IT UNIVERSITY OF COPENHAGEN

RESEARCH PROJECT

Comparing Vanilla and Constraint-Aware RAG for Exchange Student Educational Chatbot System

Sneha Shrestha

snsh@itu.dk

Agnieszka Kujawska

agku@itu.dk

Supervised by:

Arturo Valdivia Martinez

arva@itu.dk

3 February 2026

Contents

Abstract

Universities consistently provide more and more information on their websites by the use of static web pages, making it time-consuming for students to answer complex, constraint-based questions such as course selection, credit requirements, or eligibility conditions. In this study, we evaluate two Retrieval-Augmented Generation (RAG) approaches for answering queries related to becoming an exchange student at the IT University of Copenhagen (*ITU*): a standard vector-based Vanilla RAG (*V-RAG*) and a Constraint-Aware RAG (*C-RAG*) that integrates structured SQL-based retrieval (courses metadata) with unstructured document retrieval (*ITU* content as in *V-RAG*). As an absolute baseline, we propose a keyword-based search (*KBS*) representing manual information retrieval from official *ITU* website and treat found answers as golden. We conduct both qualitative and quantitative evaluations across a diverse set of queries that triggered different routes of retrieval. Our results show that while both RAG systems avoid hallucinations through strict prompting, *C-RAG* consistently produces more precise, complete, and faster responses for all types of queries. In terms of numbers, *C-RAG* achieves higher claim-level accuracy (0.90 vs. 0.78) and lower average latency (11.04s vs. 18.58s) compared to *V-RAG*, while maintaining comparable user satisfaction. The findings highlight the advantages of combining structured data retrieval with semantic search and demonstrate that constraint-aware retrieval substantially improves usability for real-world academic information systems.

1 Introduction

Browsing different websites to find relevant information often requires a lot of time and causes frustration, as the UIs are not always easy to navigate and user-friendly. Interfaces vary widely in usability, and essential details are frequently distributed across multiple pages. It may be difficult to trace back to initial important information once a concept is explored further and further. This case is especially challenging for prospective or current university students who are preparing for an exchange semester, where the application process requires precision in order to fulfill all the requirements and not miss any deadlines. The information required for formalities is usually easy to find within a few steps. However, becoming an exchange student involves not only submitting the application, but also meeting courses requirements and aligning with students' expectations.

Each semester, incoming exchange students at the *ITU* face a number of challenges when trying to navigate the course selection and application process. Course descriptions must be manually browsed by clicking them one by one, opening in a separate tab and closing afterwards. Students often struggle to determine which courses match their expectations, such as interests, semester, or language of instruction. Additionally, courses can't be chosen arbitrarily or solely based on interest; they must add up to the required ECTS points. The most common way of searching information is a keyword search or using static FAQ pages which provide limited support. They lack semantic understanding and cannot be used for the type of constraint reasoning that students ask for, such as filtering by semester, language, or any just relevance. An even more challenging task arises when course selection depends on less common conditions, such as the absence of mandatory activities or the format of the exam. Some students may want to avoid submitting weekly assignments due to other responsibilities, or they may have difficulty with oral exams. Exchange students frequently rely on manual comparison, trial-and-error browsing, or external guidance to find information and make decisions. Even though the information needed is available on the website, *ITU* employees may still spend their time replying to emails that often include conceptually similar questions. As a result, students spend unnecessary time gathering all needed information piece by piece, focus on search rather than making confident and well-informed decisions. A more intelligent and constraint-aware system is needed to enhance this process and help students save time and nerves, as well as to unburden the *ITU* administration.

To address these issues, this project investigates whether a Retrieval-Augmented Generation (RAG) system can improve the quality of requested information alongside with time efficiency due to the lack of usage of manual tools. Specifically, we explore a hybrid architecture that integrates both relational databases and vector databases to combine precise constraint filtering with semantic retrieval of unstructured content. We compare it against the baseline keyword-based search (*KBS*) and Vanilla RAG (*V-RAG*) that stores all the information onle in the vector database. The project is guided by the following research question: *To what extent does a chatbot, using a constraint-aware RAG architecture that integrates relational and vector databases,*

improve exchange students' ability to find accurate and relevant information about studying at ITU compared to Vanilla RAG and keyword-based search?

2 Background

2.1 Large Language Models and Retrieval-Augmented Generation

Large Language Models (LLMs) [1] are transformer-based neural architectures trained on large-scale corpora to perform tasks such as question answering, summarization, and dialogue generation through next-token prediction. Because of their scale, these models show emergent abilities including in-context learning and generalization across tasks without explicit fine-tuning [1]. However, their parametric knowledge is fixed at the time of training. As a result, LLMs often generate outdated or incorrect factual information, especially for domains that change frequently or require precise and authoritative data. This limitation are known as hallucinations, where seemingly plausible but wrong claims are produced [2]. In the context of university support systems, this can lead to errors such as incorrect course availability, misreported prerequisites, or outdated campus-life information.

Retrieval-Augmented Generation (RAG) [3] frameworks address these shortcomings by integrating an external retrieval component into the LLM pipeline. In a RAG system, the user query first triggers a retrieval step that selects semantically relevant documents from one or more external knowledge sources. These documents are then provided to the LLM as additional context. This enables the model to keep up with up-to-date content without modifying its underlying parameters. RAG therefore improves factual consistency, reduces hallucinations, and allows dynamic updates to the knowledge base [3]. Such characteristics are especially advantageous in domains with evolving content, such as university course information and campus life descriptions.

2.2 Structured and Unstructured Data with Storage and Querying Modalities

Information systems usually separate data into structured and unstructured types, based on how organized the data is and how easily it can be searched or queried. Structured data refers to relatively well-organized information according to rigid schemas such as relational tables with defined columns for numbers, categories, dates, and relationships between records [4]. In a university dataset, this category often includes course codes, credit values, enrollment limits, or prerequisite relationships. Traditional relational database management systems (RDBMS) [5] use the SQL declarative language to perform complex filtering, joining, and aggregation with high accuracy and low latency [4]. Structured data can be inserted into traditinal RDBMS and supports deterministic and exact querying, where correctness depends on precise matches, range queries, or relational joins. Structured data can be queried using many different approaches [6].

State-of-the-art NL2SQL systems typically require supervised training on large annotated datasets and involve non-trivial model architectures, which increases implementation complexity compared to lightweight rules [7]. For structured information such as course catalogues, parameters for SQL queries (course codes, ECTS ranges, semesters) can often be extracted using lightweight rule-based templates, for example regex patterns and keyword slots for common query forms. These approaches are attractive because they are interpretable, deterministic, and relatively easy to maintain [6].

In contrast, unstructured data such as web pages, PDFs, student guides, or free-text descriptions lacks a predefined schema, making it unsuitable for exact relational querying. This type of content is typically accessed using semantic search via vector embeddings. In this setting, text segments are encoded into high-dimensional vectors, stored in vector databases, and retrieved based on similarity. This approach is the core of vanilla RAG pipelines [3], which often embed all content uniformly. While this is effective for semantic understanding, embedding-based retrieval struggles on tasks requiring exact numeric matches, temporal constraints, or relational logic [4].

The difference between structured and unstructured data leads to important design choices in chatbot architectures. Depending on what a user asks and what kind of operations are needed, the system has to decide whether to query a relational database, a vector database, or a combination of both.

2.3 Limitations in RAG and Hybrid Retrieval Architecture

Although RAG improves factual accuracy, it has its shortcomings. Traditional (vanilla) RAG pipelines treat all knowledge as unstructured text and depend solely on vector embeddings for retrieval. This approach struggles with handling queries that requires handling of high-precision constraints [8], such as filtering by course credits, identifying time-conflicting classes, or retrieving the exact prerequisites for a program. Embedding-based retrieval may return semantically related documents but still sometimes fail to meet strict logical or numerical requirements. Moreover, vector search also brings challenges like embedding drift, limited context windows, and retrieval noise. Long documents have to be split into chunks, which risks losing relational or hierarchical information. When chunks lack sufficient context, retrieval quality degrades, and the LLM may compensate with hallucinations [8].

These limitations motivate the use of hybrid retrieval architectures. In such systems, structured data is stored in SQL databases, while unstructured documents are kept in vector databases. Queries are then routed to the most suitable backend: SQL for constraint-based questions, semantic search for descriptive content, or a combination of both when a query involves multiple types of data.

3 Previous Research

Handling university responsibilities poses recurring challenge for students, especially those who must navigate unfamiliar university websites when preparing for an exchange semester. While universities typically provide course descriptions, prerequisites or ECTS points and administrative guidelines online, the information can be found across multiple pages which requires time for exploration. There are different approaches towards gaining the desired information and this section reviews the literature on traditional search systems, semantic retrieval or RAG architectures, and hybrid retrieval approaches combining both structured and unstructured data.

Most university websites rely on keyword-based information retrieval systems built on classic Information Retrieval (*IR*) techniques such as inverted indexes, Term Frequency–Inverse Document Frequency (*TF-IDF*) scoring, or simple filtering [9]. Although it is computationally efficient, these systems struggle when user queries are more ambiguous, phrased differently than in the underlying documents, or require reasoning across multiple criteria. Prior studies have shown that students often spend significant time manually browsing course catalogs or backtracking through multiple tabs. Even then, they may not always find the information they need, and often have to contact administrative staff to clarify or get the final response[10]. Key pain points are highlighted in here: fragmented information, poor navigability, and limited support for multi-constraint decision-making. This motivates the need for more intelligent retrieval system capable of interpreting natural language queries and integrating multiple data sources.

In the last decade, embedding-based semantic retrieval has emerged as a promising solution to the limitations of *KBS*. Vector databases such as *FAISS* or *Chroma* enable similarity search across embedding spaces, allowing systems to retrieve documents based on meaning rather than lexical overlap. Such retrieval mechanisms form the foundation of Retrieval-Augmented Generation (RAG), where retrieved information is fed into a Large Language Model (*LLM*) to produce final answers in a natural language [11].

Researches within that domain demonstrate RAG's advantages for tasks that require a lot of knowledge in order to provide accurate answers [12]. Additionally, there exist different variations of RAG such as Blended RAG which significantly outperform vanilla RAG on multiple *IR* benchmarks[13]. Traditional retrieval systems and search engines often return a long list of passages ranked by lexical similarity. However, these systems sometimes fail to correctly rank the most relevant passages at the top. RAG with re-ranking provides a solution that improves retrieval accuracy substantially by applying the re-ranking method as a second step in the retrieval process, which helps to assign more accurate relevance scores and leverage deep contextual understanding. Moreover, Agentic RAG was introduced as a more advanced type of *RAG* system that combines retrieval, generation, and autonomous decision-making—essentially turning the system into an "agent" that can plan, reason, and act in multiple steps [14].

Other work explores specialized embedding models for education. In 2025, three authors introduce an open-source dual-loss embedding model tailored to semantic retrieval in higher-

education contexts, improving alignment between student queries and course syllabi [15]. It demonstrates that these domain-specific embeddings outperform general-purpose models on retrieval tasks in higher-ed contexts.

RAG has also been widely adopted in educational chatbots, with Swacha and Gracel surveying applications across 47 studies [16]. These works highlight RAG's potential to support learning, streamline guidance systems, and reduce administrative load.

4 Data

4.1 Data sources

For our project, we use two complementary types of data: structured course information and unstructured university documents. The structured dataset includes course descriptions collected from LearnIT (Moodle) pages, covering 144 courses across undergraduate and graduate programs at *ITU*. Each course entry contains detailed metadata, such as the course title, course code, ECTS credits, semester availability, academic level (BSc or MSc), language of instruction, prerequisites, learning objectives, teaching methods, examination format, grading, instructor information, scheduling details, and course availability for exchange or guest students. This metadata helps for constraint-aware retrieval, allowing the system to respond to queries with specific filters such as ECTS theresholds, semester restrictions, or language requirements.

The unstructured data consists of documents collected from public *ITU* webpages. These documents cover topics such as admissions, application requirements, student housing, campus facilities, student services, and other aspects of university life. While the structured course data mainly supports fact-based queries with specific constraints, the unstructured text helps answer open-ended questions about studying at *ITU* and the overall exchange experience for students. Together, these two sources create a diverse knowledge base that can address the information needs of exchange students.

4.2 Data collection and preprocessing

The structured course data used in the project is obtained through a custom multi-stage HTML parsing pipeline. First, all available exchange course pages are manually downloaded and archived as raw HTML documents. The raw HTML files had a substantial heterogeneity in page structure and captured over thirty distinct metadata fields which were all important for our project. Therefore, we started with a section-based extraction using BeautifulSoup [17] that identifies logical content blocks by detecting heading elements and aggregating subsequent sibling nodes until the next heading at an equivalent or higher hierarchical level. Then, we used key–value extraction on structured HTML constructs such as definition lists, table rows, and instances where labels appear in `<strong>` tags followed by descriptive content. Finally, we added pattern-based extraction methods that uses regular expressions to identify fields that has

predictable syntactic structure. The extracted data is then exported into an intermediate CSV file and is subsequently imported into SQLite. This constructs a relational schema with over thirty-five columns covering course metadata, academic requirements, and availability information. To support efficient querying, B-tree indices are automatically created on frequently accessed fields such as course code, course title, semester, level, and programme name. This helped enabling multi-criteria searches with logarithmic-time complexity (O(log n)) across filters such as ECTS, language, semester, academic level etc.

Unstructured content from *ITU* is collected using a custom web crawler which performs breadth-first traversal starting from a seed URL (*https://en.itu.dk/Programmes/Exchange-students*). URL discovery and retrieval are handled concurrently with ten worker threads. The crawler uses domain-specific filters to focus on pages relevant to academic programs (e.g., URLs containing keywords such as "exchange" or "programme") while excluding irrelevant sites like social media (Facebook, Instagram, LinkedIn, YouTube), non-HTML files, and noisy sections (`/News`, `/Event`). Then, the filtered HTML content is parsed using BeautifulSoup. Boilerplate elements like headers, footers, navigation bars, and social media sections are removed. The remaining content is organized hierarchically extracting page titles, section headings with their level metadata, paragraphs, and list items. The text is cleaned by collapsing whitespace, removing persistent interface artifacts, and applying basic length filtering. Cleaned texts are combined into a single field with newline separators and enriched with metadata including the source URL, heading structure, raw paragraph arrays, and overall word count. All processed documents are saved as UTF-8 encoded JSON. For semantic retrieval, the JSON corpus is split into overlapping chunks using a sliding-window approach. Text is tokenized by whitespace, with a window size of 500 words and a 100-word overlap to preserve context across chunks. Chunks with fewer than 20 words are discarded. Each valid chunk is embedded using the all-MiniLM-L6-v2 SentenceTransformer model, producing 384-dimensional dense vectors. These embeddings are L2-normalized and stored into a FAISS IndexFlatIP structure, allowing efficient cosine-similarity-based retrieval.

This preprocessing pipeline finally results in a hybrid knowledge base for our chatbot system that is both structured and searchable, combining relational storage for course metadata with vector embeddings for broader *ITU* information.

5 System Architecture

5.1 System overview and components

Our research examines three information retrieval approaches for supporting queries about becoming an exchange student at *ITU*. The first one is a manual Keyword-Based Search (*KBS*) serving as an absolute baseline. This is not explicitly implemented as part of our system., but rather used as a benchmark to evaluate potential improvements and to assess whether more advanced systems provide any benefits over classic approach. Secondly, we implement Vanilla

Retrieval-Augmented Generation (*V-RAG*) pipeline, and an extended version of the this baseline incorporating additional components, which we call Constraint RAG (*C-RAG*). The latter two systems are deployed as conversational interfaces, commonly referred to as chatbots.

Keyword-based search

This approach serves as the absolute baseline against which we evaluate whether the two proposed systems achieve improved performance and provide more efficient responses. The method relies on manually answering user queries by navigating the *ITU* website. For each query, we asked participants to record the time required to get the information.

Vanilla RAG (Baseline System)

The *V-RAG* system represents a single-source retrieval approach, where all data is stored in a vector database without the support of another, structured, domain-specific table. Both course metadata and other unstructured *ITU* information are embedded and stored in a single vector table. The system contains a query classification step, where each input is classified to one of the two labels: *VECTOR* or *REASONING*. Although the architecture includes a query classification step, both labels rely solely on querying vector database and the reasoning part is a layer on top the retrieval step. Consequently, the system lacks differentiated query routing, and every query follows similar processing pipeline regardless of its intent. This system adheres to the standard *V-RAG* paradigm [18]: relevant documents are retrieved using vector similarity search, and responses are generated solely based on the retrieved context.

V-RAG performs a top-k similarity search over the vector database using embeddings. Given that the total number of courses offered at *ITU* (144), we set the default retrieval size to *k=144*. This allows the system to answer queries such as: "Give me a list of all courses offered at ITU." Found vectors are merged and formatted in order to create an input to large language model (*LLM*). It then generates the final response with the use of extracted knowledge and user's query. The system has a strict system prompt to enforce consistent formatting and accuracy:

```
system_prompt = (
    "You are a concise, strict-format assistant for exchange"
    "students at the IT University of Copenhagen (ITU)."
    "Important rules (follow exactly):"
    "1) Always use ONLY the provided Context. Do NOT invent facts"
    " or add information not present in Context."
    " If something is not in Context, reply:"
    " 'I couldn't find that in the available ITU data.'"
    "2) FORMATTING FOR GENERAL INFORMATION:"
    " - Use only vector data from Context"
    " - Write concisely in one or two short paragraphs"
```

```
    " - Do NOT include citations or source references"
    "3) Be concise and well-formatted: clear section headers,"
    " proper spacing between sections"
    "4) Do not include any intermediate JSON or streaming fragments -"
    " output a single clean text response only.\n"
    "5) If the query is unrelated or Context is empty, respond with:"
    " 'I couldn't find relevant information to answer your question.'"
)
```

If the *LLM* isn't enable or fails for some reason, the system returns a deterministic response based on provided template using the retrieved content. This ensures that users always receive a response, providing graceful degradation and prevents the user from being exposed to system's failures. This design provides a transparent and reproducible baseline against which more advanced RAG variants (e.g., multi-route, hybrid, or tool-augmented systems) can be compared.

Constraint RAG

C-RAG extends the *V-RAG* architecture into a hybrid RAG system, integrating structured SQL-based retrieval alongside unstructured vector-based retrieval. SQL queries support constraint-based filtering (e.g., language, semester, minimum ECTS) and combination reasoning, while the vector database continues to store unstructured *ITU* content as in the baseline system. A multi-stage query classifier determines the appropriate retrieval path for each query: *SQL* for structured queries (course codes, ECTS constraints), *VECTOR* for general information queries, *HYBRID* for queries benefiting from both sources, and *REASONING* for advanced course selection, e.g., finding courses that sum up to a specified ECTS points. For general course-related queries, the classifier performs active SQL probing by generating 1-4 word n-grams with noise filtering that excludes common stopwords (e.g., "show," "give," "available"), and testing them against course titles. Queries focused on administrative or exchange topics are directed to *VECTOR* retrieval for unstructured content. Queries combining both course and administrative signals use the *HYBRID* approach, combining *SQL* and *VECTOR* results with greater weight on structured data. Any unmatched queries default to *VECTOR* search. Finally, retrieved contexts from all the retrieval types are formatted and truncated to maintain token efficiency before being passed to the LLM for response generation.

Reasoning layer

While both systems include optional reasoning layer (e.g., ECTS combination reasoning), these are implemented strictly on top of vector-retrieved (in *V-RAG*) or SQL-retrieved, vector-retrieved (in *C-RAG*) and do not introduce additional data sources. It allows both systems for combinatorial course-selection queries with specified condition. This extension enables the systems to answer questions that require reasoning over multiple items, such as identifying combinations of courses that sum to a specified number of ECTS credits. A query detection mechanism

identifies reasoning-oriented queries by analyzing the user input. Such queries are detected first through keywords like “total,” “add up,” or “combine” paired with ECTS values. When such a query is detected, it is classified as a *REASONING* query and routed to the reasoning layer. In the next step, depending on the system, the context data is either retrieved from SQL or vector database. From these chunks, course attributes such as course code, ECTS value, semester, and language are heuristically extracted from text using pattern matching. The extracted courses are then used to employ a backtracking algorithm with pruning, memoization, and time constraints to compute valid course combinations that exactly satisfy the requested ECTS total. The resulting combinations are formatted into a structured, human-readable context, which is then passed further.

LLM Integration and Response Generation

For response generation, both systems issue HTTP streaming requests to an Ollama server using its cloud-hosted gpt-oss model. Model behavior is controlled through environment-level configuration, including temperature and maximum token limits. Streaming responses are supported, with safeguards implemented to ensure stable decoding and interruption handling. The prompt architecture relies on a strict system message that enforces several rules: responses must use only the provided context and refuse out-of-scope queries; course lists must follow a fixed numbered format; hybrid responses must separate response from different sources with a blank line; general information must be written as concise paragraphs derived solely from vector data; and citations or source references are prohibited. The streaming protocol parses JSON line by line, extracting only the permitted ‘response’ or ‘output’ fields and suppressing intermediate “thinking” content until a final done:true flag appears. Temperature (default 0.1), maximum tokens (default 800), and a 30-second timeout are configurable. A fallback mechanism maintains system reliability. If Ollama is unavailable, returns an error, exceeds time limits, or produces an invalid response, the system switches to deterministic template-based generation. SQL-only queries yield numbered course lists populated with metadata. Vector-only queries return the highest-similarity snippet truncated to the context limit. Hybrid queries combine numbered course lists with general-information paragraphs. This approach ensures consistent, usable responses even when LLM generation fails, preserving overall availability as a core design requirement.

5.2 User Interaction

The user interface (*UI*) and user experience (*UX*) of our *ITU* Chatbot were designed to prioritize simplicity and responsiveness, creating a natural and engaging conversational experience for students. The design adheres to common instant messaging paradigms, minimizing the cognitive load associated with navigating complex interfaces, which can become overwhelming [19]. This approach leverages user’s existing mental models of interactions - known in *UX* design as schemas. By implementing the three-part structure of an application (header, message

history, and input field), the interface instantly signals its purpose and guides the user toward next possible actions without the need to overthink or get lost in the setup. In contrast to traditional university websites that often involve scanning menus and navigating multiple pages, the chatbot offers access to information without extensive training that can be foun in one place.

The chatbot incorporates three primary visual elements that immediately suggest their purpose: a large text box with a *Type your message here...* - placeholder clearly indicating where to input text, a paper airplane icon serving as the send button and a distinct avatars with message bubbles that clearly differentiate between the Bot and the User.

Given that most users are already familiar with messaging platforms such as Messenger or WhatsApp, the interface requires no learning time. Students can start interacting immediately and efficiently access relevant information - supporting the overall goal of helping exchange students to quickly gather important information. The visual layout is implemented using an HTML container-based structure and CSS styling with clarity and consistency, and alignment with ITU's simple color scheme.

HTML Structure

The interface is logically divided into: a header with the name of the chatbot and university branding, a scrollable message history and an input area.

CSS Styling The design employs a clean and ITU-based aesthetic with black, grey, white and orange as primary colors. Customized message bubbles differentiate user and bot messages through distinct shapes and colors, enhancing readability. In order to maintain a smooth conversational experience - especially given the variable latency of LLM-based responses - key dynamic features were implemented:

- **Typing Indicator**: A *Bot is typing...* message appears immediately after a message is sent, providing users with real-time feedback while the system processes the query
- **Asynchronous Message Handling**: User messages are sent to the backend without blocking the *UI*. This prevents freezing and retains smooth interaction even when the process takes long
- **Real-Time Rendering**: Messages are appended to the DOM as soon as they are generated, and always the most recent response is being shown to the user
- **Relative Timestamps**: Messages such as *Just now* or *5m ago* are produced to give user the sense of live conversation

All the above components together enhance system responsiveness and maintain the continuity, which provides the user with much better experience when interacting with the app. Additional interface elements that help users manage message constraints and support intuitive interaction are:

- **Character Counter**: A dynamic counter guides users to avoid reaching 500-character

limit

- **Flexible Input Submission**: Messages can be submitted either by clicking the send button or by pressing *Enter*, accommodating different user habits and aligning with common messaging conventions.

6 Experimental Setup

6.1 Systems Evaluated

This section describes the experimental design used to evaluate the *ITU* Chatbot system, comparing two distinct RAG approaches: *V-RAG* and *C-RAG*. We asked 18 students, to answer 30 questions we defined and track the time it took to get the final answer.

6.2 Test Query Design

To evaluate and compare both chatbot systems, we constructed a benchmark of 30 test questions. The questions were intentionally designed to cover the full range of information types that exchange students typically seek. To ensure balanced evaluation across retrieval approaches, the test set included:

- 10 SQL-type questions (structured facts such as ECTS, prerequisites, course schedules, semesters)
- 10 Vector-type questions (unstructured information such as housing, events, application process)
- 10 Hybrid questions (queries requiring both structured and unstructured data)

SQL-type questions can be answered entirely from structured database. This will be relevant in order to compare both chatbots and check whether incorporating SQL table (in *C-RAG*) improves the accuracy of the answers. The Vector-type questions relate to narrative or descriptive information contained in unstructured text, whereas hybrid questions require retrieving both structured and unstructured data from different sources. This distribution ensures that each retrieval type based on *C-RAG* is triggered and thus allows for evaluating integrated information and reliable comparison. Questions designed solely for descriptive answers would never query the SQL database, and a comparison between vanilla and *C-RAG* in that case would provide no meaningful insights. Representative examples from each category are shown below.

- "What is the level (BSc or MSc) of the course Data Visualisation and Data-driven Decision Making?" (SQL)
- "Can you suggest combinations of courses totaling 22.5 ECTS that are offered in Danish language?" (SQL)

- "Tell me about campus life and student housing." (Vector)
- "What documents do I need for admission application?" (Vector)
- "Give me advanced programming courses and application deadlines for exchange students." (Hybrid)
- "List all courses available for Autumn 2026. What research opportunities does *ITU* provide for exchange stduents?"

By designing questions across these three categories, we aimed to stress-test the systems' retrieval paths (SQL, vector, hybrid) and verify whether constraint-aware routing improves accuracy and efficiency.

6.3 Evaluation Metrics

We evaluated system performance using four primary metrics that fall into two subcategories: user-related (response latency, user satisfaction) and chatbot-related (accuracy and hallucination rate).

User-related metrics

Response latency was measured in seconds and captured the time each participant took to get an answer to the query. For complex, open-ended queries that involved navigation through multiple pages or documents, latency measurements were right-censored at 5 minutes; that is, any response taking 5 minutes to get was recorded as >5 minutes rather than measuring the full duration. Because some open-ended questions could require long searching time to find the final answer, participants were instructed to stop once the 5-minute threshold was reached and to report *>5 minutes* instead of continuing. This approach ensured consistent data collection while preventing excessive task time for participants. Consequently, for the keyword-search condition, latency measurements were collected from 18 participants for all 30 questions. On the other hand, the chatbots generate responses independently of user searching behavior, so their response times were treated as fixed values for each question after the answer was generated for the first time. The second metric, user satisfaction, was assessed on 1-5 scale for each question and each system. We assumed a default satisfaction score of 5 for the keyword-search method, as users are free to explore results without time constraints. Eventually, no matter the time, everybody should get the desired answer. For our research, we assign to all these answers scores of 5, meaning highest satisfaction. For the chatbot systems, each participant provided a score for every question, and we subsequently computed the mean satisfaction score across all users.

Ultimately, this procedure yielded three sets of aggregated performance metrics—latency and satisfaction scores for the keyword-based search tool and both chatbots, enabling a comparative evaluation across all three approaches.

Chatbot-related metrics

In order to provide quantitative metrics of the chatbots' performance, we decided to calculate accuracy and its complement - hallucination rate. Once each question was submitted to both systems, the resulting answers were compared against golden truth answer that we found on *ITU* website. Accuracy was measured at the claim level, following standard RAG-evaluation practices. Each answer was decomposed into "atomic" factual claims:

$$\text{Accuracy} = \frac{\text{Number of Correct Claims}}{\text{Total Claims}}$$

This method allows uniform scoring across SQL, vector, and hybrid questions, even when binary scoring (correct/incorrect) is insufficient. Hallucination rate measures unsupported or incorrect factual statements:

$$\text{Hallucination Rate} = \frac{\text{Incorrect + Unsupported Claims}}{\text{Total Claims}}$$

This is complementary to accuracy but highlights wrong or fabricated statements rather than correctness.

6.4 Annotation

To ensure a fair and reproducible evaluation, all system outputs were annotated using a structured process. A detailed annotation guide with step-by-step instructions was provided to all annotators during the process. The full guide is included in Appendix A for reference. A total of 18 student annotators participated in the evaluation. All annotators were asked to answer all 30 test queries using keyword-based searches on the *ITU* website itself to establish ground-truth reference answers. During the process, they also recorded the time required to get each answer manually using keyword searches. Next, they were given answers to all test questions from the two evaluated chatbots. Answers were anonymized so annotators did not know whether the response came from the *C-RAG* or the *V-RAG* baseline. The annotation focused only on the user-related metric: user-satisfaction. They were highly encouraged to rate each answer based on how well the answer help the student, how actionable the information is and how easy the answers were to understand.

7 Results and Analysis

7.1 Qualitative results

In this section, we qualitatively compare three approaches to retrieving and generating answers: *V-RAG*, *C-RAG*, and *KBS*. The golden answers retrieved within *KBS* serve as the reference, providing fully accurate and complete responses. Table 1 below provides summarized results for three systems across sample of three types of questions.

Question	**Completeness / Correctness**
Course Level: Data Visualisation and Data-driven Decision Making	Both systems correctly identify BSc. *V-RAG* provides objectively accurate, but unnecessary information alongside.
15 ECTS and ML-related	Both RAG systems correctly identify the two ways to reach 15 ECTS.
Price (EU/EEA) for course	*KBS* results in the correct price (10625 DKK). Both RAG systems fail to find the price. *C-RAG* is marginally faster in admitting failure (3s vs 12s).
Courses that combine IT with business or management topics	*C-RAG* provided more courses that are actually relevant to the condition in the query.
Courses with Written Exams	*C-RAG* returns much longer list (40 items) of relevant courses than *V-RAG* (6 items), suggesting *C-RAG* is more complete for this aggregation query.
All Exchange Courses	*C-RAG* provides complete list of all the courses in 37s. *V-RAG* is slower (40s) and returns incomplete list, noting the limitation of listing all results.
What can I learn in Computer Systems Performance course	Both systems returned general learning outcomes with *V-RAG* including more details.
How do I apply for admission?	*V-RAG's* response contains numbered list with stage-by-stage approach that clearly guides the applicant through the process. *C-RAG* is more general, but still passes important information.
Documents needed for admission	Both answers include all relevant information without redundancy.
Courses with exam format as submission	*V-RAG* couldn't provide an answer for that question. On the other hand, *C-RAG* has a short, but complete list with relevant courses.
Courses that combine IT with business or management topics	*C-RAG* provided more courses that are actually relevant to the condition in the query. Additionally, we have noticeable time difference: 7s (*C-RAG*) and 17s *V-RAG*.

Table 1: Comparison of *C-RAG* and *V-RAG* across different queries

As shown above, both systems demonstrate high accuracy in most instances; however, *V-RAG* is prone to verbosity and occasional incompleteness. Although it is generally capable of answering complex queries, its outputs often merge information from multiple sources, which can introduce redundancy. In contrast, *C-RAG* most of the time answers the question more precisely and directly, consistently outperforming both *V-RAG* and the *KBS* in terms of response time. The example below illustrates this behavior.

Question: Is the course "Probabilistic Programming" available for exchange students?

System	Answer
Vanilla RAG	1. Probabilistic Programming (Spring 2026) – KSPRPRO1KU – 7.5 ECTS – Spring 2026 The course "Probabilistic Programming" is offered in Spring 2026 and is explicitly listed as available for exchange students. This MSc-level course introduces Bayesian statistics, Bayesian data analysis...
Constraint RAG	1. Probabilistic Programming (KSPRPRO1KU) – 7.5 ECTS – Spring 2026 2. Probabilistic Programming, BDS (BSPRPRO1KU) – 7.5 ECTS – Spring 2026 The course is available for exchange students.

Table 2: Comparison of Vanilla RAG vs Constraint RAG answers

When both chatbots are asked open-ended questions, such as "Which courses in Spring 2026 have written exams?", both systems return correct results. However, *C-RAG* usually produces a longer list that still remains accurate. For simpler factual queries, identifying the correct answer does not appear to be challenging for either chatbot, but *C-RAG* consistently responds significantly faster. Table 3 presents the response latency of both systems for a selection of representative queries.

Question	V-RAG	C-RAG
Level of the course	12s	4s
Course available for exchange students	13s	7s
Price of a course	12s	3s

Table 3: Comparison of *V-RAG* vs *C-RAG* response times

In general *C-RAG*, shows marked improvements in precision and time over *V-RAG* by explicitly enforcing query-specific constraints during retrieval and generation. This approach produces answers that closely align with what can be found on the *ITU* website. For instance, when asked to suggest 15 ECTS of elective courses related to natural language processing, *C-RAG* accurately generated only the relevant combinations. Similarly, filtering by Danish-language courses or summing ECTS points yields concise and actionable course selections, reducing noise and improving efficiency compare to *KBS*, which is unfeasible for such questions. *C-RAG* is also generally faster than *V-RAG* for complex, constrained queries because irrelevant retrievals are

filtered out early in the process, resulting in more focused outputs. While it may omit information outside the scope of the constraints, this trade-off enhances usability and ensures that the generated answers remain highly relevant and interpretable. However, the general trend is that both systems are significantly faster than the manual golden answers, but this speed comes with a trade-off in accuracy or completeness for certain complex queries.

Quantitative results

Complementary to the qualitative results, this section presents quantitative comparison of the three evaluated systems. The evaluation focuses on claim-level accuracy, hallucination rate, user satisfaction, and response latency, as summarized in Table 4. It is important to note that because all *KBS* answers were manually verified against official *ITU* webpages, we treated them as golden answers and assumed perfect accuracy (1.0), zero hallucination rate (0.0), and maximum user satisfaction (5) for all test queries. This choice is made to reflect the theoretical correctness of manually retrieved information when sufficient time is available.

Table 4: Overall System Performance

System	Claim-Level Accuracy	Hallucination Rate	User Satisfaction (1–5)	Latency (s)
Keyword-search	1	0	5	160
Vanilla RAG	0.78	0	4.1	18.58
Constraint-Aware RAG	0.9	0	4.3	11.04

Both RAG systems achieve a hallucination rate of 0, indicating that neither system produces unsupported or fabricated claims. Instead when they cannot find the information, they respond with statements such as *"I couldn't find that in the available ITU data"*. This is due to strict prompting strategy used in both systems, which forces the models to abstain when information is unavailable. However, the two systems differ substantially in overall claim-level accuracy. *V-RAG* achieved an accuracy of 0.78, while *C-RAG* improved accuracy to 0.9. The accuracy gap is primarily due to incomplete answers produced by *V-RAG* for structured queries. Because *V-RAG* relies exclusively on vector-based semantic search, it sometimes fails to retrieve specific course attributes such as exam format in a course. In such cases, the system correctly avoids hallucination but returns partial answers. In contrast, *C-RAG* uses SQL-based retrieval for structured attributes using exact course metadata and resulting in more complete responses. For example, when asked *"What is the level of the course Large-Scale Data Analysis, and what exam format does it use?"*, *V-RAG* correctly identified the course level as Bachelor but stated that no exam format was available in the *ITU* data. In reality, the course uses a written exam on premises, which is explicitly documented in the structured course metadata. In contrast, *C-RAG* retrieved both the course level and the correct exam format by querying the structured SQL database. This example shows how *C-RAG* produces more complete answers for constraint-based queries, which contributes to its higher overall accuracy score.

This improvement in accuracy is also reflected in user satisfaction scores, where *C-RAG* with score 4.3 slightly outperforms *V-RAG* with 4.1. Although the numerical difference is modest, it is consistent across participants and query types. Annotators frequently rated *C-RAG* higher due to its more concise, focused, and actionable answers, particularly for course-selection queries involving multiple constraints. For example, in SQL and hybrid questions, *C-RAG* typically returned short, well-filtered course lists that directly addressed the query, whereas *V-RAG* often included additional descriptive context that required further manual filtering. For purely descriptive vector-type questions (e.g., campus life or housing), satisfaction scores between the two systems were similar, reflecting their shared reliance on the same unstructured document corpus.

In terms of response latency, *KBS* required substantially more time than both RAQ systems due to its fully manual nature. Participants had to navigate multiple webpages, open course descriptions individually, and cross-check information across different sections of the *ITU* website, which resulted in an average response time of 160 seconds. Some queries were likely to take hours to find Amanually and cross-check information but we capped the time to 5 minutes. Compared to *KBS*, both RAG-based systems significantly improved response time by automating information retrieval and answer generation. This allows students to obtain answers within seconds rather than minutes or hours. Among the two RAQ systems, *C-RAG* achieved the lowest latency with an average response time of 11.04 seconds, compared to 18.58 seconds for *V-RAG*. This improvement is mainly due to SQL-based filtering, which retrieves only a small and relevant subset of information for some queries. In contrast, *V-RAG* often retrieves a large number of vector-based documents that increases the amount of context passed to the language model and, consequently, the response time.

8 Limitations and Future Work

One of the limitation encountered during the evaluation process concerns more open-ended questions and their corresponding golden answers. Manually finding ground-truth answers for such questions would have required a lot of time and effort. Consequently, rather than doing it, we evaluated the chatbot responses by verifying their factual correctness through manual fact-checking through *ITU* website. Additionally, the questions with assigned latency of >5 minutes lack the time granularity that is available for simpler questions. As a result, these questions weren't taken into account when calculating average scores. Regarding the satisfaction score, we assumed that all answers returned by keyword-based web searches correspond to a score of 5 on a 1–5 scale. While this assumption simplifies the evaluation process, it introduces uncertainty, as the true quality of such answers cannot be guaranteed in all cases. Moreover, if the information seeked by the user is not available online (on the university website), it is impossible to expect any chatbot to outperform the *KBS*. In such cases, neither users nor automated systems can reliably resolve the query based purely on publicly accessible data. For certain questions, particularly those involving non-public information, users should contact the

university directly in order to get the answer. This approach leads to overestimated satisfactory score for *KBS* and makes it almost unfeasible to beat that.

Another limitation stems from the reliance on a substantial number of hardcoded rules and heuristics to detect specific conditions in user queries, such as references to ECTS credits, semester, language, and similar attributes. While these rules allow for better retrieval, they reduce system flexibility and robustness. Hardcoded patterns may fail to work well on badly phrased queries, leading to missed detections or incorrect interpretations. Moreover, maintaining and extending such rules requires manual effort every time new condition is introduced, which limits scalability. As a result, system performance may be sensitive to these predefined assumptions, and improvements observed in this setup may partially reflect engineering choices rather than inherent advantages of the Constraint RAG approach itself.

The last issue found in *V-RAG* is related to time of the response, which seems to be rather long compared to *C-RAG* and moreover the *KBS*. In the configuration of the vector top-k parameter, the number was set to 144. As mentioned before, this corresponds to the total number of courses available at *ITU*. This design choice was motivated by the need to support queries that potentially reference all courses simultaneously. However, retrieving such a large number of documents significantly increases the amount of context passed to the *LLM*, leading to longer processing times and higher latency. In some cases, the basic *KBS* provides a better experience in finding answers rather than the chatbot. Our *C-RAG* solves that problem by providing another SQL table alongside. However, the potential solution to improve the performance of our baseline chatbot could be the introduction of dynamic k. Instead of using a fixed value of 144, the system could dynamically adjust it based on the query type, as some of them may not need that much context. Broad, aggregate queries (e.g., "all courses") could trigger higher k, while specific queries would retrieve only a small, relevant subset.

Apart from improving the system by selecting more appropriate LLMs or embedding models, future work could address the aforementioned limitation by replacing the hard-coded rules with a learned natural language–to–SQL (NL2SQL) component. Such a model could translate user queries directly into structured SQL statements, enabling more flexible and robust interpretation of constraints. This would reduce the amount of code, improve generalization to unseen query formulations, and allow the system to scale more easily to additional domains. Additionally, the effectiveness of our systems are dependent on the availability and quality of structured metadata. The number of pages that were scraped is limited, and thus the answers may not always be fully complete. Future iterations could incorporate additional *ITU* data sources e.g., integrating scheduling information (e.g., weekly time slots) would enable detection of timetable conflicts and support more advanced course-planning scenarios.

Lastly, the system relies on strict prompting to control hallucinations. More benefits could be achieved through domain-specific fine-tuning of both embedding models and the generation model. Fine-tuning on *ITU* content and question–answer pairs could improve semantic retrieval quality and response coherence.

9 Conclusion

This paper investigated whether a constraint-aware Retrieval-Augmented Generation architecture can improve the quality and efficiency of information access for exchange students compared to both a traditional keyword-based search and a standard vector-based RAG system. Motivated by the challenges students face when navigating fragmented university websites and handling queries with multiple constraints, we designed and evaluated two chatbot systems: Vanilla RAG (*V-RAQ*) and Constraint-Aware RAG (*C-RAQ*) using real ITU exchange courses and administrative data.

Our results show that while both RAG systems substantially outperform manual keyword-based search in terms of response time and usability, the integration of hybrid retrieval in *C-RAQ* provides clear advantages. Quantitative evaluation shows that *C-RAQ* achieves higher claim-level accuracy and lower average latency compared to *V-RAQ*, while maintaining similar high user satisfaction. Moreover, both systems successfully avoid hallucinations due to strict prompting strategies, indicating that the observed differences in results are driven by completeness of information retrieval rather than unsafe or irrelevant generation.

Qualitative analysis further highlights that *V-RAQ*, although being generally accurate, sometimes produces incomplete answers for structured queries due to its sole reliance on vector-based semantic retrieval. In contrast, *C-RAQ* consistently retrieves precise course information by querying structured metadata directly. This leads to more concise, focused, and actionable responses for multi-constraint queries that are common among exchange students.

In summary, the findings confirm that combining structured and unstructured data within a hybrid RAG architecture significantly improves system performance for academic information systems. By explicitly aligning retrieval mechanisms with the nature of the query, *C-RAQ* reduces unnecessary context, improves response time, and delivers more complete answers without increasing a risk of hallucination. These results suggest that hybrid *C-RAQ* is a promising direction for applying reliable and scalable chatbot systems in educational institutions where accuracy, clarity, and efficiency are critical.

10 References

[1] H. Naveed et al., 'A comprehensive overview of large language models,' *ACM Trans. Intell. Syst. Technol.*, vol. 16, no. 5, Aug. 2025, ISSN: 2157-6904. DOI: 10.1145/3744746. [Online]. Available: https://doi.org/10.1145/3744746.

[2] F. Antonius et al., 'Incorporating natural language processing into virtual assistants: An intelligent assessment strategy for enhancing language comprehension,' *International Journal of Advanced Computer Science and Applications*, vol. 14, 10 2023. DOI: 10.14569/ijacsa.2023.0141079.

[3] Y. Gao et al., 'Retrieval-augmented generation for large language models: A survey,' *arXiv preprint arXiv:2312.10997*, vol. 2, no. 1, 2023.

[4] S. Mishra and A. Misra, 'Structured and unstructured big data analytics,' in *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, 2017, pp. 740–746. DOI: 10.1109/CTCEEC.2017.8454999.

[5] L. Qin, J. X. Yu and L. Chang, 'Keyword search in databases: The power of rdbms,' in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '09, Providence, Rhode Island, USA: Association for Computing Machinery, 2009, pp. 681–694, ISBN: 9781605585512. DOI: 10.1145/1559845.1559917. [Online]. Available: https://doi.org/10.1145/1559845.1559917.

[6] X. Liu et al., 'A survey of text-to-sql in the era of llms: Where are we, and where are we going?' *IEEE Transactions on Knowledge and Data Engineering*, vol. 37, no. 10, pp. 5735–5754, 2025. DOI: 10.1109/TKDE.2025.3592032.

[7] X. Liu et al., 'A survey of nl2sql with large language models: Where are we, and where are we going?' *arXiv preprint arXiv:2408.05109*, 2024, NL2SQL Handbook: https://github.com/HKUSTDial/NL2SQL_Handbook. [Online]. Available: https://arxiv.org/abs/2408.05109.

[8] J. Genesis, 'Retrieval-augmented text generation: Methods, challenges, and applications,' 2025.

[9] J. Zobel, *Inverted files for text search engines*, Lecture notes / tutorial (PDF), Available online, 2003. Accessed: 3 Dec. 2025. [Online]. Available: https://dmice.ohsu.edu/bedricks/courses/cs506-problem-solving-with-large-clusters/articles/week1/zobel_invertedindex.pdf.

[10] A. Muhammad et al., 'Evaluating usability of academic websites through a fuzzy analytical hierarchical process,' *Sustainability*, vol. 13, no. 4, p. 2040, 2021, Open access. DOI: 10.3390/su13042040. [Online]. Available: https://www.mdpi.com/2071-1050/13/4/2040.

[11] M. Klesel and H. F. Wittmann, 'Retrieval-augmented generation (rag),' *Business & Information Systems Engineering*, vol. 67, no. 4, pp. 551–561, 2025, Open access. DOI: 10.1007/s12599-025-00945-3. [Online]. Available: https://link.springer.com/article/10.1007/s12599-025-00945-3.

[12] W. Yu, 'Retrieval-augmented generation across heterogeneous knowledge,' in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies – Student Research Workshop (NAACL-SRW 2022)*, Association for Computational Linguistics, 2022, pp. 52–58. DOI: 10.18653/v1/2022.naacl-srw.7. [Online]. Available: https://aclanthology.org/2022.naacl-srw.7.pdf.

[13] K. Sawarkar, A. Mangal and S. R. Solanki, *Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers*, arXiv preprint, 2024. eprint: 2404.07220. [Online]. Available: https://arxiv.org/abs/2404.07220.

[14] S. Yao et al., 'React: Synergizing reasoning and acting in language models,' in *International Conference on Learning Representations (ICLR) Workshops*, arXiv preprint, arXiv:2210.03629, 2023. [Online]. Available: https://arxiv.org/abs/2210.03629.

[15] R. Sajja, Y. Sermet and I. Demir, 'An open-source dual-loss embedding model for semantic retrieval in higher education,' *arXiv preprint*, vol. abs/2505.04916, 2025. arXiv: 2505.04916 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2505.04916.

[16] J. Swacha and M. Gracel, 'Retrieval-augmented generation (rag) chatbots for education: A survey of applications,' *Applied Sciences*, vol. 15, no. 8, p. 4234, 2025, Open access. DOI: 10.3390/app15084234. [Online]. Available: https://www.mdpi.com/2076-3417/15/8/4234.

[17] L. Richardson, *Beautiful soup*, https://www.crummy.com/software/BeautifulSoup/, Version 4.x, 2004.

[18] P. Lewis et al., 'Retrieval-augmented generation for knowledge-intensive nlp tasks,' *Advances in Neural Information Processing Systems*, vol. 33, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401.

[19] *Cognitive Science*, DOI: 10.1207/s15516709cog1202_4. [Online]. Available: https://doi.org/10.1207/s15516709cog1202_4.

11 Appendix

Appendix A: Annotation Task

What is the goal of the annotation?

The goal is to evaluate how well two anonymized chatbots designed for ITU exchange students answer 30 test queries about courses offered in ITU, administrative procedures, and campus life. Annotators rate **user satisfaction** by comparing chatbot responses to ground-truth answers found via manual keyword search on the ITU website (`en.itu.dk`), focusing on helpfulness, actionability, and clarity for real students.

What should the annotators do?

For each of the 30 test queries:

- Step 1: Answer the query yourself using keyword search on the ITU website. Record your ground-truth answer and time taken (max 5 minutes). Any query that takes more than 5 minutes, you can stop at 5 minutes and record time as ">5 minutes".
- Step 2: Read the two anonymized chatbot responses (Chatbot A and Chatbot B).
- Step 3: Rate each on **User Satisfaction** (1-5 scale).

User Satisfaction Scale (1-5):

5 - Complete, accurate, actionable (specific courses/actions), saves all manual time.
4 - Mostly complete, specific guidance, minor gaps.
3 - Partially helpful, some useful info but vague/incomplete.
2 - Limited relevance, requires almost full manual search.
1 - Wrong/irrelevant/confusing—no time saved.

Query types (10 each): SQL (course facts like ECTS/codes), Vector (campus info like housing), Hybrid (mixed). Full query list provided separately.

What are the common mistakes/things to watch out for?

- Rate based on your ground truth - don't assume facts or reward "nice phrasing."
- Hallucinations (wrong facts) = 1-2, even if fluent.
- Don't penalize style/length and focus on student utility (actionable? clear?).
- Be consistent across all 30 queries; compare strictly to manual search results.
- If a chatbot answer matches your ground truth perfectly $\rightarrow$ 5; if unusable $\rightarrow$ 1.

Appendix B: GitHub

Link to our *GitHub repository* used for this project.