🔎 جستجو برای: {q}")
        combined_context += query_database(q) + "\n---\n"

    # ۲. دستورات لاتی و تخصصی تو (همون که فرستادی)
    system_instruction = """
```

File Edit Selection View Go Run Terminal Help
shia_ai
EXPLORER
main.py
ingest_all.py
db_manager.py
SHIA_AI
OUTLINE
TIMELINE
Ln 15, Col 46 Spaces: 4 UTF-8 CRLF { } Python
3:18 PM
12/29/2025

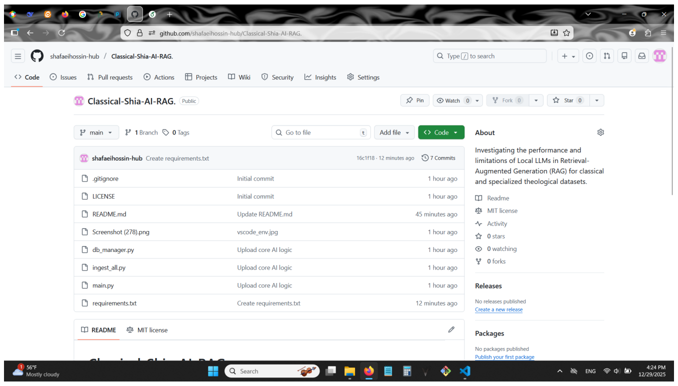
Fig. 2. Implementation of the RAG pipeline in VS Code.

IV. Project Management

The project is open-sourced under the MIT License and hosted on GitHub.

V. Conclusion

This study validates that Small Language Models (SLMs) can achieve professional-grade performance in specialized fields when paired with intelligent retrieval mechanisms.



github.com/shafaeihossin-hub/Classical-Shia-AI-RAG.

shafaeihossin-hub / Classical-Shia-AI-RAG.

Classical-Shia-AI-RAG. Public

main · 1 Branch · 0 Tags

shafaeihossin-hub Create requirements.txt · 16c1f18 · 12 minutes ago · 7 Commits

.gitignore	Initial commit	1 hour ago
LICENSE	Initial commit	1 hour ago
README.md	Update README.md	45 minutes ago
Screenshot (278).png	vscode_env.jpg	1 hour ago
db_manager.py	Upload core AI logic	1 hour ago
ingest_all.py	Upload core AI logic	1 hour ago
main.py	Upload core AI logic	1 hour ago
requirements.txt	Create requirements.txt	12 minutes ago

About

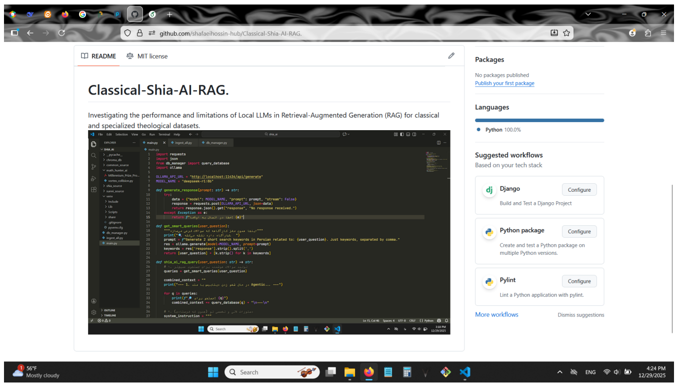
Investigating the performance and limitations of Local LLMs in Retrieval-Augmented Generation (RAG) for classical and specialized theological datasets.

Readme · MIT license · Activity · 0 stars · 0 watching · 0 forks

Releases: No releases published. Create a new release

Packages: No packages published. Publish your first package

Fig. 3. Project structure and version control on GitHub.



github.com/shafaeihossin-hub/Classical-Shia-AI-RAG.

Classical-Shia-AI-RAG.

Investigating the performance and limitations of Local LLMs in Retrieval-Augmented Generation (RAG) for classical and specialized theological datasets.

```python
import requests
import json
from db_manager import query_database
import ollama

OLLAMA_API_URL = "http://localhost:11434/api/generate"
MODEL_NAME = "deepseek-r1:8b"

def generate_response(prompt: str) -> str:
    try:
        data = {"model": MODEL_NAME, "prompt": prompt, "stream": False}
        response = requests.post(OLLAMA_API_URL, json=data)
        return response.json().get("response", "No response received.")
    except Exception as e:
        return f"خطا در اتصال به اولاما: {e}"

def get_smart_queries(user_question):
    """اینجا همون مغز کارآگاهه که سوالات فرعی می‌سازه"""
    print("🔍 کارآگاه داره نقشه می‌کشه...")
    prompt = f"Generate 2 short search keywords in Persian related to: {user_question}. Just keywords, separated by comma."
    res = ollama.generate(model=MODEL_NAME, prompt=prompt)
    keywords = res['response'].strip().split(',')
    return [user_question] + [k.strip() for k in keywords]

def shia_ai_rag_query(user_question: str) -> str:
    # ۱. تولید سوالات هوشمند برای جستجوی عمیق‌تر
    queries = get_smart_queries(user_question)

    combined_context = ""
    print("--- 1. در حال شخم زدن دیتابیس با متد Agentic... ---")

    for q in queries:
        print(f"🔎 جستجو برای: {q}")
        combined_context += query_database(q) + "\n---\n"

    # ۲. دستورات لاتی و تخصصی تو (همون که فرستادی)
    system_instruction = """
```

Packages: No packages published. Publish your first package

Languages: Python 100.0%

Suggested workflows — Based on your tech stack

Django	Build and Test a Django Project
Python package	Create and test a Python package on multiple Python versions.
Pylint	Lint a Python application with pylint.

Fig. 4. Project Documentation and README details.