Leveraging Large Language Models for Enhanced Student Advising

Kiran Nimmakayala
Center for AI Learning, Emory University

Abstract

This project investigates the application of large language models (LLMs) to augment the undergraduate business advising process at Emory University's Goizueta Business School. Traditional university advising involves complex interdependencies between courses, designations, and degree requirements, which challenge advisors to maintain comprehensive knowledge for a large student population. We propose a retrieval-augmented generation (RAG) approach that integrates both vector-based and graph-based retrieval methods, orchestrated through LangChain, to ensure accurate and contextually relevant responses to student queries. Our methodology involves extracting course and program information from academic catalogs and syllabi into structured JSON data, and incorporating insights from advisor interviews and student surveys via prompt engineering. Evaluation metrics, including response accuracy and query handling efficiency, indicate that the LLM-enhanced advising system significantly improves the accuracy of responses, student accessibility to information, and the overall efficiency of the advising process.

Keywords: Retrieval-Augmented Generation (RAG), Academic Advising, Knowledge Graph, Generative AI, Large Language Models

1 Introduction

Academic advisors are integral to higher education, offering essential guidance to students in navigating academic requirements and career pathways. Their role includes helping students choose appropriate courses, understand degree requirements, and explore career or enrichment opportunities. Institutions like Emory's Goizueta Business School (GBS) strive to provide effective advising, but with approximately 1,300 undergraduate students across 10 area depths (specializations), it is challenging for a limited number of advisors to have in-depth knowledge of every academic track.

This challenge is especially pronounced during peak advising periods (such as course registration or add/drop), when high demand makes it difficult for students to promptly connect with advisors. As a result, advisors often spend significant time searching through complex curriculum catalogs and checklists, limiting the opportunity for personalized guidance within the short duration of an advising appointment.

To address these challenges, we explore the use of LLM-based generative AI to enhance academic advising. Modern LLMs can produce human-like, context-dependent text and automate processes such as information extraction and answer generation. However, a critical issue with deploying LLMs in high-stakes domains like academic advising is their tendency to produce *hallucinations*—plausible-sounding but incorrect or misleading information—when they lack sufficient context or up-to-date data.

These inaccuracies undermine the reliability of purely LLM-based solutions in an advising context, where precision and trustworthiness are paramount. In this project, we mitigate these issues by incorporating retrieval mechanisms with the generative model, an approach known as retrieval-augmented generation (RAG). By integrating both graph-based and vector-based retrieval techniques with the LLM (using the LangChain framework), we aim to provide more accurate and contextually relevant information while minimizing hallucinations.

This hybrid approach has significant potential for assisting academic advisors by increasing the accessibility, accuracy, and efficiency of support provided to students.

2 Literature Review

2.1 Large Language Models and Hallucination

LLMs have become increasingly prominent across various sectors—including healthcare, customer service, and content generation—demonstrating a remarkable ability to generate human-like text and perform complex language tasks. In the education domain, educators and advisors have experimented with LLMs to generate course materials, provide automated feedback, and connect students to additional resources (Gan et al., 2023).

Despite these advantages, significant limitations remain, notably the phenomenon of *hallucination* (Peláez-Sánchez et al., 2024). Hallucination refers to an LLM generating information that appears credible but is factually incorrect or not grounded in the provided data. Since LLMs are trained on large text corpora and rely on learned patterns rather than true understanding, they may produce confident yet inaccurate responses. In academic advising, such errors can mislead students and erode trust in the system, as precise and reliable information is critical.

Addressing hallucinations in LLM outputs is an active area of research. While no definitive solution has been found to completely eliminate this issue, one promising direction is to augment LLMs with external knowledge sources so that the model's outputs are grounded in verified information. In particular, combining LLMs with retrieval mechanisms or knowledge bases has been shown to reduce hallucinations (Erdl, 2023). By supplying the model with relevant context from an authoritative data source, the likelihood of false or unrelated content in responses can be reduced.

2.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances LLM performance by coupling the generative model with an external retrieval component that fetches pertinent information from a knowledge source in real time. This approach, introduced by Lewis et al. (2020), ensures that the LLM's responses are supported by up-to-date, contextually relevant data rather than relying solely on the model's pre-trained knowledge.

In practice, RAG involves converting the user's query into one or more searches against a knowledge repository (such as a document database or knowledge graph), retrieving the most relevant pieces of information, and providing these to the LLM as additional context for answer generation. This method helps ground the generated content in factual data, making it particularly beneficial for high-stakes applications like our academic advising scenario.

There are two primary forms of retrieval used in RAG systems: vector-based retrieval and graph-based retrieval. In a *vector-based* approach, documents or text segments are embedded into high-dimensional vectors, and relevant content is retrieved via similarity search. This is effective for capturing semantic similarity and handling unstructured text (e.g., course descriptions or policy documents).

A *graph-based* approach leverages a knowledge graph (KG)—a structured representation of information where entities (nodes) are connected by relationships (edges) that encode semantic links. Knowledge graphs allow retrieval of information through explicit relationships and structured queries, which can yield highly precise answers for relational questions. Graph-based RAG is particularly powerful in scenarios requiring a deep understanding of interconnected data, as it can exploit rich semantic information encoded in the graph. However, constructing and maintaining a comprehensive KG is resource-intensive, and the effectiveness of graph-based retrieval depends heavily on the quality and completeness of the underlying graph data (Ji et al., 2021).

Each of these methods has its strengths: vector RAG excels at broad semantic searches and summarization tasks, while graph RAG provides exacting detail for queries that map well to structured data. By employing both methods in tandem, our project aims to capitalize on their complementary advantages. The combined RAG approach enhances the reliability and accuracy of LLM outputs in academic advising by using vector search to gather relevant unstructured information and graph queries to ensure consistency with official curricular requirements. This dual strategy directly addresses the challenge of hallucinations, ensuring that AI-generated advice is both precise and relevant to the student's query.

3 Stakeholder Analysis

Before designing our solution, we conducted a comprehensive stakeholder analysis to identify pain points in the existing advising process and to inform the requirements for the LLM-based advising system. We gathered input from both academic advisors and students at Emory to ensure the system would address the needs of its users and fit into the current advising workflow.

From the advisors' perspective, we interviewed specialized advising staff from GBS, the Office of Undergraduate Education (OUE), and faculty advisors in the Quantitative Theory and Methods (QTM) department. These discussions provided insights into common challenges advisors face. A recurring theme was that advisors rely heavily on personal experience, institutional memory, and key documents to answer student inquiries. For instance, QTM advisors often refer to detailed track checklists outlining required and elective courses, while GBS advisors frequently consult the BBA program catalog—an 83-page document covering program structures, course descriptions, and academic policies. This reliance on extensive documentation highlights the need for a tool that can streamline access to relevant information, reducing the time advisors spend searching through manuals and allowing more time for personalized student guidance.

On the student side, we interviewed over twenty undergraduate students to learn about the questions and issues they most commonly bring to advising sessions. From these interviews, we compiled a list of frequently raised topics and identified five top categories of advising queries:

1. **Course Selection and Requirements**: Choosing courses, understanding prerequisites, and planning area depth or major requirements.

2. **General Advising and Support**: Finding academic resources, scholarships, tutoring, or guidance on the graduation process.

3. **Special Programs and Opportunities**: Inquiries about study abroad options, accelerated degree programs, elective courses, or pursuing double majors.

4. **Academic Policies and Procedures**: Questions regarding academic deadlines, leave of absence, major/minor declaration or changes, and related procedures.

5. **Grading and Academic Standing**: Understanding grading policies, requirements for good academic standing, graduation honors criteria, or grade appeal processes.

These categories represent the areas where students most frequently seek guidance, underscoring the importance of providing accurate and accessible information through an AI-driven advising assistant.

Our stakeholder findings guided the scope of the project. Although the analysis covered multiple departments across Emory, we chose to focus our prototype on the GBS undergraduate program. This decision was driven by the availability of comprehensive, well-structured data sources for GBS (such as the BBA catalog) and the expertise of our faculty mentor in this domain. By initially concentrating on a single, data-rich environment, we were able to develop and refine a robust AI-driven advising solution. This targeted approach provides a strong foundation that can later be expanded to other departments or schools, ultimately creating a scalable tool for broader academic advising support.

4 Methodology

To effectively provide relevant information on university policies, degree requirements, and course details, our system employs a RAG architecture that combines vector and graph retrieval. The overall implementation comprises three main components: (1) data collection and ingestion, (2) knowledge graph construction (with an accompanying vector database), and (3) system integration through LangChain for query processing and response generation. Figure 1 illustrates the implementation process of our RAG-based advising system.

4.1 Data Collection and Ingestion

The primary data source for our advising system is the GBS BBA Program Catalog, supplemented by several course syllabi from specific area depths provided by faculty. Based on the insights from the stakeholder analysis, we identified the key information needed to address common advising questions and then extracted those data from the source documents.

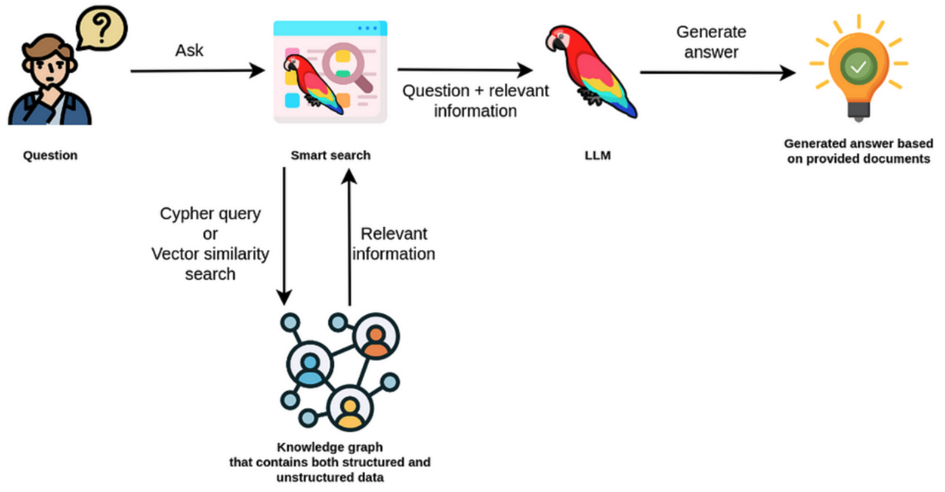

Question
Ask
Smart search
Question + relevant information
LLM
Generate answer
Generated answer based on provided documents
Cypher query or Vector similarity search
Relevant information
Knowledge graph that contains both structured and unstructured data

Figure 1: Knowledge graph-based RAG implementation process.

We employed a prompt-engineering approach using the OpenAI GPT-4 model to convert information from unstructured PDF documents into structured JSON format. The BBA catalog (a lengthy PDF) was split into smaller sections, each processed individually to ensure that no content was missed. We devised a two-stage extraction method: first, we segmented the catalog into manageable chunks and used the LLM to parse each segment into JSON; second, we structured the JSON output into around 30 predefined fields to capture detailed information while keeping the data ingestion efficient. Each JSON object recorded information in categorized fields (such as degree requirements, course prerequisites, etc.), which facilitated subsequent loading into the knowledge graph.

For individual course syllabi, we developed a specialized prompt to extract information into four key categories: basic course information, course objectives, course structure, and unique features. These categories cover the primary aspects of each course that students might inquire about. The use of a structured extraction schema helped standardize the data across courses.

Despite the largely successful automated extraction, the raw JSON data contained minor inconsistencies and omissions due to the variability in document formats and the limits of the model's parsing. We manually reviewed and corrected the extracted data to ensure accuracy and completeness before ingestion into the Neo4j database. This manual verification step was necessary to ensure that the knowledge base was reliable and reflected the most up-to-date information, thereby improving the trustworthiness of the advising agent's responses.

4.2 Knowledge Graph Development and Vector Database

After obtaining the structured JSON data, we constructed a knowledge graph using Neo4j to model the relationships among academic entities and enable efficient query operations. The knowledge graph serves as a dynamic representation of the academic program structure and requirements within the GBS context. Each node in the graph represents a specific entity or concept—such as a *Course*, *Minor*, or *Interdisciplinary Area*—and is labeled accordingly. Figure 2 provides a visualization of the knowledge graph schema, highlighting the core types of nodes and the relationships between them as defined in our dataset.

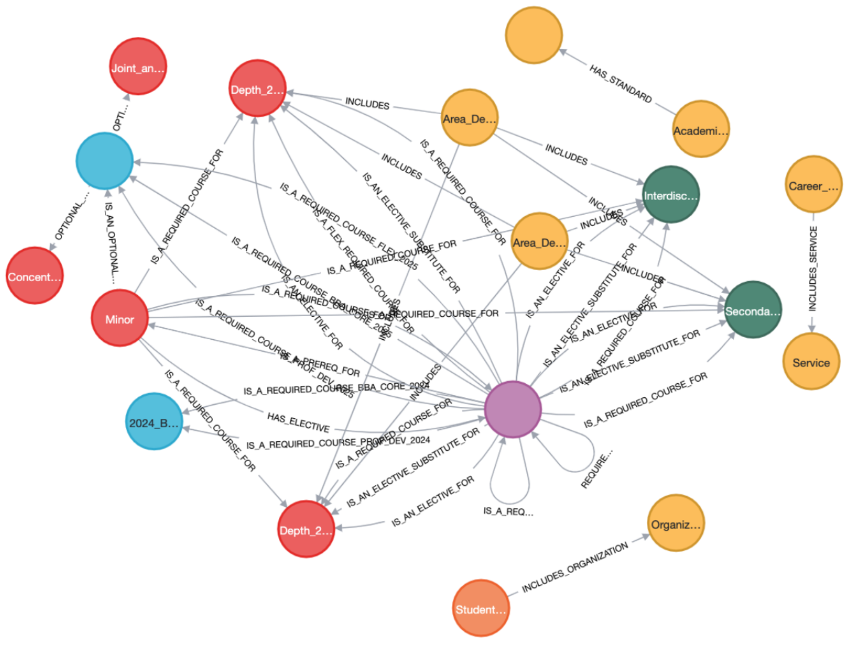


Figure 2: Knowledge graph schema visualization.

Within the knowledge graph, relationships link nodes to represent academic structures. For example, a course node may be connected to an interdisciplinary area node to indicate that the course is a requirement for that concentration; the same course node might connect to another course node to denote a prerequisite relationship, or to a minor node indicating it counts as an elective in that minor. This relational mapping allows the system to trace and retrieve pertinent academic pathways and requirements in response to user queries.

Each node in the graph carries properties that store detailed information relevant to advising. For instance, a node representing an academic area (analogous to a "major" at other institutions) might include properties such as the number of electives required and the maximum number of courses that can be substituted. These properties, along with the web of relationships, provide a comprehensive view of the curriculum rules and options, enabling the agent to deliver precise, personalized guidance.

In parallel with building the knowledge graph, we set up a vector database to index and retrieve unstructured textual information. We processed the documents (catalog sections, course descriptions, etc.) by splitting them into smaller text chunks and converting each chunk into a vector embedding using a pre-trained language model. These embeddings were stored in a Neo4j database (using its vector indexing capabilities) or an associated vector store. The vector database allows the system to perform semantic searches: given a user's query, the system can quickly find semantically similar text segments, which is especially useful for capturing context or details that are not explicitly encoded in the structured graph (for example, detailed course descriptions or examples given in policy documents).

4.3 RAG Integration with LangChain

We utilized the LangChain framework to integrate the knowledge graph, vector database, and LLM into a cohesive advising agent. LangChain provides tools for chaining together various components of a generative AI system (including language models, vector stores, and other agents) and proved instrumental in orchestrating our complex workflow.

A key feature of our implementation is the coordinated use of both the graph and vector knowledge sources for retrieval. When a student's query is received, the system employs a custom prompt template (via LangChain) to guide the LLM in how to use the external knowledge. The query is first analyzed to determine relevant keywords and entities. From there, two parallel retrieval strategies ensue:

1. **Vector-based retrieval**: The query (or its key components) is transformed into an embedding, which is then used to perform a similarity search against the vector database. The top-matching embedded text chunks are retrieved as relevant context.

2. **Graph-based retrieval**: In parallel, the system attempts to identify relevant nodes in the knowledge graph based on the query. This can involve recognizing specific course codes, program names, or topics mentioned by the user. For straightforward cases, the agent pulls all directly connected nodes (neighbors) of the identified node, gathering related information. For more complex questions, the agent can formulate a Cypher query (Neo4j's query language) via LangChain's natural language-to-query conversion, extracting specific node properties or traversing the graph for particular relationship patterns.

The information retrieved through these two methods is then combined and provided to the LLM as context for generating the final answer. By leveraging vector similarity search, the agent captures unstructured but relevant details (e.g., explanatory text from a handbook), while graph query results ensure that the answer respects the formal academic requirements and relationships (e.g., prerequisites and program rules). This dual retrieval mechanism enables the LLM to produce responses that are both comprehensive and grounded in the official curriculum data.

Using LangChain, we implemented prompt templates that structure the LLM's input to include the retrieved context and to instruct the model on how to formulate its answer (for example, prioritizing accurate factual information and citing the source document when appropriate). This setup effectively guides the LLM to synthesize the structured and unstructured data into a coherent response.

4.4 System Architecture

Figure 3 depicts the overall client–server architecture of our academic advising chatbot system. The design is centered around the integration of data pipelines, databases, and the LLM-driven query processor, rather than a complex user interface, as the project's primary focus is the backend intelligence.

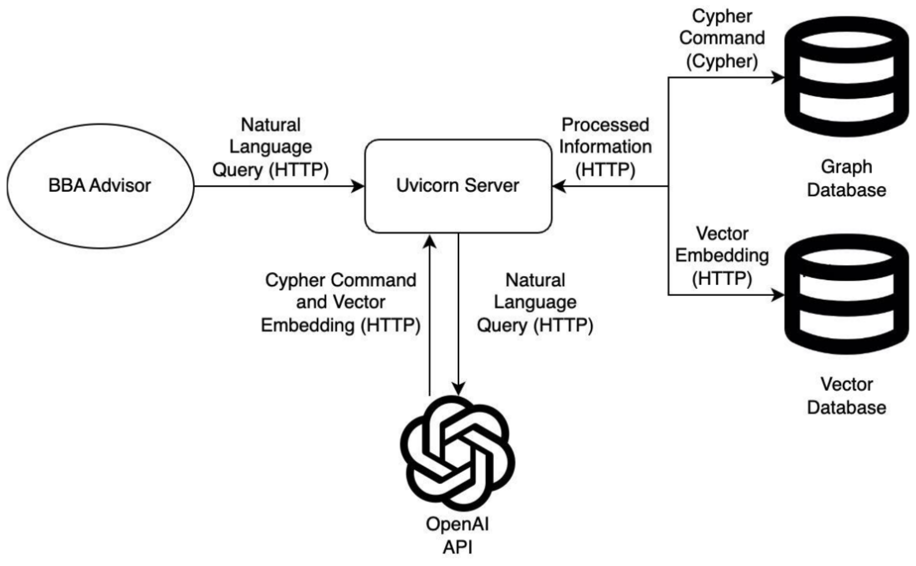

BBA Advisor
Natural Language Query (HTTP)
Uvicorn Server
Processed Information (HTTP)
Cypher Command (Cypher)
Graph Database
Vector Embedding (HTTP)
Vector Database
Cypher Command and Vector Embedding (HTTP)
Natural Language Query (HTTP)
OpenAI API

Figure 3: Client-server architecture of the academic advising agent prototype.

In the system's workflow, raw data (the BBA catalog and course syllabi) are first processed using the GPT-4 model through the OpenAI API. This step transforms unstructured documents into structured JSON data, as described earlier. The structured data is then loaded into the Neo4j Aura database to form the knowledge graph that encapsulates courses, programs (minors, area depths), and their interrelations. Simultaneously, textual content from these documents is encoded into vector embeddings and stored in the vector database for semantic retrieval.

When a user poses a question via the chatbot's GUI, the query is sent to the server, which coordinates the retrieval and generation processes. The server first consults the vector database to fetch contextually similar information and queries the knowledge graph (either by direct node lookup or via a generated Cypher query) to gather relevant structured data. This combined context is then passed to the LLM (through LangChain's interface) to generate a response.

The system also maintains a short-term memory of the dialogue (chat history) on the server, which helps in handling follow-up questions or clarifications within the same session. To maintain performance and relevance, this chat history is capped at a certain length or timeframe.

By using this dual-database architecture, the advising chatbot can draw upon detailed, interconnected knowledge as well as nuanced semantic context, providing answers that closely align with the student's needs. The outcome is an AI-driven advising assistant capable of delivering accurate, personalized academic guidance, thereby enhancing the overall advising experience for students and easing the load on human advisors.

5 Results and Discussion

5.1 Performance Analysis

We evaluated the performance of our advising chatbot in the context of the Information Systems and Operations Management (ISOM) area depth at GBS. The evaluation involved testing the system on a variety of typical advising questions and comparing the quality of answers produced using different retrieval configurations: using only the vector-based RAG, only the graph-based RAG, and the combined approach (which is the configuration of our final system). Key metrics of interest were the completeness and accuracy of the responses for each type of query.

In our tests, the vector-only RAG approach struggled with queries that spanned multiple sections of the curriculum or required piecing together information from different parts of the catalog. For example, when asked *"Which courses should I take if I want to pursue the ISOM area depth?"*, the vector method retrieved several relevant snippets but failed to compile them into a complete answer, omitting some required courses for the area depth. Likewise, when prompted for detailed information about a specific course (*"What does the ISOM 351 course cover?"*), the vector-based system provided some information (e.g., an excerpt of the course description) but missed other important details such as prerequisites or how it fit into the overall program.

The graph-only RAG approach, conversely, excelled at questions that were narrowly about structured requirements but had difficulty with more free-form queries. For a question like *"If I want to gain practical skills in data analytics (for example, using the DCT model), what courses should I take?"*, the graph method could identify related courses via the knowledge graph (e.g., courses labeled with certain skills or topics), but since not all such details were explicitly encoded as relationships or properties in the graph, the answer lacked context and explanation. Essentially, the graph approach could list relevant courses but struggled to justify or elaborate on why those courses were relevant to the specific skill mentioned, because that information was buried in unstructured course descriptions.

The combined RAG approach effectively addressed the shortcomings of the single-method approaches. It provided more comprehensive and coherent answers by using the knowledge graph to ensure inclusion of all structurally relevant items (courses, requirements, etc.), while simultaneously using the vector search to pull in descriptive details. For the ISOM area depth question above, the combined system was able to enumerate all the required courses and recommended electives (thanks to the graph data) and also supply additional context about each course's content (thanks to the vector data). For the question about practical data analytics skills, the system not only identified specific courses that teach the DCT model or similar techniques, but also included information from the course syllabi explaining how those courses develop practical skills.

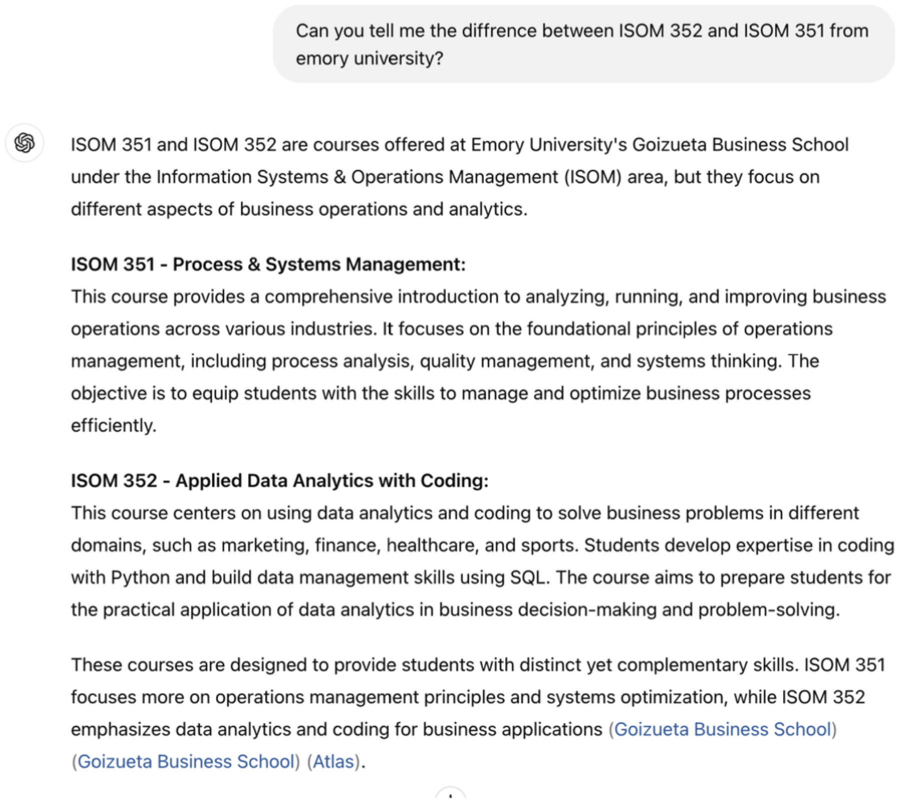
These results suggest that the hybrid approach yields an advising assistant that can handle both checklist-style queries and more open-ended informational queries with equal ease.

5.2 Comparison with Conventional LLM-Based Systems

In addition to internal testing, we compared our RAG-enhanced chatbot's responses with those from a conventional LLM-based system that does not use an external knowledge base (for instance, a standard GPT-4 chatbot with internet browsing or a static knowledge cutoff). Overall, the RAG system demonstrated superior performance in providing accurate and context-rich answers.

A conventional LLM, even when it has access to some online data, often returns answers that are either too general or partially incorrect when faced with detailed academic advising questions. It might provide definitions or generic advice but struggle to assemble a comprehensive answer that touches on all relevant aspects of a specific query. For example, when prompted to compare two similar courses (such as ISOM 351 vs. ISOM 352), a vanilla LLM could retrieve basic descriptions of each course. However, it typically would not clearly articulate the differences and similarities in their focus, prerequisites, and roles within the curriculum.

In our trials, the baseline ChatGPT gave separate summaries of the two courses but did not connect them or compare their content in depth (Figure 4). In contrast, our RAG-powered chatbot delivered an integrated response that directly compared the courses—highlighting, for instance, that ISOM 351 focuses on process management and practical operations techniques, whereas ISOM 352 emphasizes data analytics and programming skills, and noting that both are part of the ISOM track with complementary roles (Figure 5). This level of comparative insight is possible because the chatbot could query the structured knowledge graph for relationships (like common prerequisites or track requirements) while also retrieving specific details from course descriptions.



Can you tell me the diffrence between ISOM 352 and ISOM 351 from emory university?

ISOM 351 and ISOM 352 are courses offered at Emory University's Goizueta Business School under the Information Systems & Operations Management (ISOM) area, but they focus on different aspects of business operations and analytics.

ISOM 351 - Process & Systems Management:
This course provides a comprehensive introduction to analyzing, running, and improving business operations across various industries. It focuses on the foundational principles of operations management, including process analysis, quality management, and systems thinking. The objective is to equip students with the skills to manage and optimize business processes efficiently.

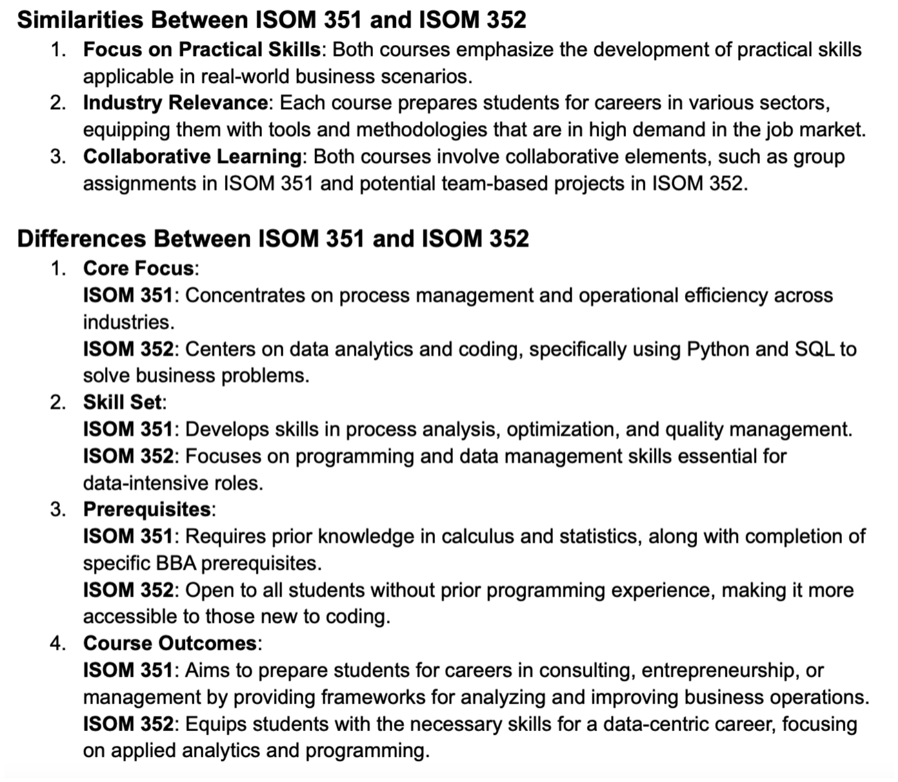
ISOM 352 - Applied Data Analytics with Coding:
This course centers on using data analytics and coding to solve business problems in different domains, such as marketing, finance, healthcare, and sports. Students develop expertise in coding with Python and build data management skills using SQL. The course aims to prepare students for the practical application of data analytics in business decision-making and problem-solving.

These courses are designed to provide students with distinct yet complementary skills. ISOM 351 focuses more on operations management principles and systems optimization, while ISOM 352 emphasizes data analytics and coding for business applications (Goizueta Business School) (Goizueta Business School) (Atlas).

Figure 4: ChatGPT (conventional LLM) response to a sample query about two courses.

We found that the RAG system also excels at interpreting and answering complex, multi-part questions. Its query parsing mechanism (through LangChain) breaks down a complicated question into components and searches the knowledge graph and document repository systematically. As a result, the chatbot can handle questions that would typically require a human advisor's synthesis of multiple information sources.

The answers generated by the RAG-based system were not only accurate but also tailored to the context of the question, often pointing the student to specific programs or courses that addressed their interests or needs. Overall, by grounding the LLM's responses in a


Similarities Between ISOM 351 and ISOM 352

1. **Focus on Practical Skills**: Both courses emphasize the development of practical skills applicable in real-world business scenarios.
2. **Industry Relevance**: Each course prepares students for careers in various sectors, equipping them with tools and methodologies that are in high demand in the job market.
3. **Collaborative Learning**: Both courses involve collaborative elements, such as group assignments in ISOM 351 and potential team-based projects in ISOM 352.

Differences Between ISOM 351 and ISOM 352

1. **Core Focus**:
   ISOM 351: Concentrates on process management and operational efficiency across industries.
   ISOM 352: Centers on data analytics and coding, specifically using Python and SQL to solve business problems.
2. **Skill Set**:
   ISOM 351: Develops skills in process analysis, optimization, and quality management.
   ISOM 352: Focuses on programming and data management skills essential for data-intensive roles.
3. **Prerequisites**:
   ISOM 351: Requires prior knowledge in calculus and statistics, along with completion of specific BBA prerequisites.
   ISOM 352: Open to all students without prior programming experience, making it more accessible to those new to coding.
4. **Course Outcomes**:
   ISOM 351: Aims to prepare students for careers in consulting, entrepreneurship, or management by providing frameworks for analyzing and improving business operations.
   ISOM 352: Equips students with the necessary skills for a data-centric career, focusing on applied analytics and programming.

Figure 5: Chatbot response using the RAG system for the sample course comparison query.

curated knowledge graph and a rich set of documents, the advising agent provides a level of comprehensiveness and correctness that a stand-alone LLM could not easily achieve. This demonstrates the value of the RAG approach in an academic advising setting: it significantly reduces misinformation and improves the relevance of the chatbot's guidance, thereby making it a more trustworthy and useful tool for students.

6 Limitations and Future Work

While our LLM-augmented advising agent shows promise, we identified several limitations that suggest directions for future work:

- **Coverage and currency of data**: The knowledge base (especially the JSON-derived knowledge graph) may have missing or outdated entries, since it was built from a snapshot of available documents. If curriculum changes occur or if certain edge cases were not captured in the initial data extraction, the system could provide incomplete or outdated advice. Improving the *extract-transform-load* pipeline with more robust parsing and periodic updates is necessary to keep the knowledge graph current. Additionally, incorporating better prompt engineering or fine-tuning for data extraction could reduce omissions.

- **Handling of edge cases and depth of knowledge**: Some queries may fall outside the scope of the current knowledge graph or require a level of detail not represented in the data. For instance, highly specific questions about uncommon academic scenarios or personalized advice (e.g., combining multiple programs or dealing with exceptions to policies) may not be fully answerable by the chatbot. Expanding the knowledge graph with more nodes and relationships, or integrating additional data sources, would improve coverage. Automating the expansion of the knowledge graph by mining semantic relationships from unstructured text (as opposed to the largely manual schema we used) could also enhance scalability and reduce the manual effort in updating the graph (Ji et al., 2021).

- **Data availability and integration**: Our project was limited by the data accessible to us. Many institutional details are not centrally stored or are restricted, which made it challenging to compile a comprehensive advising knowledge base. In future iterations, collaboration with university departments to gain access to official databases or APIs would allow for more direct integration of up-to-date information. This would streamline data ingestion and improve the reliability of the agent's responses.

- **Model performance and resource constraints**: We primarily worked with a single LLM (OpenAI's GPT-4) and did not experiment with alternative models due to resource limitations. High-performing proprietary models may not be cost-effective or necessary for all advising tasks. Exploring open-source LLMs or smaller fine-tuned models could be worthwhile to ensure the system remains scalable and affordable, especially if deployed at an institutional scale. The trade-off between model complexity and real-time performance should be assessed.

- **Evaluation metrics**: Our evaluation of the chatbot's performance was largely qualitative, based on comparative assessments of answer completeness and accuracy. We did not employ standard quantitative metrics (such as accuracy against a ground truth dataset or user satisfaction ratings) to rigorously measure performance improvements. Future work should include the development of an evaluation framework with clear metrics—potentially accuracy, precision/recall, or user survey scores—to objectively assess the system's effectiveness and track improvements over time.

For the next phase of the project, we plan to develop a more polished graphical user interface (GUI) and conduct a pilot deployment with academic advisors at GBS. As part of this deployment, we will prepare a user manual and training session for advisors to ensure they can effectively utilize the chatbot. We will gather feedback from this pilot to identify any usability issues or gaps in the chatbot's knowledge. This iterative refinement process will guide enhancements to both the knowledge base and the interaction design of the system.

By addressing the limitations outlined above and continuously incorporating user feedback, we aim to evolve the advising agent into a reliable, accurate, and widely applicable tool that can augment academic advising across various departments and institutions.

7 Conclusion

In summary, this project demonstrates that large language models, when carefully integrated with a structured knowledge graph and vector-based retrieval techniques, can significantly enhance the academic advising process. Our LLM-driven advising agent for Emory's Goizueta Business School was able to deliver more accurate, comprehensive, and context-aware responses to student inquiries than a conventional LLM-based chatbot.

The incorporation of a RAG approach allowed the system to draw consistent and relevant information from multiple sources—providing coherent guidance that mirrors what a well-informed human advisor might offer, rather than the fragmented or occasionally incorrect answers from an unaugmented model. The RAG-enhanced advising tool not only improves the quality of information available to students but also has the potential to reduce the workload on academic advisors by handling routine queries and providing instant support. Advisors can then devote more time to complex or personalized guidance that requires human judgment.

Given the successful outcome within the GBS context, this approach could be extended to other programs at Emory and to other institutions facing similar challenges in academic advising. It highlights the transformative potential of combining LLMs with knowledge graphs and other retrieval strategies in educational support systems.

Moving forward, we envision this system being adopted as part of the university's advising resources, continually updated with new data and refined through user feedback. The collaboration between AI tools and academic advisors can create a more accessible and efficient advising ecosystem, ultimately helping students make better-informed decisions about their academic trajectories and career goals.

References

Erdl, A. (2023, December 7). *Unifying LLMs & Knowledge Graphs for GenAI: Use Cases & Best Practices*. Neo4j Blog. Retrieved from https://neo4j.com/blog/unifying-llm-knowledge-graph/

Gan, W., Qi, Z., Wu, J., & Lin, J. C. W. (2023, December). Large language models in education: Vision and opportunities. In *2023 IEEE International Conference on Big Data (BigData)* (pp. 4776–4785). IEEE.

Ji, S., Pan, S., Cambria, E., Marttinen, P., & Philip, S. Y. (2021). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems, 33*(2), 494–514.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., … Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems* (NeurIPS 2020), 33, 9459–9474.

Peláez-Sánchez, I. C., Velarde-Camaqui, D., & Glasserman-Morales, L. D. (2024). The impact of large language models on higher education: Exploring the connection between AI and Education 4.0. *Frontiers in Education, 9*, 1392091.