SyllabusRAG: AI Academic Advisor

Cristina Papi, Marco Rosato, Rowyda Askalani

Abstract
Navigating academic syllabi can be a complex task for students, often involving the synthesis of fragmented information regarding course contents, prerequisites, and administrative rules. This project is about SyllabusRAG, a Retrieval-Augmented Generation (RAG) system designed to act as an AI Academic Advisor for the Data Science Master's degree at the University of Milano-Bicocca. Leveraging a small but capable Large Language Model (Qwen3-4B-Instruct) and a hybrid retrieval strategy (BGE-M3), the system addresses specific challenges in academic data, such as integrated course modules. Furthermore, we implement a conversational history module capable of query rewriting to handle context-dependent follow-up questions. Evaluation against a ground-truth dataset demonstrates that while standard RAG improves factual retrieval, our history-aware architecture achieves an 88.1% improvement in accuracy for context-dependent follow-up questions compared to stateless baselines.

Contents

1 Introduction

University syllabi contain dense information spread across tables and multiple web sections. Students often need to cross-reference this data, such as verifying prerequisites for an elective or understanding exam grading within integrated modules. Traditional keyword-based search struggles with these semantically rich queries. Recent advances in Large Language Models (LLMs) enable natural language interaction, but they are prone to hallucinations and lack domain-specific knowledge. Retrieval-Augmented Generation (RAG)[3] addresses this by grounding responses in external documents. This report presents SyllabusRAG, a system that applies established RAG techniques to academic syllabi through: structured semantic chunking, which respects course hierarchies (e.g., standalone courses vs. integrated modules), and history-aware query optimization, which rewrites user queries using conversation context to reduce ambiguity before retrieval.

2 Knowledge Base Construction

Steps of the Knowledge base construction

2.1 Data Enrichment

The effectiveness of a RAG system is heavily dependent on the quality of its knowledge base. The initial dataset contained the id, Course Name, Learning Objectives, Contents, Detailed Program and Source URL. To improve the quality of the RAG, we performed a manual data enrichment phase on the MongoDB[6] collection. Specifically, we enriched the database with the following key-value attributes:

- **Mandatory**: To distinguish between core curriculum and elective courses.
- **Year**: To identify if the course belongs to the first or second year
- **General Course Name**: Crucial for identifying the parent group of integrated modules.
- **Assessment Method**: To answer grading-specific queries.
- **ECTS**: Number of European credits.
- **Field of Research**: (e.g., INF/01, SECS-S/01) To classify the course.
- **Prerequisites**: Extracted to aid students in course sequencing.

2.2 Integrated Courses

A challenge is the concept of "Integrated Courses" which are exams composed of multiple modules (e.g., "Data Science Lab in Business and Marketing" containing "Big Data in Behavioural Psychology" and "Big Data in Business, Economics and Society"). A student asking about one module must be informed about the other to understand grading mechanisms. If a course is identified as part of a group, the system dynamically injects an "integration note" into the text chunk. This ensures that even if the LLM only retrieves the specific module's chunk, the context contains the necessary warning regarding the integrated exam.

2.3 Chunking Strategy

Initially, we implemented the chunking with "sliding window" (e.g., splitting text every 500 characters). However, applied to a syllabus, this approach is flawed; a chunk describing "Assessment Methods" might be separated from the course title, making it semantically ambiguous during retrieval. To address this, we implemented a "contextual anchoring" strategy combined with a fixed four-part semantic division.

2.3.1 Contextual Anchoring

We injected the course name into every text segment. This ensures that the generated embedding vector always encodes the specific subject identity alongside the content.

2.3.2 Semantic Segmentation

We construct four distinct chunks for every course document, mapping specific database fields to semantic categories:

- **Identity Chunk**: It combines the Course Name, General Course Name, Year, Mandatory status, ECTS Credits, and Field of Research code;
- **Content Chunk**: Combines the Contents and Detailed Program fields;
- **Objectives Chunk**: Isolates the Learning objectives, allowing the system to distinguish between topics covered and skills acquired;

- **Requirements & Assessment Chunk**: It combines Prerequisites and Assessment method. Furthermore, strictly for this chunk, we append the Integration Note.

2.4 Hybrid Embeddings and Storing

To create the embeddings of the chunks, we utilize the BGE-M3 model[1]. This model is chosen for its ability to generate two types of vectors:

- **Dense Vectors**: Capture semantic meaning.
- **Sparse Vectors (Lexical Weights)**: Extract keyword importance. This is crucial for academic data where specific codes or acronyms must be matched exactly.

After the creation of the embeddings, we stored them in a new MongoDB collection for an easier retrieval. This choice was made because the new collection is small and we didn't need a specific system to store the embeddings.

2.5 Embeddings Retrieval

The retrieval is made in two steps:

1. **Hybrid Search**: Retrieves the top 20 candidates using a weighted sum of Dense and Sparse scores.
2. **Reranking**: A Cross-Encoder (BGE-Reranker-v2[4]) re-scores the top candidates selecting the top 7 for the context window.

3 Conversational Engine

Construction of the SyllabusChatbot class

The core of the interaction is managed by "SyllabusChatbot" class. We used the Qwen3-4b-instruct[7] quantized to 4-bit via QLoRA/BitsAndBytes[2]. We wanted to create a chatbot such that if a user may ask, "Tell me about Deep Learning," followed by, "Is it mandatory?" the bot would be able to interpret the pronoun "it" and leading to relevant search results, unlike a bot that treats user queries as isolated, independent events.

3.1 Sliding Window

The SyllabusChatbot class manages the conversational state by appending "user-agent turns" to a cumulative history list. While this structure tracks the ongoing dialogue, feeding the entire sequence into the Language Model would exceed model's context window limits and saturate GPU memory. To address this, we implemented a "Sliding Window" mechanism during inference. Regardless of the total conversation length, the system slices the history to extract only the most recent 6 messages (representing the last 3 interaction turns). These selected turns are inserted into the generation prompt. This approach ensures that the model retains immediate context which is crucial for answering follow-up questions while maintaining a fixed token consumption.

3.2 Query Rewriting

To understand the user intent the system performs an intermediate inference step made through the implementation of the "contextualize_query" method. We designed a system prompt that optimizes the query (governed by strict logic: "do_sample=False"). It is instructed to perform three tasks:

- **Identify active topic**: It analyzes the chat history to determine the specific course discussed in the previous turn.
- **Pronouns change**: It replaces pronouns (e.g., "it", "that course") with specific en-

tities.

- **Detect topic shifts**: The prompt includes logic to detect when the user changes the subject.

4 Evaluation

Setup and Test set

4.1 Setup

To evaluate how well our model performs, we compared the following configurations:

1. **Baseline**: Qwen answering without the RAG system.
2. **RAG (No History)**: Retrieval enabled, but each query is treated in isolation.
3. **RAG (History)**: The full SyllabusChatbot implementation with query rewriting.

4.2 Test Set

We created two different test sets to evaluate the model in a single query evaluation and in a ”long conversation” (Following a natural conversation flow)

1. **Single Query**: we created 28 different questions and answer (that serves as ground truth). The questions ranges from general questions about the courses to ones with more specific details.
2. **Long Conversation**: we created a set of 10 different question and answer that recreate an hypothetical conversation of a user and a chatbot. This one was done, in particular, to test the memory system of the model.

5 Results

Models Metrics Comparison

We will compare three different metrics:

- **Cosine similarity**: Measures the vector alignment between the generated answer and the ground truth within the embedding.
- **BERTScore**[9]: Evaluates semantic overlap at the token level using a pre-trained language model. Unlike strict keyword matching, it accounts for synonyms and paraphrasing
- **Semantic Answer Similarity (SAS)**[8]: SAS employs a Cross-Encoder model (stsb-roberta-large)[5] to predict a similarity score based on the full sentence pair. It assesses whether the factual meaning of the answer is identical to the truth, regardless of significant differences in phrasing or structure.

Metric	Base	RAG (No H)	RAG (Hist)	Gain (No H)	Gain (Hist)
Cosine	0.6826	0.8115	0.7705	18.9%	12.9%
BERTScore	0.7505	0.8419	0.8103	12.2%	8.0%
SAS	0.6418	0.7067	0.6648	10.1%	3.6%

Table 1: First test: Factual Retrieval

The Baseline model performed poorly, confirming that without RAG, the LLM hallucinates specific syllabus details. The RAG (No History) configuration achieved the highest performance across all metrics. Interestingly, the RAG (History) model showed lower performance compared to RAG (No History). We think this can be an expected behavior because standalone questions require no context and the query rewriting step introduces potential noise without adding semantic value.

Metric	Base	RAG (No H)	RAG (Hist)	Gain (No H)	Gain (Hist)
Cosine	0.6670	0.7827	0.8110	17.3%	21.6%
BERTScore	0.7475	0.8159	0.8087	9.1%	8.2%
SAS	0.5969	0.6208	0.7449	4.0%	24.8%

Table 2: Second Test: Long Conversation

In this scenario, the limitations of the stateless RAG (No History) model became evident. Even if this model is always better than the baseline, the presence of some questions related to the context of the conversation, showed that the RAG (History) outperformed the RAG (No History) in the SAS score.

Metric	RAG (No H)	RAG (Hist)
SAS	0.4524	0.8509

Table 3: The Memory Test

To isolate the impact of the SyllabusChatbot logic, we filtered the conversation for specific follow-up questions that relied on context. When processing these questions, the RAG (No History) failed; treating the incomplete user inputs as generic searches. In contrast, the RAG (History) successfully re-wrote the query to resolve these ambiguities, merging the current query with the active conversational topic to target the correct document chunks. The resulting 88.1% performance gap confirms that context management is important to perform human like conversation.

6 Conclusions

SyllabusRAG wants addresses the complexities of academic syllabus by integrating a Retrieval-Augmented Generation system with a conversational engine designed for the Data Science master degree at the University of Milano-Bicocca. By relying on a semantic chunking strategy, the system effectively manages the data structures like integrated course modules. The architecture employs a hybrid retrieval method using BGE-M3 and a small LLM (Qwen3-4B-Instruct) to ensure factual grounding.

The implementation of a history-aware module with query rewriting allows the system to resolve ambiguous follow-up questions by leveraging conversation context. Evaluation results demonstrate that while stateless RAG models perform well on isolated queries, the SyllabusRAG system significantly outperforms them in multi-turn dialogues, achieving an 88.1% increase in accuracy for context-dependent interactions and proving the necessity of context-aware logic for effective academic assistance.

References

[1] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2025.

[2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[3] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

[4] Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. Making large language models a better foundation for dense retrieval, 2023.

[5] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[6] MongoDB Inc. Mongodb. https://www.mongodb.com/.

[7] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[8] Julian Risch, Timo Möller, Julian Gutsch, and Malte Pietsch. Semantic answer similarity for evaluating question answering models, 2021.

[9] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert, 2020.