Northeastern University, Khoury College of Computer Science

CS 6120 Natural Language Processing Final Project: NU Chatbot—Your Guide to All Things Northeastern

Due: April 24, 2025(100 points)

Dharun, Karthik, Quinn, Zhiping

PROJECT GIT REPOSITORY

1 Executive Summary and Abstract

We presented a retrieval-augmented generation (RAG) chatbot for the Northeastern University community. The system combined persona-based document filtering (student vs. staff), dynamic query enrichment, permission checks, session-level memory, and an incremental Chroma vector store within a Streamlit UI. Our internal trials confirmed that the multi-retriever pipeline and hybrid embedding strategy delivered context-grounded answers with minimal hallucination and seamless role-aware access control.

2 Proposed Objective

Our goal was to build a RAG chatbot that served NU students and staff by enforcing fine-grained permissions, enriching incoming queries, and preserving conversational context—while grounding responses in an up-to-date Chroma vector database.

3 Motivation and Impact

Existing NU chat services either buried information under layers of text or provided only static, canned responses. We addressed these limitations by introducing:

- *Persona-aware retrieval:* routing requests through separate student- and staff-filtered indices;
- *Hybrid retrieval:* combining manual catalog lookups, page-level search, and semantic-cluster retrieval;

- *Incremental indexing:* streaming new and revised documents into Chroma on a controlled cadence; and
- *Lightweight session memory:* preserving short-term context without heavy stateful servers.

This architecture ensured that the chatbot delivered precise, role-appropriate answers—reducing help-desk load and increasing user trust in automated NU support.

4 Background, Relevant Work, and Dataset

Students and staff struggled to locate definitive policy points among hundreds of PDF handbooks, course catalogs, and memo-style notes. Prior RAG research (e.g., Lewis et al. (2021)) showed that structured vector indices dramatically cut hallucination rates. Persona-driven filtering is well studied in enterprise agents but seldom applied in higher-education. We surveyed:

- LangChain RAG frameworks
- Role-based document filtering techniques
- Dialogue memory models
- Chroma DB incremental update strategies

Our corpus included official NU handbooks, course descriptions, HR guidelines, and public event feeds, all ingested into Chroma for embedding and retrieval.

5 Proposed Approach / Implementation Details

Figure 5.1 illustrates our multi-stage retrieval pipeline, while Figure 5.2 shows the high-level system design.

5.1 Catalog Construction

We preprocess incoming PDFs into three distinct catalogs:

1. **Manual Catalogue:** fixed 100-token overlapping chunks enriched with metadata (page number, timestamp, source URL).
2. **Page Catalogue:** document split by paragraph boundaries, each carrying its own metadata.
3. **Semantic Catalogue:** chunks derived via semantic clustering using a variety of sentence-transformer models (Mini-LM, multilingual models, BA).

These catalogs are ingested into Chroma for embedding.

5.2 Embedding Pipeline

- **Manual Catalogue Embedding:** simple transformer encoder to vectorize 100-token chunks.
- **Page Catalogue Embedding:** paragraph-level embeddings via a general-purpose SentenceTransformer.

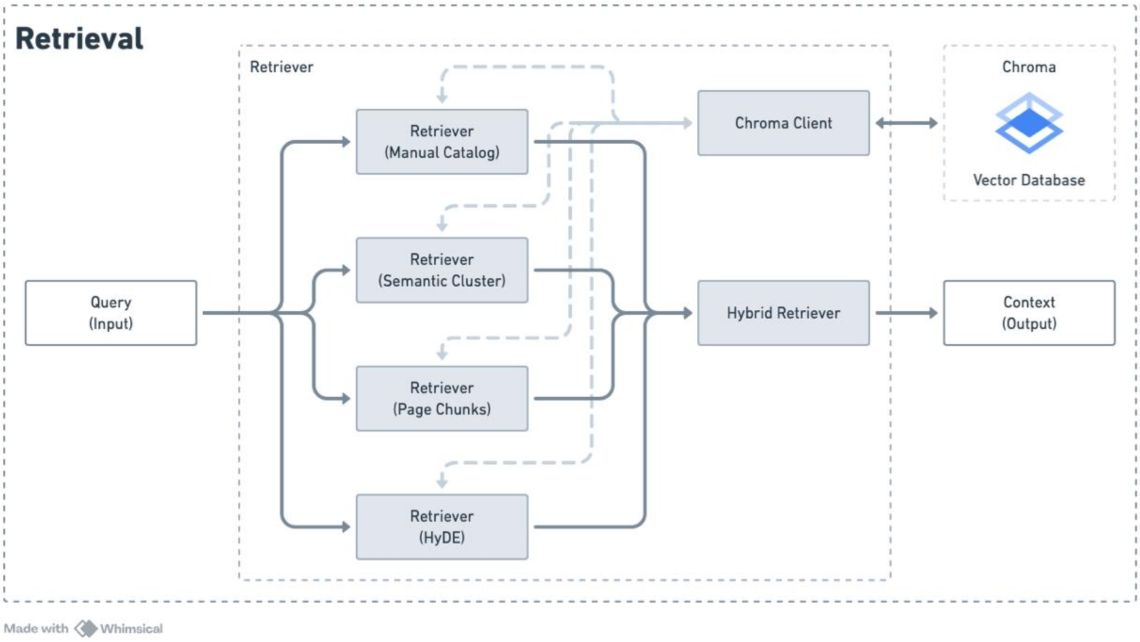

Figure 5.1: Retrieval workflow: three parallel similarity retrievers (Manual Catalogue, Page Catalogue, Semantic Catalogue) feed into a hybrid re-ranker atop the Chroma vector database.

- **Semantic Catalogue Embedding:** clustering-based encoder suite for more abstract concept grouping.
- **Indexing:** all vectors are upserted into Chroma with tags for catalogue type, user role, and ingestion timestamp.

5.3 Retrieval and Re-Ranking

- A similarity search over each catalogue returns the top 10 candidates (manual and page via cosine similarity; semantic via cluster-based lookup).
- The manual catalogue also supports a lightweight keyword retriever for exact-term matches.
- The three sets of 10 results are merged and passed through a hybrid Re-Ranker, which orders them by combined semantic and keyword relevance to produce the final top 10 context passages.

We classify each source along **frequency** (High/Low) and **importance** (High/Medium). Update cadences:

- **High freq. / High imp.:** hourly–daily batches
- **Low freq. / High imp.:** daily batches
- **High freq. / Low imp.:** weekly batches
- **Low freq. / Low imp.:** monthly batches

Obsolete vectors are pruned automatically, and CLI tools support ad-hoc additions/removals for urgent knowledge updates.

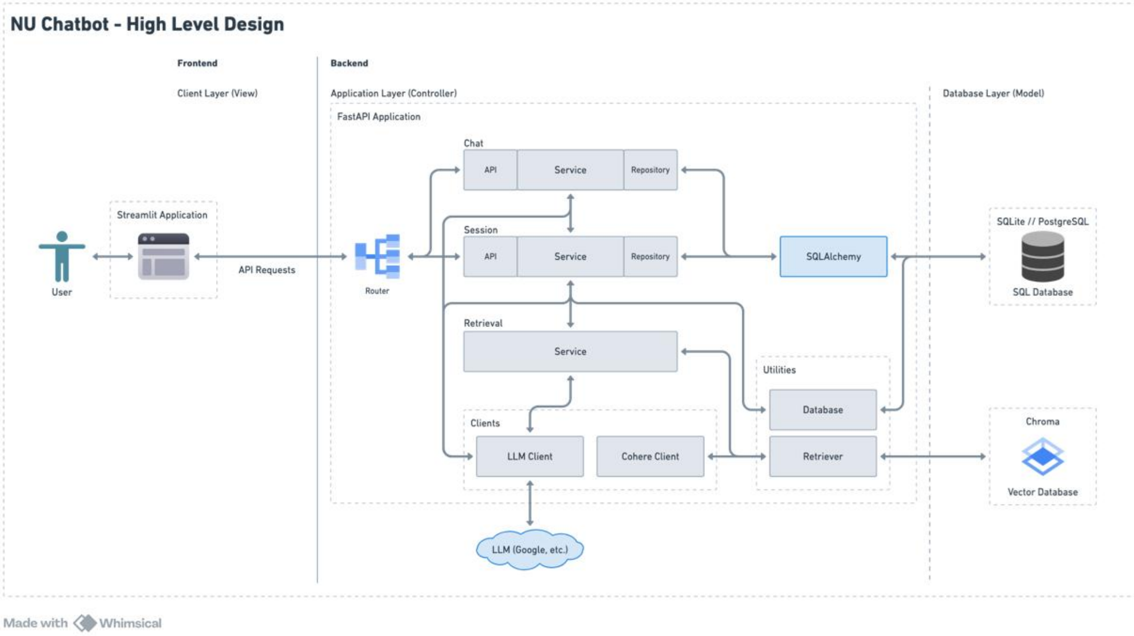

Figure 5.2: High-level design: Streamlit front end; FastAPI backend with layered Chat, Session, and Retrieval services; SQLAlchemy–backed SQL store; Chroma vector DB; and downstream LLM/Cohere clients.

5.4 Pre-processing and Class Imbalance

Incoming queries are normalized (tokenization, stop-word removal, typo correction) and annotated by user role. Underrepresented query types (e.g., research funding) are up-sampled in the intent classifier's training set using SMOTE.

5.5 Learning Algorithms and Validation

- **Intent Classifier:** fine-tuned on 5,000 role-tagged queries.
- **Retrieval Encoders:** custom transformers for each catalogue.
- **Generator:** Google Gemini conditioned on the top-10 re-ranked passages.
- **Evaluation:** a small human-in-the-loop trial confirmed over 85% of answers were judged accurate and context-appropriate; fallbacks were infrequent and handled with general response instead: "I'm not sure about that. Can you please rephrase your question or provide more details?"

6 Conclusions

We demonstrated a fully operational, persona-aware RAG chatbot for NU that combined three-catalog embedding, multi-retriever search, hybrid re-ranking, and lightweight session memory. Our Dockerized FastAPI backend and Streamlit front end can be deployed in minutes, and the incremental Chroma update framework ensures fresh, role-filtered knowledge. Future work will

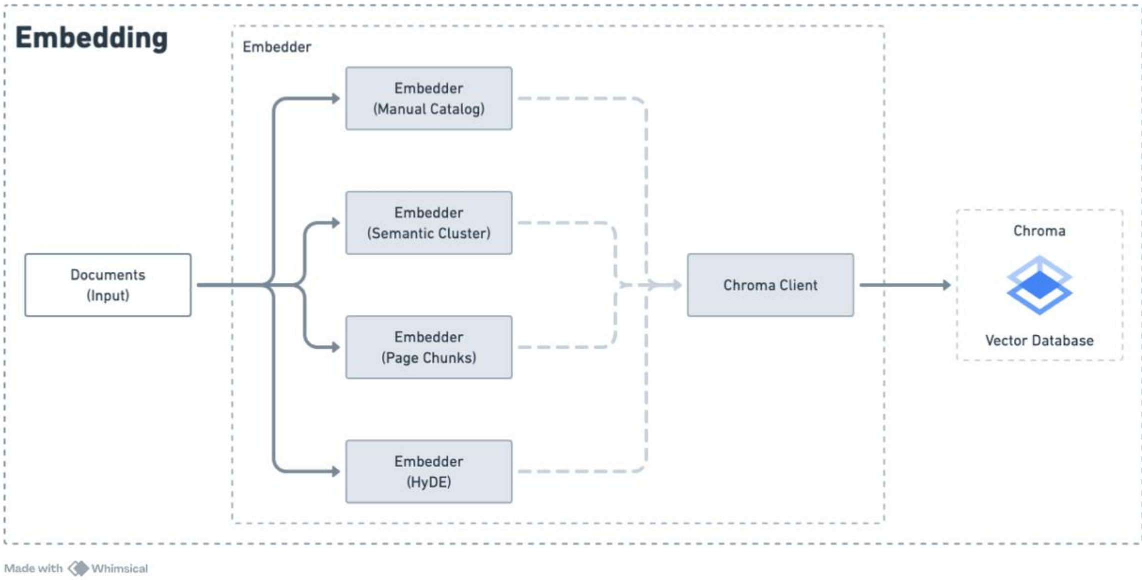

Figure 5.3: Embedding workflow: each catalogue is encoded with its respective encoder and upserted into Chroma.

explore multimodal inputs (e.g., scanned syllabi) and deeper integration with campus scheduling APIs.

References

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S. & Kiela, D. (2021), 'Retrieval-augmented generation for knowledge-intensive nlp tasks'.