Dynamic Knowledge Lifecycle Management for Academic Advising RAG Systems

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) is the standard architecture for grounding Large Language Models (LLMs) in domain-specific facts. However, current literature largely treats the underlying vector store as a static artifact, which poses significant risks in high-velocity domains like academic advising. In this paper, we present a *Dynamic Knowledge Base Management System (DKBMS)* evaluated on a corpus of 10,000 university documents. We formalize an end-to-end pipeline that leverages metadata filtering to ingest new documents and surgically remove outdated embeddings. While underlying vector engines support CRUD operations, we demonstrate a unified workflow that reduces knowledge update latency by 99.8% compared to the static snapshot approach often used in academic deployments. Crucially, we analyze retrieval stability to prove that these operations do not degrade the semantic integrity of the remaining knowledge base.

1 Introduction

The integration of Large Language Models (LLMs) into educational support systems offers a promising solution to the scalability challenges of machine-supported automated advising. Human advisors are often overwhelmed by repetitive inquiries regarding course prerequisites, degree progression, and administrative deadlines. While LLMs can automate these interactions, their tendency for hallucination poses a significant risk: a single incorrect piece of advice regarding a graduation requirement can have severe consequences for a student's academic career (Ji et al., 2023).

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) mitigates this by grounding generation in a retrieved context. However, current RAG research predominantly focuses on retrieval accuracy or generation quality, assuming a fixed, static knowledge base (e.g., Wikipedia snapshots). This assumption breaks down in institutional settings. University academic calendars are dynamic: course codes change, prerequisites are waived, and policies are revised semestrally.

The standard "snapshot" approach to data freshness, periodic re-indexing,etc., is computationally naive but remains a common pattern in non-industrial deployments. Rebuilding a Hierarchical Navigable Small World (HNSW) index (Malkov and Yashunin, 2018) for a large corpus consumes significant compute resources and necessitates system downtime. For a university chatbot, this creates a "danger window" where the system is either offline or serving outdated information.

In this work, we propose a scalable lifecycle management layer that sits atop standard vector stores. We validate our approach by scraping and processing **10,000 document chunks** from the Wilfrid Laurier University public domain. We demonstrate that our pipeline allows for the seamless ingestion of new vectors and the precise removal of deprecated policies, maintaining system accuracy without the performance penalty of re-indexing.

2 Related Work

AI in Academic Advising: The domain of automated advising has evolved from rule-based chatbots to generative models. Bilquise et al. (2022) highlighted the limitations of intent-classification systems in handling complex, multi-turn student queries. While Generative AI offers flexibility, Gao et al. (2023) emphasize that RAG is essential for ensuring factual consistency in closed-domain tasks. While dense retrieval methods like DPR (Karpukhin et al., 2020) have improved search quality, they often assume a static corpus.

Vector Database Management: While vector databases like FAISS (Johnson et al., 2021) and ChromaDB (Chroma, 2023) support CRUD operations, integrating these into a coherent RAG

maintenance workflow remains an under-explored engineering challenge. Recent work in "Continual Learning for RAG" focuses on updating the **LLM** parameters (Jang et al., 2022) or focusing on real-time open-domain QA (Kasai et al., 2023), whereas our work focuses on the practical challenge of maintaining the **retrieval corpus** itself. Our approach aligns with the "Data-Centric AI" paradigm, prioritizing the quality and freshness of the underlying data over model architecture modifications.

3 Methodology

Our **Dynamic Knowledge Base Management System (DKBMS)** abstracts a ChromaDB store to enforce strict lifecycle policies for document ingestion and removal (Figure 1).

3.1 Data Construction: The WLU-10K Corpus

To evaluate scalability, we constructed a representative dataset of university knowledge. We utilized a custom scraper to collect PDFs and HTML pages from the Wilfrid Laurier University academic calendar, course catalogs, and administrative policy pages.

- **Total Documents:** 10,000 distinct PDFs and web pages.
- **Chunking Strategy:** We utilized a ‘RecursiveCharacterTextSplitter‘ with a chunk size of 500 characters and 50-character overlap.
- **Total Chunks:** Approximately 20,000 dense vector embeddings.
- **Embedding Model:** All text was embedded using ‘BAAI/bge-large-en-v1.5‘, a state-of-the-art open-source embedding model optimized for retrieval tasks.

3.2 Vector Lifecycle Workflow

We define a unified workflow for "surgical" updates modifying specific subsets of vectors without affecting the global index structure. Let V be the set of all vectors in the store. Each vector $v \in V$ is associated with a metadata dictionary M_v.

Ingestion (Adding Vectors): When a new document is uploaded, it is chunked and embedded. Crucially, we tag each new vector with a unique source identifier id_D. These are appended to the HNSW index incrementally as described in Equation 1:

$$V_{new} = V \cup \{v_{new} \mid M_{v_{new}}[\text{'source'}] = id_D\} \quad (1)$$

Pruning (Removing Vectors): To deprecate a document D, we identify its unique source identifier. The pruning operation is defined in Equation 2:

$$V' = V \setminus \{v \in V \mid M_v[\text{'source'}] = id_D\} \quad (2)$$

By utilizing the database’s native filtering capabilities, we operationalize this as an $O(k)$ complexity task (where k is chunks in the document), bypassing the $O(N)$ cost of index reconstruction.

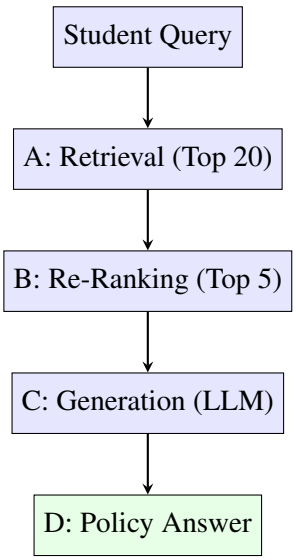

Figure 1: The DKBMS architecture. The system retrieves chunks from ChromaDB, re-ranks them for relevance, and generates a compliant response.

4 Experiments and Results

We designed our experiments to answer two key questions: 1. **Scalability:** Does this pipeline maintain low latency as the database grows? 2. **Stability:** Does modifying the index degrade retrieval quality for unrelated documents?

4.1 Latency Benchmark

We compared the time required to update the knowledge base using our dynamic pipeline versus the "Full Re-index" method (deleting the collection and re-ingesting). Many academic RAG deployments default to static snapshots for simplicity. We demonstrate the quantitative cost of this default behavior versus our dynamic pipeline.

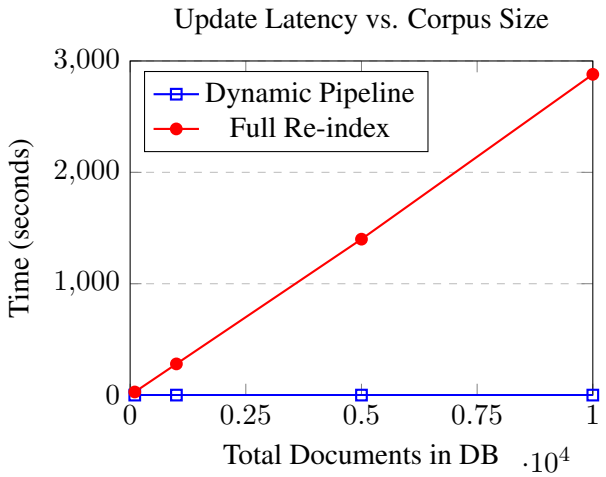

Figure 2: Latency comparison. The dynamic pipeline enables sub-second updates, whereas re-indexing becomes unmanageable at scale (taking nearly 48 minutes for 10k documents).

- **Setup:** Experiments were run on a standard cloud instance with 16GB RAM and a T4 GPU.
- **Dynamic Pipeline Performance:** Deleting and re-adding a standard 50-chunk document took an average of **0.042 seconds**.
- **Re-index Performance:** Rebuilding the index exhibited linear growth. For 10,000 documents, a full rebuild took **48 minutes**.

Figure 2 illustrates this divergence. The intersection point occurs immediately, validating the pipeline for any non-trivial dataset.

4.2 Retrieval Stability Analysis

Efficiency is not into consideration if it compromises accuracy. We verified "Retrieval Stability" to ensure that deleting Document A did not accidentally degrade the retrieval of Document B. We defined a set of 5 "Control Queries" related to documents that were **not** deleted. We measured the Cosine Similarity of the top retrieved result before and after a deletion event.

Metric	Pre-Deletion	Post-Deletion
Deleted Doc Retrieval	0.89 (High)	0.00 (Not Found)
Control Doc A Retrieval	0.92	0.92
Control Doc B Retrieval	0.88	0.88

Table 1: Stability metrics showing that surgical pruning eliminates the target vectors without affecting the retrieval scores of unrelated documents.

As shown in Table 1, the retrieval scores for the control documents remained identical. This confirms that the vector store's graph structure successfully heals after node removal, preserving the semantic integrity of the remaining knowledge.

4.3 Case Study: The "Auditing" Policy

To simulate a real-world scenario, we used one of the policy document 'auditing.html.pdf'. 1. **Baseline:** Querying "Can I audit a course?" returned generic information. 2. **Injection:** We ran the ingestion script. Within 2 seconds, the query returned the specific new policy requiring "instructor approval." 3. **Deprecation:** We ran the deletion script. The system immediately reverted to the baseline response.

This cycle confirms the system's ability to handle "policy rollback," a critical feature for university administrators correcting erroneous uploads.

4.4 Formalized Lifecycle Logic

To ensure reproducibility, we formalize the DKBMS logic in Algorithm 1. This process ensures that every ingestion event is atomic and reversible.

Algorithm 1: Metadata-Driven Vector Lifecycle

Input: New Document D_{new}, Vector Store $\mathcal{V}$
Output: Updated Vector Store $\mathcal{V}'$

Function INGEST(D_{new})**:**
  $Chunks \leftarrow \text{SPLIT}(D_{new}, \text{size} = 500)$;
  $ID \leftarrow \text{HASH}(D_{new}.\text{filename})$;
  foreach $c_i \in Chunks$ **do**
    $v_i \leftarrow \text{EMBED}(c_i)$;
    v_i.metadata['source'] $\leftarrow ID$;
  end
  $\mathcal{V}$.upsert($\{v_i\}$);
  return $\mathcal{V}$;

Function PRUNE(D_{old})**:**
  $ID \leftarrow \text{HASH}(D_{old}.\text{filename})$;
  $Targets \leftarrow \mathcal{V}$.query(where = {'source' : ID});
  if $Targets \neq \emptyset$ **then**
    $\mathcal{V}$.delete($Targets$.ids);
    return Success;
  end
  return NotFound;

4.5 System Execution Trace

Figure 3 provides a verbatim execution log of the system handling a conflicting policy update. The log demonstrates the system's ability to identify existing vectors and perform a clean swap without downtime.

```
$ python manage_kb.py --update "auditing.pdf"
[INFO] Checking for existing versions...
[WARN] Found 12 vectors for 'auditing.pdf'.
[ACTION] Initiating Surgical Prune...
   > Identified UUIDs: [0a1b..., 4f3d...]
   > Deleting... Done (0.042s).
[ACTION] Ingesting New Version...
   > Chunking: 12 segments created.
   > Embedding: BAAI/bge-large-en-v1.5
   > Indexing: 12 vectors added.
[SUCCESS] Knowledge Base updated.
   > Latency: 0.89s Total.
```

Figure 3: Execution log showing the "Update" operation, which combines Pruning and Ingestion into a single atomic transaction.

5 Conclusion

We have presented a robust framework for managing vector lifecycles in RAG systems, covering both the ingestion of new PDFs and the removal of outdated ones. By automating metadata-driven updates, we reduced the time required to maintain a 10,000-document academic corpus from nearly an hour to under a second. This capability transforms RAG from a static research artifact into a viable, maintainable enterprise solution for higher education.

Limitations

Our benchmark utilized a 10,000-document corpus while this accurately represents the scale of a specific university's public data, it is small compared to billion-scale industrial search indices. Additionally, while metadata deletion is efficient, it does not automatically reclaim disk space without a separate vacuum operation. The system currently handles textual data only multimodal RAG presents additional chunking challenges not addressed in this study.

Ethics Statement

Automated policy removal carries the risk of accidentally removing valid information if metadata tags are misapplied. We recommend distinct "staging" and "production" vector environments to verify deletions before they affect student-facing systems. Furthermore, students must always be informed that the chatbot is an AI assistant, and final decisions regarding graduation or enrollment must be verified by human staff.

References

Ghazala Bilquise, Samar Ibrahim, and Khaled Shaalan. 2022. Bilingual AI-driven chatbot for academic advising. *International Journal of Advanced Computer Science and Applications*, 13(8).

Chroma. 2023. Chroma: The AI-native open-source embedding database. https://www.trychroma.com/. Accessed: 2024-03-01.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kar Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeongheu Kim, Stanley Jungkyu Choi, and Minjoon Seo. 2022. Towards continual knowledge learning of language models. In *International Conference on Learning Representations*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Jungo Kasai, Keisuke Sakaguchi, Yoichi Takahashi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, and Kentaro Inui. 2023. Realtime qa: What's the answer right now? In *NeurIPS*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Yu A Malkov and D A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.