Utilizing Class Assignment's Database For RAG(Retrieval augmented generation) Based Response Generation

Anurag Dhiman (MA25M004) Tanmoy Ghosh (MA25M026) Rahul Ghosh (MA25M021) Ajay Kumar (MA25M003)

Department of Mathematics Indian Institute of Technology Madras

November 7, 2025

Introduction & Motivation

Problem Statement:

- Academic assignments hold structured, domain-rich knowledge.
- However, retrieving this information manually is inefficient.
- Traditional keyword-based searches (TF-IDF, BM25) fail to capture meaning.

“‘ **Emergence of Semantic Retrieval:**

- **Sentence-BERT:** Encodes text into dense semantic vectors.
- **RAG (Lewis et al., 2020):** Combines retrieval with generative models.

Our Approach

RAG System — a retrieval-augmented generator that uses an assignment database to provide fact-based, contextually grounded responses.

Technology Stack

ChromaDB (Vector DB), **Sentence-Transformers** for embeddings, **distilGPT2** for generation.

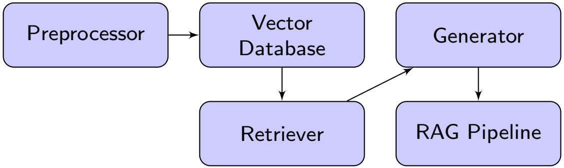
Key Goals:

- Reinforce learning with AI-driven contextual support.
- Apply OOP for modular, maintainable,

System Architecture & OOP Design

Component Architecture:

‘‘‘



Data Flow:

- User Query → Retriever → Vector DB → Generator → Answer.
- Output: Contextually grounded and factually consistent answer.

‘‘‘

OOP Principles Implemented:

- **Encapsulation:** Each class manages its own data and logic.
- **Abstraction:** Simple high-level interface (`RAGPipeline.run()`).
- **Composition:** Pipeline integrates multiple modular classes.
- **Reusability:** Components usable in other educational AI systems.

Design Philosophy

Clear separation of concerns ensures flexibility, clarity, and future scalability.

Data Preprocessing & Embedding Generation

Assignment Preprocessing:

1. Extract text from `.py` and `.ipynb` files.
2. Remove metadata and non-informative code cells.
3. Split text into smaller, coherent chunks (50–150 words).

‘‘‘

Dataset Summary

10 assignments $\rightarrow$ approximately 300 processed text chunks.

Core Tools:

- `nbformat`, `chromadb`, `sentence-transformers`, `transformers`, `torch`.

‘‘‘

Semantic Embeddings:

$$v_i = f_\theta(c_i), \quad v_i \in \mathbb{R}^{384} \tag{1}$$

- c_i: text chunk; f_θ: transformer encoder.
- Each embedding represents semantic meaning.
- Enables similarity search via cosine similarity.

Model Used

SentenceTransformer: *all-MiniLM-L6-v2*

Retrieval & Answer Generation

Retrieval Process:

$$v_q = f_\theta(q), \quad \mathrm{sim}(v_q, v_i) = \frac{v_q \cdot v_i}{|v_q||v_i|} \tag{2}$$

- Compute embedding for user query v_q.
- Compare with stored embeddings v_i.
- Retrieve top-k relevant chunks $\mathcal{C}_k(q)$.

"'

Retriever

Uses **cosine similarity** for semantic search in vector space.

"'

Generation Phase:

$$P_\phi(y|q, \mathcal{C}_k(q)) = \prod_t P_\phi(y_t|y_{<t}, q, \mathcal{C}_k(q)) \tag{3}$$

- Model: **distilGPT2**.
- Input: Query + Retrieved Context.
- Output: Grounded and coherent response.

Key Strength: Combines retrieval accuracy with generative flexibility.

Results, Conclusion & Future Work

System Performance:

Query	Retrieved Context	Generated Answer
		What is the time complexity of quick sort?
Quick sort partitions the array. Average case O(n log n), worst case O(n^2).	Quick sort has O(n log n) average and O(n^2) worst case complexity.	Breadth-first search uses queue, while DFS uses stack.
	What data structure does BFS use?	
Breadth-first search uses a queue for traversal.		

‘‘‘

Key Achievements:

- Demonstrates RAG using academic assignments.
- Object-Oriented framework ensures modularity.

References:

- Lewis, P., et al. (2020). Retrieval-Augmented Ger
for Knowledge-Intensive NLP Tasks. ACL.
- https://huggingface.co/docs/transformers.