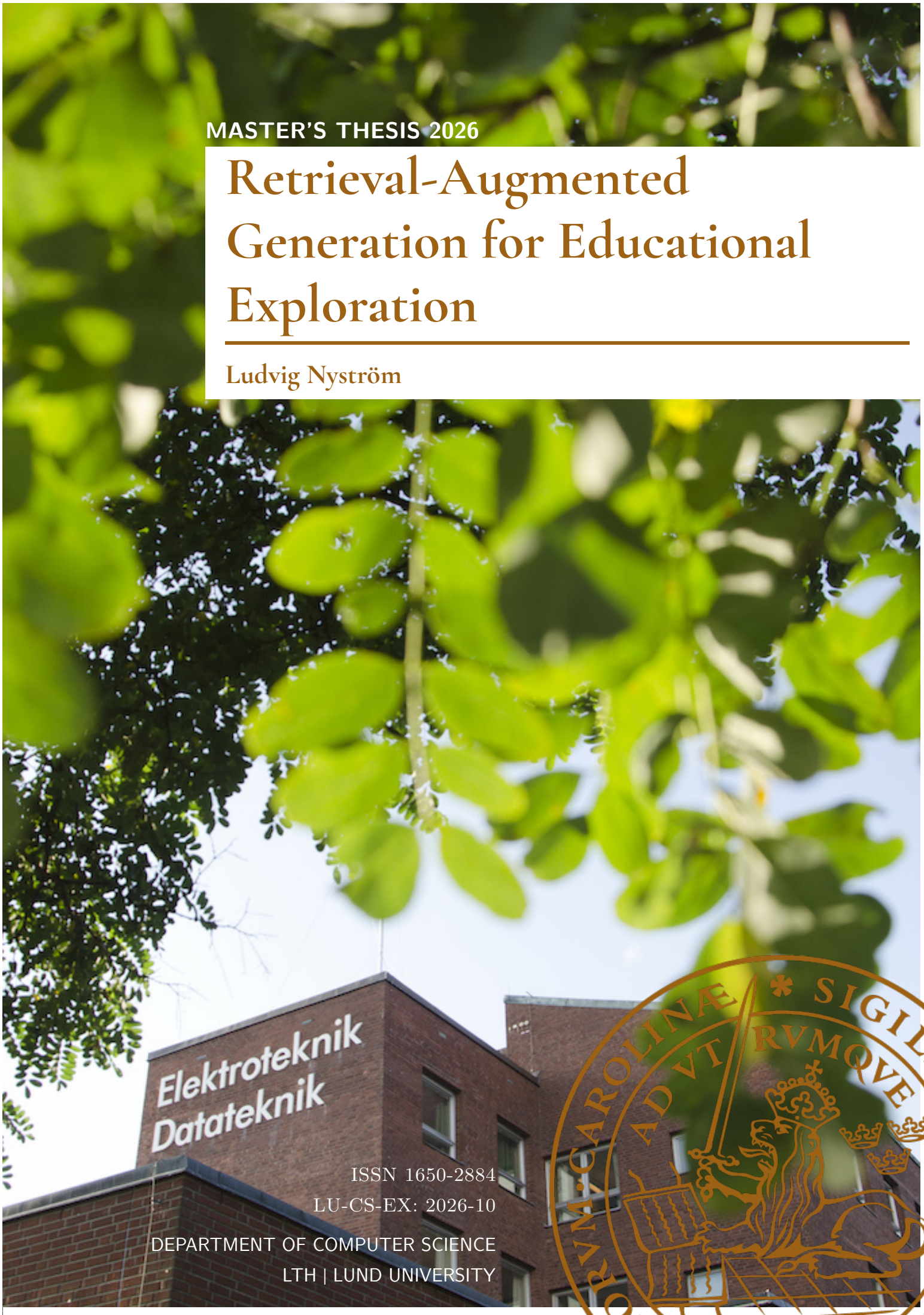


MASTER'S THESIS 2026

Retrieval-Augmented Generation for Educational Exploration

Ludvig Nyström

ISSN 1650-2884
LU-CS-EX: 2026-10

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2026-10

Retrieval-Augmented Generation for Educational Exploration

Retrieval-Augmented Generation för utbildningsvägledning

Ludvig Nyström

Retrieval-Augmented Generation for Educational Exploration

(Master Thesis – Lund University, Faculty of Engineering (LTH))

Ludvig Nyström
lu1100ny-s@student.lu.se

April 2026

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Peng Kuang, peng.kuang@cs.lth.se

Examiner: Pierre Nugues, pierre.nugues@cs.lth.se

Abstract

This thesis designs and evaluates a retrieval-augmented generation (RAG) architecture for a chatbot in Skoolie, a Swedish school platform that helps students explore higher-education programs. It addresses a key limitation of the baseline system: answers are generated without access to structured program data. The redesigned system retrieves relevant program records from a higher-education catalogue and injects them into the prompt using schema-aware context packaging and cross-encoder reranking. Experiments compare retrieval strategies and quantify trade-offs under product-like latency constraints. Evaluation combines automated metrics with a human A/B survey in which respondents rated answers without knowing which system produced them. The results show that retrieval-conditioned generation improves grounding and perceived answer quality while remaining compatible with the system's latency budget.

Keywords: Retrieval-augmented generation, hybrid retrieval, cross-encoder reranking, context packaging, educational guidance

Acknowledgements

I want to thank Peng Kuang for supervising this thesis. Our weekly meetings kept the work on track, and their feedback on both the technical details and the writing made the final result considerably better than it would have been otherwise.

At Skoolie, I owe particular thanks to Rasmus Häggkvist, who was always available to talk through design decisions and different approaches, and who put considerable effort into the evaluation work. Thanks also to the rest of the Skoolie team for their help along the way.

I am grateful to the students and career counselors who took part in the evaluation study. Their time and honest responses made the human evaluation possible.

Finally, I want to thank Pierre Nugues for taking on the role of examiner and for his input during the project.

Contents

Chapter 1
Introduction

In Sweden, the unemployment rate among 15–24-year-olds reached 24.3 percent in 2025, placing the country third-highest in the European Union [14]. More than half of those counted as unemployed are full-time students who are also looking for work, so the headline number overstates the problem. But even adjusting for that, the figure reflects a labour market where young people struggle to find their way, and where the path from upper secondary school to stable employment is far from obvious.

Part of the difficulty is navigational. The Swedish higher-education landscape includes over 3,500 undergraduate and vocational programs spread across nearly 300 providers. Students choosing between these options typically have access to a career counselor (*studie- och yrkesvägledare*) who is responsible for several hundred students at once and may meet each one individually only once during upper secondary school. The rest of the time, students are on their own, sorting through university websites, program catalogues, and admissions portals of varying quality. Students whose parents attended university or who happen to know someone in a relevant profession can navigate this landscape more easily. Others may never discover options that would have suited them well.

Skoolie AB builds a platform intended to reduce this asymmetry. The platform is deployed in schools and lets students explore educational programs and career paths based on their grades, interests, and preferences in a single environment. As students interact with the system, their career counselor sees a structured overview of saved choices, interests, and merit values, which makes the limited face-to-face time more productive.

A central component of the platform is a chatbot that uses large language models to answer questions about Swedish educational programs and suggest relevant alternatives. For this to work in a school setting, the answers need to be more than fluent. They need to be grounded in actual program data, and the connection between what the chatbot says and what it recommends needs to be visible and verifiable. If a student asks about nursing programs in Gothenburg, the explanation should describe the programs that actually appear on screen, not a plausible-sounding summary that the model assembled from its training data.

Retrieval-augmented generation (RAG) is a natural architecture for this problem. In-

stead of generating an answer from the model's parameters alone, a RAG system retrieves relevant records from an external knowledge base and injects them into the prompt before generation. The model then writes its response conditioned on that retrieved context. In Skoolie's case, the knowledge base is a catalogue of Swedish higher-education programs with structured metadata: titles, providers, locations, eligibility requirements, admission statistics, and program descriptions.

This thesis designs, implements, and evaluates a RAG pipeline for Skoolie's chatbot. The work focuses on three questions: how structured program records should be serialised into the LLM's context window, how retrieval strategy and reranking affect both relevance and latency under realistic deployment constraints, and whether the resulting answers are better aligned with the programs shown to the user.

Contributions. The thesis makes three contributions. First, it introduces explanation–recommendation alignment (ERA) as a metric for checking whether the chatbot's explanation actually matches the program cards shown on screen. Second, it builds a RAG pipeline that serialises structured Swedish program records into the LLM's context using schema-aware packaging, and measures the effect on grounding and ERA against a baseline system that generates before it retrieves. Third, it compares dense and hybrid retrieval with and without cross-encoder reranking, reporting both relevance gains and the latency cost under production constraints.

Thesis layout. Chapter 2 describes Skoolie's platform, the limitations of the baseline system, and the motivation for a RAG-based redesign. It concludes with the research questions and objectives. Chapter 3 surveys prior work on RAG, dense and hybrid retrieval, reranking, context packaging for structured records, and evaluation of groundedness and faithfulness. Chapter 4 describes the educational data corpus and the baseline system architecture. Chapter 5 presents the redesigned RAG pipeline: query routing, filter construction, hybrid retrieval, cross-encoder reranking, schema-aware context packaging, and grounded generation. Chapter 6 analyses end-to-end and stage-level latency for both architectures. Chapter 7 describes the evaluation methodology, including the ERA metric and the human A/B study. Chapter 8 reports the results of the alignment evaluation and the human evaluation. Chapter 9 discusses threats to validity. Chapter 10 addresses ethical considerations. Chapter 11 summarises the contributions and outlines future work.

Chapter 2
Background

2.1 The Skoolie Platform

Skoolie's platform helps students explore educational programs and career paths, and gives career counselors a structured view of what their students are interested in. The chatbot is meant to be the main interface for this: students type questions in natural language, and the system responds with explanations and program suggestions.

For this to be useful in practice, the chatbot's answers need to be tied to real program data. A response that sounds helpful but describes programs that do not exist, or that contradicts the recommendation cards shown alongside it, is worse than no response at all. The baseline system had exactly this problem, which is what motivated the redesign described in this thesis.

2.2 Current System Overview

Before the redesign, the chatbot worked in two disconnected steps. The LLM generated a free-text response to the student's query without access to any program data. Then, separately, the system ran a vector similarity search over the education corpus using Swedish Sentence-BERT embeddings and displayed the top results as recommendation cards next to the chatbot's answer.

Because generation happened first and retrieval happened second, there was nothing connecting them. The chatbot could describe program characteristics, eligibility rules, or institutional details that had no relation to the programs the student was actually seeing on screen. This decoupling between generation and retrieval is the structural limitation that the rest of this chapter addresses.

2.3 Limitations of the Existing Approach

The generate-then-retrieve pattern caused three recurring problems in practice, each illustrated with a screenshot from the deployed system.

The most visible problem was misalignment between the generated text and the displayed recommendations. In Figure 2.1, the chatbot describes four specific programs (Textilhögskolan i Borås, Konstfack i Stockholm, Beckmans Designhögskola i Stockholm, and HDK-Valand i Göteborg), none of which appear in the recommendation cards shown alongside the answer. Because the model had no access to the retrieved programs, it drew on its training data and produced references that looked plausible but did not match what the student could actually see on screen.

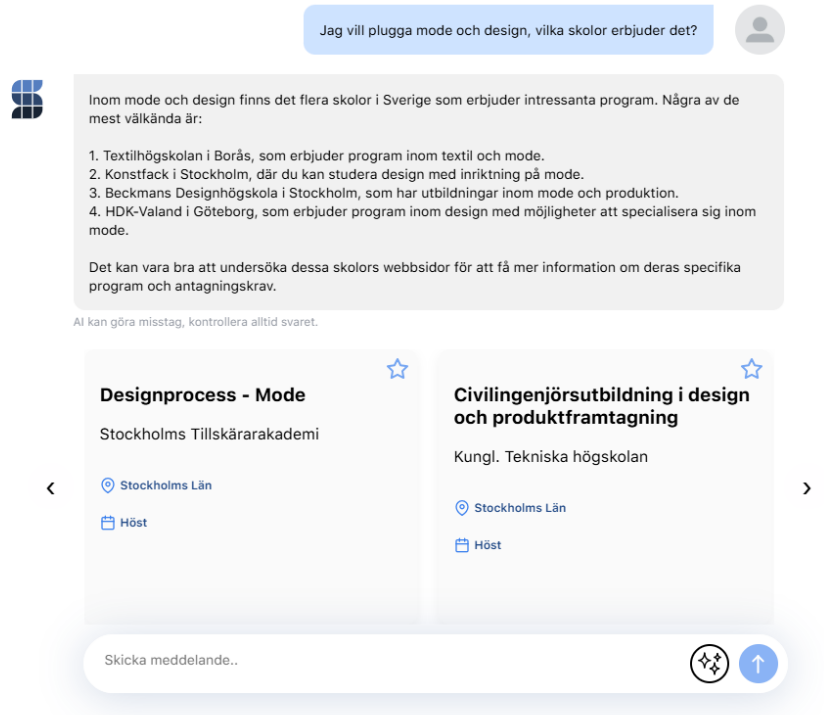


Figure 2.1: The chatbot names four programs that do not appear in the displayed recommendation cards.

A related problem was incorrect or unverifiable claims about program details. In Figure 2.2, the chatbot states that the science bachelor's program in Lund requires Fysik 1 and Kemi 1, while the actual eligibility listed in the program data is Grundläggande behörighet and Matematik 4. Since the model was never exposed to the retrieved records, it could not cite the real requirements and instead produced plausible-sounding but wrong ones.

Finally, the system offered low transparency. In Figure 2.3, a student asks why a particular program was recommended and the chatbot responds that it cannot answer the question. Because the explanation and the recommendations were produced independently, the model had no basis for justifying its own suggestions.

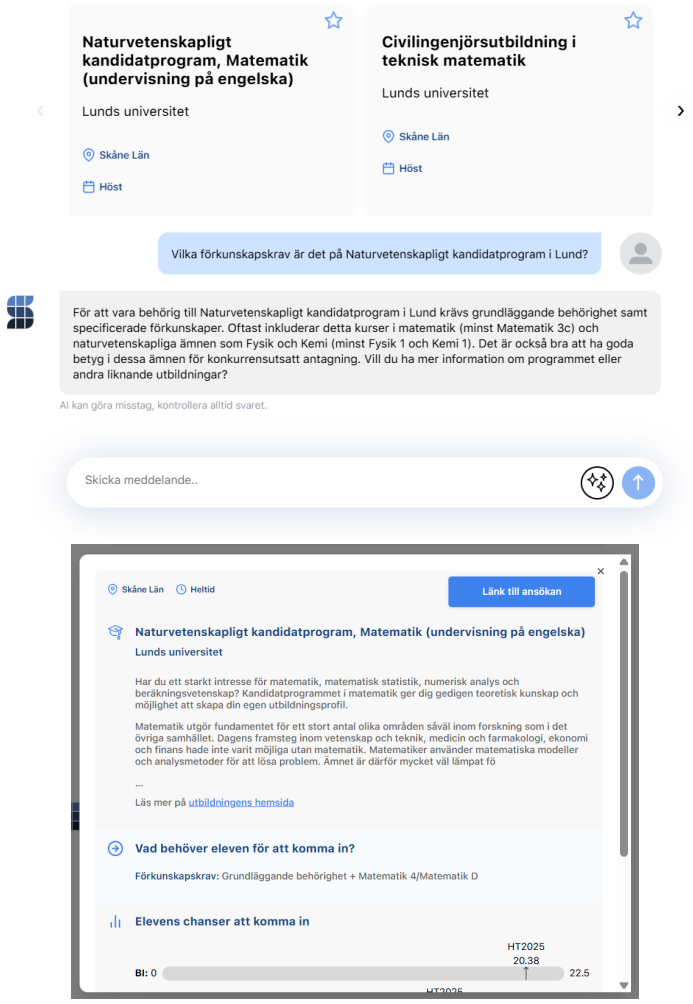


Figure 2.2: The chatbot states incorrect eligibility requirements (top) compared to the actual program data (bottom).

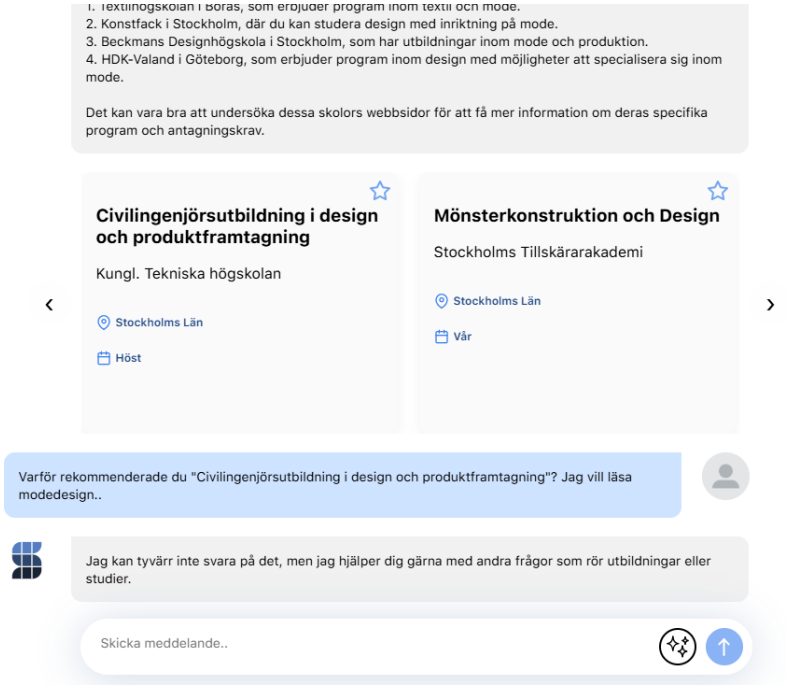


Figure 2.3: A student asks why a program was recommended and the chatbot is unable to explain.

Retrieval-augmented generation (RAG) addresses these problems by reversing the order of operations: the system retrieves relevant records first, injects them into the prompt, and then generates a response conditioned on that context [10]. This means the model sees the actual program data before it writes anything, and the same candidate set is used for both the explanation and the recommendation cards, making the two structurally aligned rather than independently produced.

2.4 User Feedback and Evolving Interaction Model

Before the redesign began, feedback from a customer who had deployed the system surfaced several practical problems.

2.4.1 Observed User Issues

The most common complaint was that the chatbot could not handle specific questions. For example:

> *"Vad behöver jag i merit för att plugga dataingenjör i Lund?"*
> ("What grades do I need to study computer engineering in Lund?")

This type of query would often receive a generic fallback response:

> *"Jag kan tyvärr inte hjälpa dig med det"*
> ("Unfortunately I cannot help you with that")

However, the answer is directly available in the program data (e.g., admission thresholds and eligibility requirements), but the model had no access to it.

Students also encountered hallucinated responses where the model produced fluent but incorrect information. In a few cases the model described application procedures or eligibility rules that did not correspond to any real program, which is a serious problem in a context where students may act on what they read.

The interaction flow was also limiting. In earlier versions, students had to complete a full conversation before receiving any program suggestions, which made it impossible to iteratively explore and refine options within a single session. Some students were also unsure what the chatbot could actually help with or how to phrase their questions.

2.4.2 Changes to the Interaction Flow

In response, the interaction model was changed so that students can continue chatting after receiving recommendations. This was an improvement, but it did not solve the underlying problem: the retrieved programs were still not injected back into the LLM's context. The chatbot could show program cards, but it could not discuss, compare, or answer questions about them in subsequent turns.

The RAG redesign addresses this directly. By retrieving and injecting program data into the prompt on each turn, the model can reference specific recommended programs as the conversation develops.

2.5 Objectives of the Redesign

The redesign started from a concrete product goal: the chatbot and the program suggestions needed to come from the same process, not two loosely coupled components that happened to appear on the same screen. In practice, this meant four things had to change.

First, the chatbot's answers had to be grounded in the actual program data rather than in whatever the model happened to know from training.

Second, the text the chatbot produces and the program cards it displays had to come from the same retrieved set, so that a student or counselor could verify one against the other.

Third, the system needed to support follow-up questions. A student who receives five program suggestions should be able to ask "what are the admission requirements for the second one?" without starting over. This requires the model to have the retrieved programs in its context during generation.

Fourth, all of this had to work within realistic latency constraints. The system is deployed in schools, students are impatient, and adding retrieval and reranking stages to the pipeline cannot push response times to the point where users give up.

2.6 Product Requirements for the RAG-Based System

The requirements below were formulated based on the user feedback, the observed system limitations, and the platform's deployment constraints. They are not aspirational design principles; they are the specific problems the redesign needed to solve.

1. The chatbot must support follow-up questions about previously recommended programs within the same conversation.

2. The programs shown to the student must be relevant to what they asked for. Because only a handful of results are displayed, even one clearly irrelevant suggestion undermines trust in the rest.

3. The chatbot's free-text responses must be grounded in the retrieved program data. The model should not make claims about programs that are not in its context.

4. The programs described in the text and the programs shown as recommendation cards must come from the same retrieved set, so that a student or counselor can verify the explanation against the recommendations.

2.7 Research Problem

The core problem is straightforward: the baseline system generates its answer before it knows what it will recommend, so the two are not guaranteed to be consistent. The redesign reverses this by retrieving first and generating second, but doing so introduces new design decisions (retrieval strategy, reranking, context packaging) that affect both quality and latency. The research problem is to evaluate whether this architectural change actually improves grounding and alignment, and at what cost.

2.8 Research Questions

1. **RQ1:** How does a retrieval-augmented generation architecture with schema-aware context packaging affect explanation–recommendation alignment (ERA) and factual grounding compared to a baseline generate-then-retrieve system?
2. **RQ2:** How do retrieval strategy (dense vs. hybrid) and cross-encoder reranking affect the relevance of retrieved candidates and the end-to-end latency of the pipeline under product-like deployment constraints?
3. **RQ3:** Do users perceive the RAG-based system as more helpful and accurate than the baseline system in a paired evaluation where respondents are unaware of which system produced each answer?

2.9 Objectives

The objective is to design, implement, and evaluate a RAG pipeline for Skoolie's chatbot, with context packaging as the primary design lever. Concretely, this means:

1. Designing and implementing a RAG pipeline with schema-aware context packaging over structured Swedish higher-education program records.
2. Defining the ERA metric and using it to measure whether the RAG pipeline improves the match between the chatbot's explanation and the displayed program cards, relative to the baseline.
3. Comparing dense-only and hybrid retrieval, and evaluating the added value of cross-encoder reranking, in terms of candidate relevance and latency.
4. Measuring end-to-end response time for both architectures under identical conditions to verify that the added pipeline stages do not exceed the product's latency budget.
5. Conducting a human A/B evaluation of perceived helpfulness and correctness with student and career counselor respondents.

Chapter 3
Related Work

This chapter surveys work across six areas that bear on the design and evaluation of the pipeline: the RAG paradigm itself, retrieval strategies, reranking, context packaging for structured data, adaptive RAG in conversational settings, evaluation of groundedness, and LLMs in educational guidance. The final section identifies two gaps in the literature that this thesis addresses.

3.1 Retrieval-augmented generation

Lewis et al. introduced the canonical RAG formulation: pair a language model with a retrieval step so that generation is conditioned on retrieved passages rather than relying on the model's parameters alone [10]. The motivation is straightforward—if the model needs to produce accurate statements about a specific domain, it should see the relevant documents before generating its response.

Since then, RAG has been characterised as a modular pipeline with stages for indexing, query processing, retrieval, reranking, and constrained generation. Survey work by Gao et al. and others emphasises that real-world RAG performance depends at least as much on engineering choices at each stage (hybrid retrieval, reranking strategy, routing logic) as on which LLM sits at the end [6, 18]. This modularity matters in product settings because grounding has to be achieved within a latency budget that users will actually tolerate.

3.2 Dense, Sparse, and Hybrid Retrieval

The retrieval component of a RAG pipeline determines what the model gets to see. Early QA systems used sparse lexical retrieval, primarily BM25 and TF-IDF, both of which score documents based on how often query terms appear in the document relative to how common they are across the corpus. Dense retrieval emerged with dual-encoder approaches like Dense

Passage Retrieval (DPR), where query and passage are embedded into the same vector space and retrieval becomes an nearest-neighbour search [7]. Dense retrieval handles paraphrased and exploratory queries well, because two texts can be semantically close even when they share few words.

The weakness shows up on entity-heavy queries. A student searching for "Sjuksköterskeprogrammet Göteborgs universitet" needs the system to match on the institution name, not just the concept of nursing. Dense retrieval treats "Göteborgs universitet" as part of the overall semantic signal and may rank nursing programs at Umeå or Malmö higher if their descriptions are otherwise similar. Hybrid retrieval addresses this by combining a dense score with a sparse lexical score. Hybrid approaches are widely adopted in production RAG systems precisely because real user queries mix semantic intent with entity-level constraints [6, 18].

It is worth noting that lexical baselines remain surprisingly strong. The BEIR benchmark showed that BM25 is competitive with or outperforms several dense retrievers on out-of-domain tasks, and that gains from more expensive methods like late interaction come with real computational cost [16]. For a system with a latency budget, this means that adding retrieval complexity has to be justified by measurable relevance gains rather than assumed.

3.3 Reranking and Cross-Encoder Methods

In a RAG pipeline, only a handful of retrieved candidates can fit into the LLM's context window. This makes precision at the top of the ranked list disproportionately important: whether a relevant result places fifth or fifteenth determines whether it reaches the model at all if only eight candidates are injected into the prompt.

Reranking addresses this with a two-stage approach. A cheap first-stage retriever (bi-encoder or BM25) returns a broad candidate set, and a more expensive model reorders the top candidates. Cross-encoders, which score each query-document pair jointly rather than embedding them independently, consistently achieve better top-k precision than bi-encoders alone. Nogueira and Cho demonstrated this with monoBERT, a reranker built on BERT (a pre-trained transformer language model), showing that a cross-encoder substantially improves passage ranking when applied as a second stage [13].

ColBERT represents a middle ground: it retains token-level interaction between query and document while allowing partial precomputation, trading some of the cross-encoder's accuracy for better efficiency [8]. In practice, commercially hosted reranking APIs offer another option entirely. They accept plain text, return scores, and keep cost predictable, which is attractive when local GPU hosting is not feasible. The question for any deployed system is whether the latency added by reranking is justified by the relevance gain, a trade-off that has to be measured empirically rather than assumed.

3.4 Context Packaging for Structured Records

Most RAG literature assumes the retrieval corpus consists of unstructured text passages. But many real systems, including educational catalogues, operate over structured records with discrete fields: title, provider, location, eligibility, study pace, and so on. When the retrieved "documents" are actually database rows, how they are serialised into the prompt becomes a

design decision that affects what the model can see and how reliably it can use the information.

Poor packaging wastes tokens, obscures important fields, and can push the model to hallucinate structure that was never made explicit [6]. The problem is compounded by position bias: Liu et al. showed that models can fail to use evidence placed in the middle of a long context, a phenomenon they call the "lost in the middle" effect [12]. This means that it matters not just which fields are included but where they appear in the serialised representation and how consistently that ordering is maintained across different queries.

Despite this, the question of how to serialise structured records into prompts has received little systematic attention. The original RAG paper concatenates retrieved passages as raw text [10], and most subsequent work either does the same or adopts ad hoc formatting without evaluating the packaging itself. This thesis adopts a schema-aware JSON serialisation with fixed field ordering and truncation (described in Section 5.5), but does not experimentally compare it against alternative packaging formats. The design is motivated by the structured nature of the data and by position-sensitivity findings in the literature [12].

3.5 Adaptive RAG in Conversational Systems

Educational guidance queries are rarely self-contained. A question like "How do I become a psychologist?" touches on prerequisites, eligible program types, and local availability, and resolving it fully may take several conversational turns. Iterative RAG frameworks handle this by alternating between retrieval and generation, so that intermediate model output can refine subsequent retrieval steps and improve evidence coverage for complex or underspecified questions [6].

Two design challenges are especially relevant in this setting. The first is deciding when to retrieve at all. Not every conversational turn needs external evidence, and unnecessary retrieval adds latency for no gain. The second is accumulating user constraints across turns. A student who mentions Gothenburg in turn one and nursing in turn three has specified a fairly precise query, but only if the system tracks both constraints. How much context has been gathered from prior turns directly affects retrieval precision at any given point in the conversation. In guidance-oriented dialogues, where students typically reveal their preferences incrementally rather than in a single query, both challenges are especially pronounced.

3.6 Evaluating Groundedness and Faithfulness

Standard IR metrics like Recall@k, MAP, and nDCG measure whether relevant documents are retrieved and well-ranked, but they say nothing about whether the model's generated answer actually uses the retrieved evidence correctly. This is the faithfulness problem: a system can retrieve the right documents and still produce an answer that ignores or contradicts them.

Reference-free frameworks like RAGAS (Retrieval-Augmented Generation Assessment) address this by decomposing RAG quality into retrieval and generation dimensions, including context relevance and faithfulness, without requiring human-annotated reference an-

swers [2]. This is useful for evaluation at scale, but it only checks whether the answer is consistent with the context that was fed to the model, not whether the answer describes the specific items shown to the user.

For systems where the user sees a specific subset of retrieved items displayed alongside the generated text, corpus-level faithfulness is not enough. A response could be perfectly grounded in retrieved evidence while referencing items that were not surfaced to the user. From the user's perspective, that is a misalignment: the explanation describes programs they cannot see. This gap between corpus-level faithfulness and UI-level alignment has not been addressed in prior evaluation frameworks. It motivates the ERA metric introduced in this thesis, which measures alignment specifically between the generated explanation and the displayed recommendation cards.

3.7 LLMs for Educational Guidance and Conversational Recommendation

RAG is increasingly explored in educational applications as a way to improve factuality and keep domain knowledge current without retraining the underlying model [11]. In parallel, work on conversational recommender systems (CRS) examines how LLMs can handle preference elicitation, recommendation, and explanation within a single dialogue [4].

Empirical evaluations of LLM-based CRS raise a concern that is directly relevant here: standard recommendation metrics (precision, recall, nDCG over item lists) may not capture the conversational quality that users actually experience [17]. A system can retrieve relevant items but explain them poorly, or produce compelling explanations that do not match the items shown. This disconnection suggests that automated metrics alone are insufficient and that human evaluation of perceived helpfulness and correctness is needed alongside them.

On the explanation side specifically, recent reviews of LLM-generated rationales in recommender systems emphasise that explanations need to be accurate, specific, and aligned with the recommended items to build user trust [15]. This is the same problem that arises in generate-then-retrieve pipelines: if the model writes its explanation before seeing the recommendations, alignment is not guaranteed and in practice frequently fails.

3.8 Summary and Positioning

Two gaps remain in the literature surveyed above. First, the serialisation of structured records into prompts has not been treated as an experimental variable. Published pipelines either concatenate raw text or use ad hoc formatting, and the packaging step itself is rarely evaluated. Second, faithfulness evaluation has not been extended to UI-level alignment, where the question is not whether the model is faithful to its retrieved context in general, but whether it describes the specific items the user can see on screen.

This thesis addresses both gaps in a Swedish educational guidance setting, where the constraint of operating on a monolingual corpus with domain-specific terminology makes both packaging fidelity and UI-level alignment especially consequential.

Chapter 4
Dataset and System

4.1 Educational data

The retrieval index and the evaluation both operate on the same program corpus. This section describes how that corpus is constructed and filtered, then analyses its composition along dimensions that matter for retrieval: education form, delivery mode, geography, and provider concentration.

4.1.1 Dataset and preprocessing

The education corpus is stored in two tables, `education` and `education_event`, joined on `education_id`.

The dataset is restricted to programs by filtering on `education_type = "program"` and excluding advanced-level (`avancerad`) programs, reflecting the platform's current focus on undergraduate and vocational education. Stand-alone courses and other education types are also excluded from both retrieval and evaluation. After filtering and joining, the corpus contains 3,511 higher-education programs.

The main fields used in the system are:

- **Text fields:** Swedish title (`education_title_swe`), Swedish markdown description (`education_description_swe_markdown`), and Swedish eligibility text (`education_eligibility_swe`).
- **Structural attributes:** education form (`education_form`) and higher-education level code (`uh_education_level`).
- **Delivery and pacing:** remote study flag (`is_distance`) and pace of study percentages (`pace_of_study_percentages`).

- **Location and provider:** municipality code (`municipality_code`) and provider name (`providers`).
- **Retrieval features:** a 768-dimensional embedding vector (`embedding`) computed from the concatenation of the values of four fields: Swedish title, provider name, description, and eligibility text.

This program-level dataset is used both to build the retrieval index and to sample items for evaluation.

4.1.2 Exploratory data analysis

The goal of the exploratory analysis is to understand the composition of the program corpus along four dimensions that are directly relevant for retrieval and evaluation: education form, delivery mode, geographic distribution, and provider distribution.

Education form

Figure 4.1 shows the distribution of education forms across the 3,511 programs. Högskoleutbildning (university education) accounts for 63% of the corpus (2,211 programs), while yrkeshögskola (vocational higher education) accounts for the remaining 37% (1,300 programs).

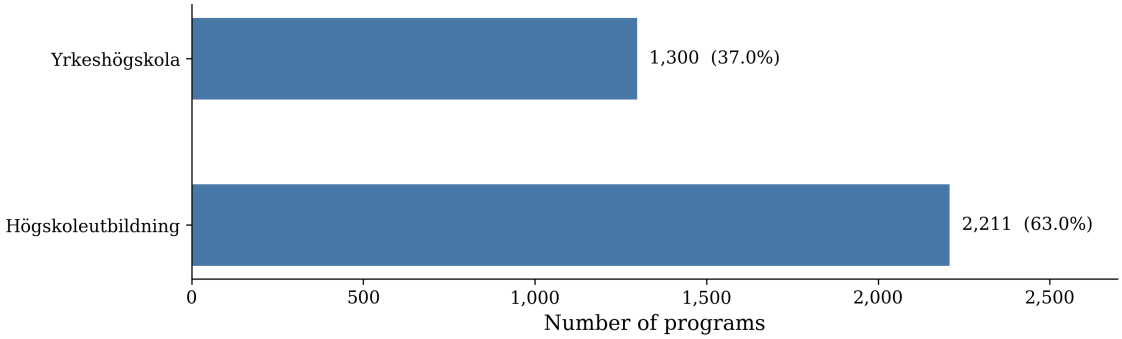


Figure 4.1: Distribution of education forms across all programs in the filtered corpus.

The two-to-one ratio between university and vocational programs means that unconstrained retrieval will naturally favour university programs. This imbalance motivates the use of education-form filters in the retrieval pipeline and stratified analysis when evaluating recall on vocational queries.

Delivery mode: remote versus on-site

Figure 4.2 shows the distribution of the boolean `is_distance` flag. The vast majority of programs (3,271, or 93.2%) are offered on-site; only 240 programs (6.8%) are classified as remote education.

This strong skew towards on-site programs means that remote-specific queries should be evaluated as a separate slice rather than expected to achieve the same recall as on-site queries.

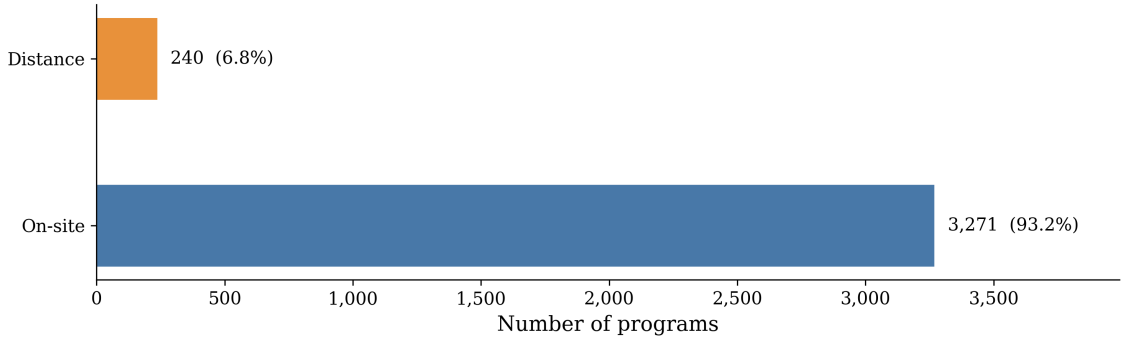


Figure 4.2: Distribution of remote vs. on-site delivery at program level for all programs in the filtered corpus.

Geographic distribution

Municipality coverage is derived from the `municipality_code` field, which lists Swedish municipality codes associated with each program. For analysis, these codes are mapped to human-readable municipality names and counts are aggregated across all programs.

The corpus spans 144 distinct municipalities. Figure 4.3 shows the top twenty ranked by program count. Stockholm leads with 487 programs, followed by Göteborg (349), Malmö (208), Uppsala (163), and Umeå (153). Several other university towns appear prominently, including Linköping, Jönköping, Växjö, and Lund.

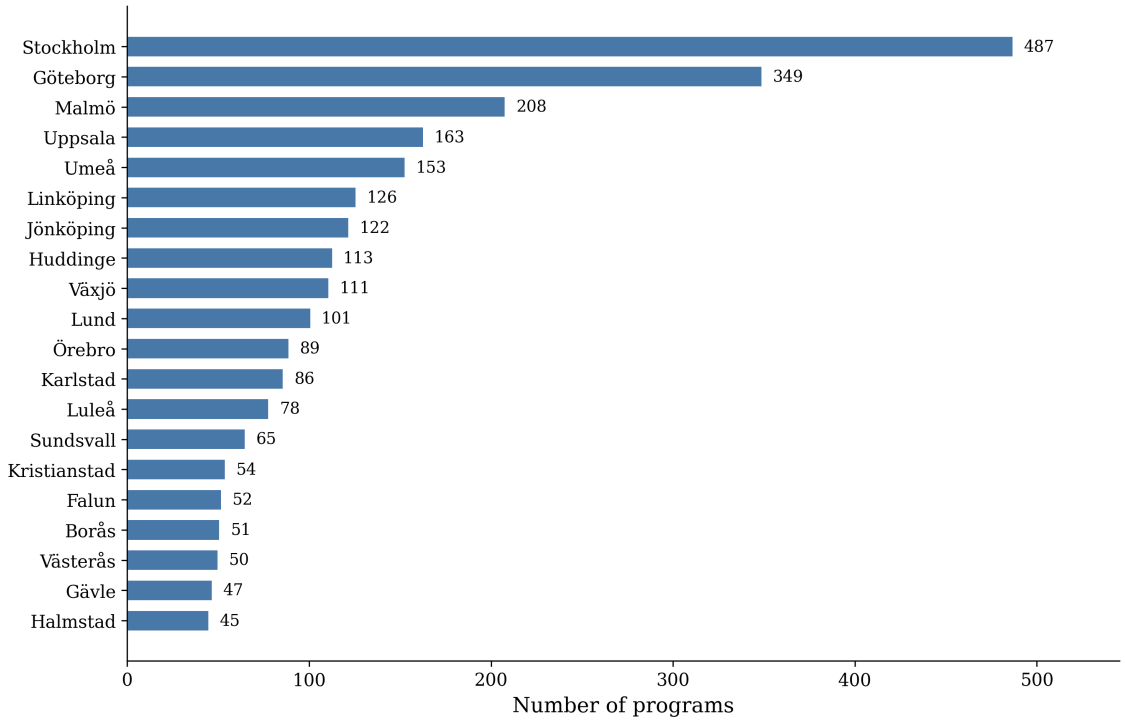


Figure 4.3: Top twenty municipalities ranked by number of programs in the filtered corpus.

This concentration supports the use of municipality-level metadata filters to ensure that

location-constrained queries are not dominated by programs in the largest university cities.

Provider distribution

The dataset has a total of 293 distinct providers. Figure 4.4 shows the top twenty providers ranked by program count. Stockholms universitet leads with 209 programs, followed by Göteborgs universitet (177), Linnéuniversitetet (133), Uppsala universitet (132), and Umeå universitet (131). A second tier of large universities includes Lunds universitet, Linköpings universitet, Malmö universitet, and Jönköping University. Notably, the top twenty also includes vocational higher-education providers such as TUC Sweden AB, YH Akademin AB, and Göteborgs Stad Yrgo, reflecting the mixed composition of the corpus across education forms.

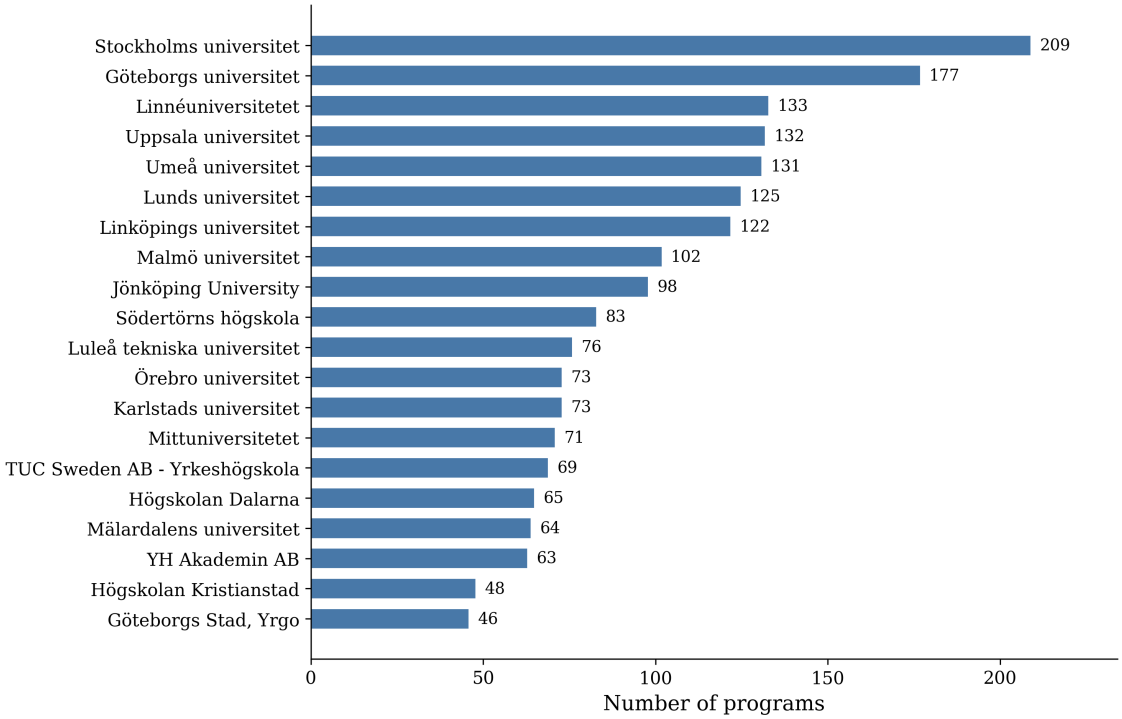


Figure 4.4: Top twenty providers ranked by number of programs.

From a retrieval perspective, provider is an informative but unevenly distributed feature: a small number of large institutions make up a disproportionate share of the corpus, so retrieval without filtering or diversification will tend to favor them.

These patterns shaped both the retrieval design, for example through filters and facets, and the evaluation setup used in later chapters.

4.2 Baseline System Architecture

Skoolie's architecture separates concerns between an *application backend*, which owns user accounts, conversations, and chatbot orchestration logic, and a *retrieval backend*, which owns

data ingestion, embedding computation, and retrieval endpoints. The two services communicate over internal APIs. The frontend presents a conversational interface alongside recommendation cards.

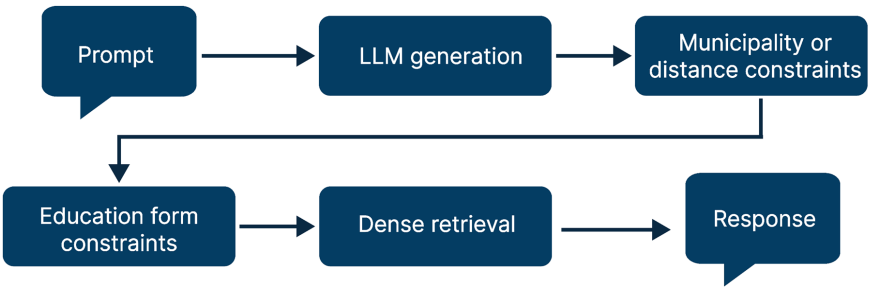


Figure 4.5: Overview of the baseline system's pipeline.

Baseline request flow. The sequence of operations for a single user turn proceeds as follows in the original (pre-RAG) system architecture (Figure 4.5):

1. The user message is received by the application backend, which prepends a fixed system instruction to guide tone and format.

2. The LLM (Azure OpenAI `gpt-4.1`) generates a free-text reply *without access to program data*. It also outputs a boolean flag indicating whether program recommendations should be appended.

3. If recommendations are requested, a short summary of the recommendation intent is derived from the LLM's output and forwarded to the retrieval backend as a search query.

4. The retrieval backend embeds the query, performs dense cosine-similarity search (nearest-neighbour lookup in a continuous vector space, as opposed to keyword-based lexical matching) over the education corpus (Section 4.3), applies any active filters, and returns the top-50 matching programs.

5. The application backend maps the returned programs to recommendation cards (title, provider, location, eligibility, links, and admissions statistics when available) and persists them.

6. The frontend displays the LLM's free-text reply together with the recommendation cards, if there are any.

The critical architectural property is that the LLM produces its answer at step 2 without seeing the programs retrieved at step 4. The LLM's output does influence the search query at step 3, but the dependence is one-directional: the retrieved programs never feed back into the generated text. This decoupling is the structural limitation that the RAG redesign (Chapter 5) addresses.

4.3 Baseline Dense Retrieval Endpoint

The retrieval backend exposes a FastAPI endpoint (`educations/suggest`) that accepts a free-text query and returns the top-k most similar programs with optional filters. Retrieval is *dense-only*, using a Sentence-BERT embedding model trained on Swedish data. All program records and their embedding vectors are stored in a PostgreSQL database using the pgvector extension for vector operations.

Request handling. The endpoint accepts a text query (`text_input`) and optional filter values: `education_types`, `education_forms`, `municipality_codes`, `is_distance`, and `limit`. Filters are applied as SQL `WHERE` clauses to narrow the candidate set before ranking.

Embedding. The `text_input` is encoded into a 768-dimensional vector using the same Swedish Sentence-BERT model used to embed the program records at index time.

Similarity search. The query vector is compared against every stored program embedding in PostgreSQL using cosine distance:

$$d = \text{cosine\_distance}(\mathbf{q}, \mathbf{e}),$$

where $\mathbf{q}$ is the query vector and $\mathbf{e}$ is a program embedding. Results are ordered by ascending d (most similar first) and truncated to `limit`. Distances are converted to similarity scores $s = 1 - d$ for downstream use.

Summary. *text input* → *Swedish Sentence-BERT embedding (768-d)* → *cosine-distance ranking over PostgreSQL/pgvector* with optional filters → *top-k results.*

Chapter 5
RAG Architecture and Design Variables

This chapter describes the retrieval-augmented generation (RAG) pipeline evaluated in this thesis, as implemented in Skoolie's environment. Figure 5.1 illustrates the full pipeline, and the sections that follow describe each stage in detail, in the order in which they execute: query routing, filter construction, hybrid retrieval, reranking, schema-aware context packaging, and grounded generation.

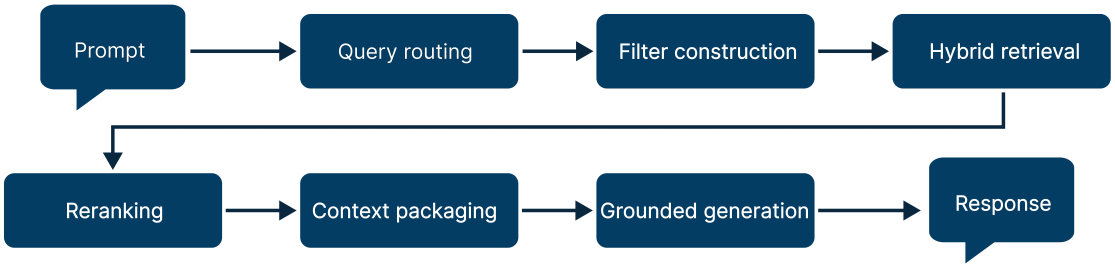


Figure 5.1: Overview of the RAG pipeline.

5.1 Query Routing

Not every conversational turn requires external knowledge. A student asking for encouragement or clarifying a previous answer should receive a direct response without incurring the latency cost of retrieval. Query routing addresses this by deciding, on a per-turn basis, whether to invoke the retrieval pipeline at all. RAG surveys describe such routing as a critical mechanism for matching retrieval depth to query requirements and for preventing the injection of irrelevant evidence into the prompt [6, 18].

Each incoming message is processed by a lightweight LLM router operating at temperature 0 (deterministic output, no sampling randomness) with a strict JSON output constraint.

The router receives the current user message together with the most recent conversation history (up to six messages), allowing it to resolve context-dependent references such as pronouns or follow-up questions. It outputs a JSON object with four keys:

- `fetchContext`: a boolean indicating whether education retrieval should be performed.
- `searchText`: a Swedish keyword-style query optimised to match program titles, descriptions, and eligibility text in the corpus.
- `intent`: one of `education_recommendation`, `question`, or `other`.
- `reason`: a brief justification for the routing decision.

The router is explicitly instructed not to reuse the user's exact phrasing, but to translate intent into catalogue-style terminology aligned with how programs are described in the corpus. If the user writes in a language other than Swedish but retrieval is nonetheless required, the router generates the search phrase in Swedish to match the corpus language. This query rewriting is conceptually related to HyDE (Hypothetical Document Embeddings) [5], which bridges the query–document gap by embedding a generated hypothetical answer rather than the raw query. When `fetchContext` is `true`, the pipeline proceeds to filter construction and retrieval; otherwise, a response is generated directly from the conversation history without injecting any program context. The full system instructions for the router are not included here due to confidentiality constraints, but the key design choices are described above.

5.2 Filter Construction

Before retrieval, structured filters are derived from the conversation history to constrain the candidate set to programs plausibly relevant to the student's stated context. Filter construction is performed by an LLM-assisted extraction step that identifies four types of constraints:

- **Municipality codes.** If the student has mentioned a preferred study location, the municipality name is resolved against a full catalogue of Swedish municipalities and translated to the corresponding code used in the retrieval index.
- **Remote studies.** If the student has explicitly requested remote study, the filter sets `is_distance = true`. When this flag is active, no municipality constraint is applied, since remote programs are not geographically bounded.
- **Providers.** If the student has named a specific institution, it is passed as a provider constraint.
- **Education forms.** By default, retrieval is restricted to `högskoleutbildning` (university education) and `yrkeshögskoleutbildning` (vocational higher education), reflecting the current scope of the platform. If the conversation indicates a preference for one form over the other, the filter is narrowed accordingly.

These filters are applied identically across both branches of the hybrid retrieval step described in the following section, and their correctness is therefore a prerequisite for retrieval quality throughout the pipeline.

5.3 Hybrid Retrieval

The educational guidance queries that arise in Skoolie span a wide range: from precise entity queries such as *"datateknik i Lund"* (computer science in Lund) to fully exploratory queries such as *"något kreativt där jag jobbar med människor"* (something creative where I work with people). No single retrieval strategy handles this range well. Sparse lexical retrieval is effective when users mention program titles or institution names, but degrades on paraphrased or interest-based input [16]. Dense semantic retrieval captures conceptual similarity but may miss exact-match candidates [7]. Hybrid retrieval combines both signals and is widely adopted as a strong baseline in practical RAG systems [6, 18].

Hybrid search is implemented in the Python retrieval service (PostgreSQL and pgvector) and executes two independent SQL queries whose results are merged prior to ranking.

Vector search. The query text is embedded using the same Swedish Sentence-BERT model used to index the corpus. Cosine similarity is computed between the query embedding and all stored program embeddings subject to the active filters, where p denotes a candidate program:

$$s_{\text{vec}}(p) = 1 - \cos(e(p),\, e(q)).$$

The top candidates by similarity are retained up to the configured candidate limit.

BM25 search. BM25 is a lexical ranking function that scores documents based on term frequency, inverse document frequency, and document length. Here it is implemented as a PostgreSQL full-text search against a `tsvector` constructed from the *title and provider fields only*, using the Swedish language configuration. The raw `ts_rank_cd` score is normalised to $[0, 1]$:

$$\tilde{b}(p) = \min\left(\frac{\texttt{ts\_rank\_cd}(\texttt{tsvector}(p),\, \texttt{tsquery}(q))}{2},\ 1\right).$$

Only candidates for which the tsquery matches the title or provider text are returned by this branch. Because description and eligibility text are excluded from the BM25 index, lexical boosts are limited to queries that contain recognisable program or institution names. For fully exploratory queries the vector branch therefore dominates.

Score fusion. The two result sets are merged by `education_id`. If a program appears in both lists, both scores are kept. If it appears in only one, the missing score is set to zero. The hybrid score is a weighted combination:

$$s_{\text{hybrid}}(p) = \alpha\, \tilde{b}(p) + (1 - \alpha)\, s_{\text{vec}}(p), \qquad \alpha \in [0, 1],$$

where α controls the relative weight of the lexical signal. In the deployed configuration, $\alpha = 0.8$, which heavily weights the lexical signal when a BM25 match exists. This reflects a deliberate "lexical priority" design: when the BM25 branch returns matches, the high weight ensures that exact title or institution hits are ranked above semantically similar but lexically unmatched candidates. When no BM25 match exists, which is the typical case for exploratory or interest-based queries, the BM25 branch returns an empty set and contributes no candidates to the merge. In that case the ranked list consists entirely of vector-search candidates, and the ranking reduces to pure dense retrieval order.

The merged list is sorted by s_{hybrid} in descending order and truncated to a candidate limit of 36. This limit was chosen to balance two constraints: it must be large enough to give the downstream reranker a sufficiently diverse pool (the reranker selects the top 8), yet small enough that the combined retrieval and reranking latency remains within the approximately 2 s budget allocated to these stages (see Chapter 6). Empirically, increasing the limit beyond 36 yielded diminishing improvements in top-8 candidate quality while adding measurable retrieval latency.

5.4 Reranking

Only a small number of education programs can be injected into the LLM prompt, therefore precision at the very top of the ranked list has a great influence on the final response. Reranking addresses this by applying a more expressive relevance model to the retrieved candidate pool before the top results are selected for the prompt. Cross-encoder rerankers score query–document pairs jointly and typically achieve substantially higher top-k precision than bi-encoder similarity alone [13, 6].

Reranking is implemented using the Azure-hosted `Cohere-rerank-v4.0-pro` model. For each candidate, a plain-text document string is constructed by concatenating five fields: title, description, provider, location, and education form. This string is submitted to the reranker alongside the original search query, and the reranker returns a relevance score and a new ordering.

The step is protected by a hard timeout of 3 000 ms to guard against upstream unavailability, though in practice the reranker has operated reliably within this budget.

5.5 Schema-Aware Context Packaging

Context packaging determines how the top-k candidates are serialised into the prompt. LLMs exhibit sensitivity to formatting and to position effects in long contexts [12], so a schema-consistent representation reduces prompt variance across queries and makes it easier for the model to locate and compare specific attributes across candidates. Compact representations also reduce token consumption and, consequently, generation latency.

The choice of structured JSON over free-text concatenation is motivated by the nature of the data. The program records are not continuous text but structured objects with discrete fields (title, provider, eligibility requirements, admission scores). Serialising them as labelled key-value pairs preserves this structure in the prompt, allowing the model to reliably extract and compare individual attributes across candidates. A free-text representation would require the model to infer field boundaries from context, which introduces ambiguity and increases the risk of misattributing information between programs [6].

The top $k = 8$ candidates are serialised using a packaging function that enforces two invariants. First, every record uses the same field ordering (Table 5.1). This ensures that positionally sensitive attributes such as title and provider always appear near the top of each record, regardless of the shape of the underlying data object. Second, free-text fields are truncated before serialisation: descriptions are capped at 400 characters and eligibility text at 300 characters, with truncated strings suffixed by `"..."` to signal incompleteness to the

model.

The packed records are JSON-serialised and injected into the LLM's system message alongside a brief field-schema description so the model knows the semantics of each key. Table 5.1 lists the fields and their truncation policy.

Table 5.1: Fields included in each packed education record and their truncation policy.

Field	Content	Truncation
`id`	Education identifier	None
`title`	Swedish program title	None
`provider`	Institution name	None
`description`	Swedish program description	400 chars
`location`	Municipality / city	None
`terms`	Duration in terms (derived)	None
`paceOfStudy`	Study pace percentage	None
`distance`	Remote / On Site	None
`eligibility`	Swedish eligibility text	300 chars
`admissionNotes`	Latest BI/HP scores + explanation	None
`links`	Education and application URLs	None
`educationForm`	Form code	None
`educationType`	Type code	None
`relevance`	Rerank or similarity score	None

The `terms` field is derived from program start and end dates by computing the interval in months and dividing by six, reflecting the standard Swedish academic calendar. If either date is unavailable, the field is omitted.

Before packaging, candidates are enriched with the most recent admission score data (grade-based entry `BI` and Swedish aptitude-test `HP` lanes) via a batch call to an external statistics service. Critically, the same candidate objects packaged into the prompt are also persisted as recommendation cards in the UI, ensuring that the textual explanation and the displayed cards are always grounded in the same retrieved set.

5.6 Grounded Generation and LLM Selection

The final stage generates a conversational response conditioned on the packed education records. The LLM (Azure OpenAI `gpt-4.1`) receives a structured prompt containing the chatbot system instruction, the student's grade context, the packed records with the field schema description, and the current user message. Crucially, the model does not merely compose an answer: it also performs a second-stage selection over the injected candidates, choosing which programs to include as recommendations.

The system prompt constrains the model's domain to Swedish education and career guidance, with an explicit refusal instruction for out-of-scope queries. It also enforces a structured Markdown output format and limits the model to one follow-up question per turn. These

constraints shape both the tone and the length of the response. Notably, the single-question rule influences how student context accumulates across turns, which in turn affects the quality of filter construction in subsequent retrievals.

The model is constrained to return a JSON object with two fields:

- `content`: the natural-language answer presented to the student.
- `educationIds`: an ordered list of education IDs selected from the injected candidates, ranked from most to least relevant.

Only the programs identified in `educationIds` are persisted and displayed as recommendation cards, in the specified order. This means the model controls both the explanation and the recommendation list, which is what ensures the two are aligned.

Chapter 6
System Performance & Latency Engineering

Adding reranking, context packaging, and a routing stage to the pipeline raises an obvious question: does the RAG system respond slowly enough to hurt usability? This chapter measures end-to-end response time for both architectures and breaks down where time is spent in the RAG pipeline.

6.1 Response Time and Measurement Approach

In the deployed interface, the user sees an answer only after the full backend pipeline completes. There is no streaming or progressive disclosure of intermediate results. Perceived performance is therefore governed by a single metric: **Time-to-Visible-Answer (TTVA)**, the total time from prompt submission until the response appears in the interface.

RAG introduces stages that the baseline does not have (reranking, context packaging, and a routing step), so it is reasonable to expect it to be slower. Whether that expectation holds, and by how much, has to be measured.

6.2 Experimental Setup

Both systems were run on the same set of 30 prompts, all constructed to trigger retrieval. They were evaluated back-to-back on the same machine, using the same data snapshot, database instance, and network configuration, with no other workloads running.

Since both systems call external services (Azure OpenAI for generation, Cohere for reranking), the measured times include network round trips and provider-side processing that can vary between runs. Each prompt was run once per system rather than averaged over repeated runs, and no warm-up requests were discarded. The results therefore reflect realistic single-

run performance under matched conditions. Given the external service variability, small differences in individual prompt times should be interpreted with caution.

6.3 Overall Response Time

Metric	Baseline (s)	RAG (s)
Mean	6.66	5.98
Median	6.47	5.82
p90	8.07	6.71
p95	8.99	6.74
Minimum	5.31	4.79
Maximum	12.32	12.02

Table 6.1: End-to-end TTVA for baseline and RAG architectures (30 matched prompts).

Statistic	Difference (s)	Relative Change
Average improvement	0.68	10.2% faster
Median improvement	0.62	9.6% faster
RAG faster prompts	24 / 30	80% of cases
Largest improvement	3.20	48% faster
Largest regression	1.40	21% slower

Table 6.2: Prompt-level TTVA differences. Positive values indicate RAG is faster.

Tables 6.1 and 6.2 report end-to-end TTVA and prompt-level differences for both architectures. RAG reduces median latency by roughly 10%. The more meaningful gain is in the worst cases: p95 drops by about 2.2 seconds, which means the slow responses that frustrate users become substantially less common.

6.4 Why RAG Is Not Slower

The RAG pipeline adds stages the baseline does not have (routing, reranking, context packaging), yet it is faster on 80% of prompts. The explanation is that the baseline was already doing much of the same work — filter extraction, retrieval, candidate enrichment — but in a more fragmented and less predictable way.

The baseline retrieves up to 50 candidates. RAG retrieves 36, which is enough to give the reranker a diverse pool while keeping retrieval time bounded.

Candidate enrichment in the baseline happens per item: each program triggers its own occupation lookup and a conditional database query for admission statistics. These sequential calls add up, especially when the candidate list is long. RAG batches the enrichment into a single operation.

Filter construction in the baseline is spread across multiple LLM calls: one for municipality and remote constraints, another for education form. RAG consolidates this into a single extraction step.

None of these differences are individually dramatic, but together they shorten the critical path enough to more than offset the cost of reranking (250 ms) and the slightly larger prompt that context packaging produces. The net effect is not that RAG does less work, but that it does similar work with less iteration and less per-item overhead. This primarily reduces worst-case latency: the slow outliers in the baseline come from retry loops and long enrichment chains, both of which RAG avoids.

6.5 Stage-Level Latency in the RAG Pipeline

To understand where time is spent within the RAG pipeline, each stage was instrumented and averaged across the 30 prompts (Table 6.3).

Pipeline Stage	Avg Time (s)	Share of Total
Response generation	2.2	37%
Retrieval	1.6	27%
Query routing	1.0	17%
Filter construction	0.6	10%
Reranking	0.25	4%
Remaining overhead	0.2	5%

Table 6.3: Average stage-level latency in the RAG pipeline.

Generation dominates, accounting for over a third of TTVA. Retrieval is the second-largest contributor. Reranking, despite reshuffling roughly 60% of the top-10 composition (Section 8.3.1), adds only about 250 ms on average and is not a bottleneck. Filter construction is visible at 10% but stable across prompts.

The practical takeaway is that latency optimisation efforts that ignore generation cost will have limited impact. Prompt size and output length are the primary levers.

6.6 Engineering Implications

The measurements point to three priorities for keeping TTVA within the product's budget as the system evolves:

1. Cap the number of candidates injected into the prompt and constrain output verbosity, since generation is the dominant cost.

2. Keep a fixed upper limit on retrieved candidates.

3. Batch metadata enrichment rather than making per-item calls.

Summary

Despite adding routing, reranking, and context packaging stages, the RAG pipeline is faster than the baseline on most prompts and substantially reduces the worst-case response times. The gain comes not from doing less work but from organising similar work more efficiently: fewer LLM calls for filter construction, a smaller candidate set, and batched enrichment instead of per-item calls. Generation remains the dominant cost, making prompt size the primary lever for further improvement.

Chapter 7
Experimental Setup

7.1 Goal and Research Questions

The overall goal is to evaluate whether a retrieval-augmented generation (RAG) pipeline produces more grounded and UI-consistent guidance than the baseline generate-then-retrieve system.
RQ1. How does a retrieval-augmented generation architecture with schema-aware context packaging affect explanation–recommendation alignment (ERA) and factual grounding compared to a baseline generate-then-retrieve system?
RQ2. How do retrieval strategy (dense vs. hybrid) and cross-encoder reranking affect the relevance of retrieved candidates and the end-to-end latency of the pipeline under product-like deployment constraints?
RQ3. Do users perceive the RAG-based system as more helpful and accurate than the baseline system in a paired evaluation where respondents are unaware of which system produced each answer?

7.2 Systems Compared

We compare two versions of the chatbot:

- **Baseline:** The previous non-RAG system (Chapter 4), which generates answers without context injection. Recommendation cards are appended to the response after generation based on a separate retrieval step.
- **RAG:** The retrieval-augmented system described in Chapter 5, which retrieves relevant educational information and injects it as grounding context *before* generation. The same retrieved documents inform both the generated response and the recommendation cards displayed in the interface.

Both systems are evaluated on the same sets of queries. To ensure comparability, the educational data corpus is treated as a fixed snapshot for the duration of all evaluations; no changes are made to the underlying data during the study.

7.3 Evaluation Overview

The evaluation consists of three complementary methods, each targeting a different research question:

1. **Explanation–recommendation alignment (ERA).** Both systems are run on 30 prompts, and each response is checked for consistency with the recommendation cards shown in the UI. This evaluation addresses RQ1 (Section 7.4).

2. **Retrieval strategy and reranking comparison**, a controlled experiment comparing dense-only and hybrid retrieval, followed by a cross-encoder reranking evaluation. This evaluation addresses RQ2 (Sections 7.5–7.6).

3. **Human A/B study**, a blinded paired evaluation in which respondents rate answers from both systems on a combined helpfulness–correctness scale. This evaluation addresses RQ3 (Section 7.7).

The three methods differ in what they measure—alignment with the recommendations, retrieval relevance, and perceived answer quality—and together provide converging evidence on the effectiveness of the RAG pipeline. The remainder of this chapter describes the design and methodology of each evaluation in turn.

7.4 Explanation–Recommendation Alignment

7.4.1 Metric Definition

ERA measures whether the generated answer is about the same education programs that are displayed as recommendation cards in the UI, and whether the answer avoids introducing programs not present in the recommendation set.

For a query instance i, let R_i be the set of recommended programs (cards) and M_i the set of program entities mentioned in the generated response. We define:

$$T_i = |R_i \cap M_i| \quad \text{(true mentions)}, \qquad F_i = |M_i \setminus R_i| \quad \text{(false mentions)}.$$

Let $N_i = |R_i|$ denote the number of recommended programs. Mention precision and recall are:

$$\text{Precision}_i = \frac{T_i}{T_i + F_i}, \qquad \text{Recall}_i = \frac{T_i}{N_i}.$$

We summarise mention quality using the F1 score:

$$\text{F1}_i = \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}.$$

To avoid overstating alignment when the recommendation list itself contains irrelevant items, we apply a discount based on the number of unrelated recommended items U_i:

$$\mathrm{ERA}_i = \mathrm{F1}_i \cdot \left(1 - \frac{U_i}{N_i}\right).$$

ERA ranges from 0 to 1, where higher values indicate stronger explanation–recommendation coupling at the UI level. Figure 7.1 illustrates the three components with a concrete example.

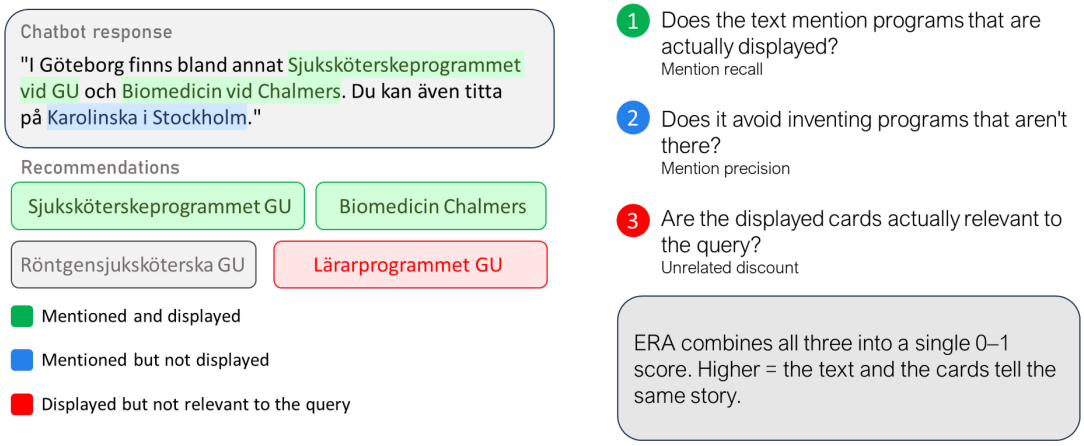


Figure 7.1: Illustration of the three components captured by ERA: mention recall, mention precision, and unrelated discount.

7.4.2 Operationalisation

Both system variants were run on the same 30 prompts, each producing a response text and an ordered list of recommendation cards, yielding 60 response–recommendation pairs in total.

A program was counted as *mentioned* if the response included its title or a recognisable variant (e.g. *"Socionomprogrammet vid Högskolan Väst"*). Each mention was matched against the recommendation cards: matches were recorded as true mentions (T_i), and references to programs absent from the cards as false mentions (F_i). A recommendation card was classified as *unrelated* (U_i) if the program had no discernible connection to the topic, field, or constraints expressed in the query. This can occur when the retrieval system returns programs that are semantically similar in the embedding space but do not match the user's actual intent.

The annotation was conducted by a single reviewer (the industry supervisor at Skoolie). Because no second annotator was available, inter-annotator agreement could not be computed. The judgements are relatively straightforward—a program is either named in the response or it is not, and a recommendation either fits the query topic or it does not—but some borderline cases inevitably involve judgement, and a second annotator would have made the scores more credible.

7.5 Retrieval Strategy Evaluation

To address the relevance component of RQ2, we conduct a controlled comparison of two retrieval strategies: dense-only retrieval and hybrid retrieval (dense combined with BM25

keyword matching). This evaluation is independent of the generation stage and measures how well each strategy surfaces the correct education programs in response to realistic user queries.

7.5.1 Query Design

We construct a test set of 100 queries distributed across three categories that reflect distinct user behaviours observed in production logs:

- **Category A — Provider-constrained queries ($n = 50$):** The user specifies both a program and an institution or city, e.g. *"Juristprogrammet Umeå universitet"* or *"Sjuksköterskeprogrammet Göteborg"*. These queries have a well-defined ground truth: the correct provider must appear in the top-k results. Category A is subdivided into three groups to test different naming conventions:
  - A1 — explicit university name ($n = 20$),
  - A2 — city name only ($n = 20$),
  - A3 — smaller or regional institutions ($n = 10$).
- **Category B — Topic-only queries ($n = 25$):** The user names a program or field without specifying a provider, e.g. *"Arbetsterapeutprogrammet"* (the occupational therapy program). Any result matching the program type is considered relevant.
- **Category C — Exploratory queries ($n = 25$):** Natural-language, intent-driven queries such as *"Jag vill jobba med HR och personalfrågor"* ("I want to work with HR and personnel issues"). Relevance is judged by topical fit to the expressed interest, no single correct answer exists.

The three-way split allows us to isolate the dimension on which dense and hybrid retrieval are expected to diverge most (entity-level provider matching in Category A) while verifying that hybrid retrieval does not degrade semantic search quality on the remaining categories.

7.5.2 Metrics

For Category A queries, relevance is defined as a binary label: a result is relevant if its provider field matches the institution specified in the query. We report the following rank-based metrics:

- **Hit@k** ($k \in \{1, 3, 5, 10\}$): the proportion of queries for which at least one relevant result appears in the top-k positions.
- **Mean Reciprocal Rank (MRR@10):** the average of the reciprocal of the rank of the first relevant result, computed over all queries:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i},$$

  where rank_i is the position of the first relevant result within the top 10, or ∞ (contributing 0) if no relevant result appears.

For Categories B and C, where ground truth is soft, we report MRR based on a topical relevance judgement: whether the top-ranked result matches the queried program type (B) or is topically appropriate to the expressed interest (C).

7.5.3 Procedure

Each of the 100 queries is submitted to both the dense-only and hybrid retrieval endpoints under identical conditions: the same data, the same embedding model, and the same filter configuration. The dense endpoint performs cosine-similarity search over pgvector (Section 4.3), the hybrid endpoint adds a BM25 branch as described in Section 5.3. Both return the top 10 results with relevance scores. Latency is recorded per request.

Because each query is submitted to both systems, the observations are paired. For Category A, where relevance is binary and the expected effect is large, we apply a Wilcoxon signed-rank test on per-query reciprocal ranks to confirm that the observed difference is not due to chance.

7.6 Cross-Encoder Reranking Evaluation

The retrieval strategy comparison in Section 7.5 establishes the first-stage retrieval baseline. This section describes a follow-up evaluation addressing the second component of RQ2: whether cross-encoder reranking improves the relevance of the final top-k results returned to the generation stage.

7.6.1 Pipeline Under Evaluation

The reranking stage operates as a two-stage retrieve-then-rerank pipeline. First, hybrid retrieval returns a candidate set of 36 documents. A cross-encoder model then rescores each candidate against the original query and selects the top 10 by reranker score. The evaluation compares the top-10 list produced by hybrid retrieval alone (*hybrid top-10*) against the top-10 list produced after cross-encoder reranking (*reranked top-10*).

7.6.2 System-Level Evaluation (100 Queries)

All 100 queries from the retrieval strategy evaluation (Section 7.5.1) were submitted to the full retrieve-then-rerank pipeline. For each query, both the hybrid top-10 and the reranked top-10 were recorded, along with per-stage latency and reranker confidence scores. System-level agreement between the two rankings is measured using three metrics:

- **Jaccard@10:** the size of the intersection divided by the size of the union of the two top-10 result sets, measuring overall composition overlap.
- **Top-1 agreement:** the proportion of queries where both rankings place the same document at rank 1.

- **New documents introduced:** the number of documents in the reranked top-10 that were not present in the hybrid top-10, drawn from positions 11–36 in the candidate pool.

These metrics quantify *how much* the reranker changes the ranking but do not directly assess whether the changes improve relevance. To address the quality question, we conduct a relevance-annotated evaluation on a subset of queries.

7.6.3 Relevance Evaluation (10 Queries)

A stratified sample of 10 queries was selected to cover the performance spectrum: four provider-constrained queries, three topic-only queries, and three exploratory queries. The selection was guided by variation in observed Jaccard@10 values and includes cases where hybrid retrieval already performs well, cases with substantial reranker reshuffling, and cases with known retrieval noise.

For each of the 10 queries, the union of the hybrid top-10 and the reranked top-10 was collected, yielding between 11 and 18 unique documents per query depending on how much the two lists overlapped. Each document was labelled as relevant or not by a single annotator (the supervisor at Skoolie) based on criteria defined before annotation began. For provider-constrained queries, a result was relevant if it matched the queried program type at any Swedish institution. For topic-only and exploratory queries, relevance was judged by whether the program fit the expressed interest.

Using the binary labels, we compute **Precision@10** and **nDCG@10** for both the hybrid and reranked top-10. The normalised discounted cumulative gain is defined as:

$$\text{nDCG@10} = \frac{\text{DCG@10}}{\text{IDCG@10}}, \qquad \text{DCG@10} = \sum_{i=1}^{10} \frac{\text{rel}_i}{\log_2(i+1)},$$

where $\text{rel}_i \in \{0, 1\}$ is the relevance label at rank i and IDCG@10 is the DCG of the ideal ranking given the total number of relevant documents in the judged pool.

7.6.4 Limitations

The relevance evaluation uses a single annotator and a non-random query sample of 10 queries. These limitations are partially mitigated by (i) the use of binary relevance labels on relatively unambiguous categories (a program either matches the queried type or it does not), (ii) the stratified selection covering all three query categories, and (iii) the complementary 100-query system-level metrics that confirm the patterns observed in the annotated subset.

7.7 Human A/B Study

The third evaluation component is a blinded human study addressing RQ3. Its purpose is to determine whether the quality differences measured by ERA and the retrieval evaluation are also perceived by end users and domain experts.

7.7.1 Evaluation Questions

The study uses 12 representative questions covering common study guidance needs:

- factual program information (e.g. length, location, eligibility, admission thresholds),
- constrained recommendations (e.g. location constraints, full-time study, remote learning),
- guidance and planning questions (e.g. step-by-step plans based on academic merit and constraints).

The questions were selected to vary in complexity and in the extent to which grounding in specific program data is required.

7.7.2 Participants

The survey was distributed to two target groups:

- **Career counselors** ($n = 3$): professional users with domain expertise in study guidance,
- **Students** ($n = 3$): end users representing the target audience.

The total sample size is $n = 6$. This is small and limits the statistical power of the study, we return to this limitation in the results chapter 8. The first survey question records the respondent role, enabling subgroup analysis.

7.7.3 Procedure and Blinding

A Google Forms survey is used to collect judgments. For each of the 12 questions, the respondent is shown two answers presented as screenshots labelled "Svar A" and "Svar B." One answer is produced by the baseline system and the other by the RAG system.

The mapping between system and label (A/B) is randomised across questions so that respondents cannot infer which system produced which answer. The presentation order of the question blocks is also randomised to reduce order effects and fatigue bias. Both screenshots use the same visual styling and layout to prevent respondents from distinguishing systems on the basis of interface differences.

7.7.4 Rating Scale and Outcome Measure

For each question, the respondent rates *each* answer on a single five-point Likert-scale item:

> *"How helpful and accurate is the provided answer for the student?"*

This combined item captures the respondent's overall assessment of both perceived helpfulness and correctness in a single rating. A free-text question at the end of the survey collects qualitative feedback.

7.7.5 Analysis Plan

After unblinding the A/B mapping, each score is attributed to either Baseline or RAG. Because each respondent rates both answers for the same question, the comparison is paired. We report mean and median scores per system (overall and per question) and compute the mean paired difference across all respondent–question pairs. Given the small sample size ($n = 6$), we treat the human evaluation as a directional pilot study and rely primarily on descriptive statistics rather than inferential significance tests. Results are split by respondent role (career counselor vs. student) to assess whether the direction of any observed preference is consistent across rater groups.

Chapter 8

Results

This chapter reports results from the three evaluations described in Chapter 7: (i) the metric-based ERA alignment evaluation addressing RQ1, (ii) the retrieval strategy and reranking comparison addressing RQ2, and (iii) the human A/B study addressing RQ3. The chapter concludes with a consolidated summary of findings across all three research questions.

8.1 Explanation–Recommendation Alignment

To quantify UI-level faithfulness between generated answers and the education cards shown to the user, we evaluate the ERA metric on the fixed set of 30 prompts described in Section 7.4.2. For each response we recorded the number of recommended programs, the number explicitly mentioned in the response, the number of non-recommended programs mentioned (false mentions), and the number of unrelated items in the recommendation set.

Results. The baseline system achieves a mean ERA of 0.266, while the RAG system achieves 0.501, an absolute improvement of +0.235 (Table 8.1). The improvement is primarily driven by a sharp reduction in false mentions: the baseline produces on average 0.533 false program mentions per answer, compared to 0.033 for the RAG system. This indicates that retrieval-conditioned generation constrains the model away from inventing alternative institutions or programs that are not present in the UI.

In terms of mention behaviour, the baseline has mean mention precision 0.643 and recall 0.313, whereas the RAG system reaches precision 0.729 and recall 0.518. The RAG pipeline also produces more focused recommendation lists: fewer unrelated items on average (2.0 vs. 4.867 per prompt) and fewer total recommendations (5.57 vs. 9.8), which improves the signal-to-noise ratio of the educational cards displayed to the user.

Metric	Baseline	RAG
Mean ERA	0.266	0.501
Mention precision	0.643	0.729
Mention recall	0.313	0.518
False mentions / answer	0.533	0.033
Unrelated recs / prompt	4.867	2.000
Avg. # recommendations	9.800	5.570

Table 8.1: ERA and mention statistics for baseline vs. RAG (30 prompts each).

Interpretation. These results are consistent with the expectation that tighter coupling between retrieval and generation improves explanation–recommendation coherence. The RAG system increases coverage of displayed cards (higher recall) and reduces references to programs absent from the recommendation cards (near-elimination of false mentions). It is worth noting, however, that the ERA evaluation does not assess the *correctness* of the generated text, only its alignment with the recommendation set. A response could achieve high ERA by faithfully describing irrelevant recommendations. The complementary human evaluation in Section 8.4 addresses this limitation by capturing perceived correctness.

8.2 Retrieval Strategy Comparison: Dense vs. Hybrid

This section reports results from the retrieval strategy evaluation described in Section 7.5, directly addressing the relevance component of RQ2.

8.2.1 Provider-Constrained Queries (Category A)

Table 8.2 summarises the headline metrics for provider-constrained queries ($n = 50$). The difference between the two strategies is substantial: hybrid retrieval places the correct provider at rank 1 in 90% of queries, compared to 12% for dense-only retrieval. MRR@10 increases from 0.26 (dense) to 0.92 (hybrid). A Wilcoxon signed-rank test on per-query reciprocal ranks confirms that the difference is statistically significant ($p < 0.001, n = 50$).

Metric	Dense	Hybrid
MRR@10	0.26	0.92
Hit@1	12%	90%
Hit@3	24%	94%
Hit@10	56%	96%

Table 8.2: Retrieval metrics for provider-constrained queries (Category A, $n = 50$). Hit@k is the proportion of queries where the target provider appears within the top k results.

Sub-category analysis. Table 8.3 breaks down Hit@1 by the three provider-constrained sub-categories. Dense retrieval performs poorly across all sub-groups (Hit@1 between 10% and 15%). Hybrid retrieval achieves near-perfect accuracy when the full university name is specified (A1: 95%) or when a smaller institution is named (A3: 100%), and somewhat lower accuracy for city-only queries (A2: 80%).

Sub-category	*n*	Dense Hit@1	Hybrid Hit@1
A1: University name	20	10%	95%
A2: City name only	20	15%	80%
A3: Smaller inst.	10	10%	100%
All Category A	50	12%	90%

Table 8.3: Hit@1 by provider-constrained sub-category.

Interpretation. Dense retrieval encodes the query into a single embedding that captures the semantic meaning of the program name (e.g. *nursing*) but assigns low weight to the named entity that specifies the provider. For instance, the query *"Sjuksköterskeprogrammet Göteborgs universitet"* returns nursing programs from Umeå, Malmö, and Blekinge at the top ranks under dense retrieval. Hybrid retrieval combines the dense score with a BM25 keyword match that gives exact-match credit when the provider or city name appears in the indexed fields, reliably surfacing the correct institution.

The lower hybrid Hit@1 for city-only queries (A2: 80% vs. A1: 95%) reflects the indirectness of city-to-provider mapping. A query mentioning *"Stockholm"* may match Stockholms universitet, Karolinska institutet, or KTH, and the BM25 component can only contribute when the city string appears in the provider metadata. Four of the five hybrid misses in Category A occur in this sub-category.

8.2.2 Topic-Only and Exploratory Queries (Categories B and C)

For queries that do not specify a provider, both retrieval strategies perform comparably (Table 8.4). The near-identical MRR values are expected: these queries are purely semantic, and the BM25 component in hybrid retrieval rarely produces matches against the title and provider fields, so the ranking is effectively determined by the dense scores alone.

Category	*n*	Dense MRR@10	Hybrid MRR@10
B: Topic-only	25	0.91	0.89
C: Exploratory	25	0.85	0.84

Table 8.4: MRR@10 for topic-only (B) and exploratory (C) queries. Both strategies perform comparably when no provider constraint is present.

This result is practically important: incorporating BM25 into the retrieval pipeline does not degrade semantic matching quality. The hybrid approach strictly dominates dense-only

retrieval in the provider-constrained setting while maintaining equivalent performance elsewhere.

8.2.3 Failure Analysis

Five of the 50 provider-constrained queries were not resolved at rank 1 by hybrid retrieval. We analyse these cases to characterise the failure modes:

1. **Ambiguous city mapping** (3 cases): Queries using a city name that maps to multiple institutions, e.g. *"Fysioterapeutprogrammet Stockholm"*, where Karolinska institutet does not contain "Stockholm" in its provider field.

2. **Cross-institutional title mismatch** (1 case): The query *"Personalvetarprogrammet Lund"* uses a program name offered under that exact title at several universities (e.g. Göteborgs universitet, Örebro universitet), but Lunds universitet offers the equivalent program under a different title (*"Kandidatprogram i Human Resources"*). While the dense component can in principle bridge the semantic gap between Personalvetar- and Human Resources, it assigns higher scores to the exact title matches at other institutions. The BM25 component matches *"Lund"* in the provider field but finds no lexical overlap in the program title, and the resulting combined score is insufficient to outrank the exact-title competitors. This case illustrates why the chatbot employs filter construction: when the same query is executed with Lunds university applied as a provider filter, the Human Resources program appears as the top result, since the competing exact-title matches at other institutions are excluded from the candidate set.

3. **Rank displacement** (1 case): The correct provider appears within the top 10 but not at rank 1, displaced by a closely related program at a different institution with a stronger combined score.

In the deployed chatbot, these cases are largely handled by the filter construction step (Section 5.2), which constrains retrieval to the intended provider when one can be identified from the conversation.

8.3 Cross-Encoder Reranking Results

This section reports results from the reranking evaluation described in Section 7.6, addressing the second component of RQ2.

8.3.1 System-Level Agreement (100 Queries)

Table 8.5 summarises the system-level comparison between the hybrid top-10 and the reranked top-10 across all 100 queries.

Metric	Value
Mean Jaccard@10	0.404
Top-1 agreement	41%
Mean new documents introduced	4.5 / 10
Mean reranking latency	231 ms
Reranker share of total latency	46%
Mean top-1 reranker score	0.938

Table 8.5: System-level agreement and latency metrics for hybrid retrieval vs. cross-encoder reranking (100 queries, top-10, candidate pool of 36 documents).

The mean Jaccard@10 of 0.404 indicates that the reranker replaces or reorders roughly 60% of the top-10 composition. On average 4.5 of the 10 reranked results were not present in the hybrid top-10 and were promoted from positions 11–36 in the candidate pool. This confirms that the two-stage architecture is not redundant, the reranker draws substantively on the broader candidate set rather than merely reordering the first-stage results.

Top-1 agreement varies across query categories (Table 8.6). Provider-constrained queries show 64% agreement, indicating that hybrid retrieval already identifies the correct top result in most cases when an exact program–provider match exists. Exploratory queries show only 16% agreement, indicating that the reranker produces the largest ranking changes on semantically complex, intent-driven queries where lexical matching alone is insufficient.

Category	n	Mean Jaccard@10	Top-1 agreement	Mean new docs
A: Provider-constrained	50	0.427	64%	4.2
B: Topic-only	25	0.459	48%	3.8
C: Exploratory	25	0.295	16%	5.6
All	100	0.404	41%	4.5

Table 8.6: System-level reranking agreement by query category. Lower Jaccard and top-1 agreement indicate greater reranker reshuffling.

8.3.2 Relevance Evaluation (10 Queries)

To assess whether the reranker's reshuffling improves relevance rather than merely changing composition, binary relevance labels were assigned to all documents in the union of the hybrid and reranked top-10 lists for a stratified sample of 10 queries (four provider-constrained, three topic-only, three exploratory). Annotation criteria followed those described in Section 7.6.3.

Across the 10 queries, reranking improves mean Precision@10 from 0.56 to 0.71 and mean nDCG@10 from 0.707 to 0.958. No query gets worse, the reranked ranking matches or exceeds the hybrid ranking in every case. Two patterns account for most of the gain.

Ranking concentration. On several queries the same relevant programs appear in both rankings, but the hybrid list places them at mid-range positions while the reranker

pushes them to the top. On an exploratory query about sport management programs, for instance, three relevant results sit at positions 5, 6 and 8 in the hybrid ranking but move to positions 1–3 after reranking, lifting nDCG@10 from 0.356 to 1.000 without changing which documents are in the list. This pattern, where the gain comes from better ordering rather than finding new relevant documents, accounts for the largest nDCG gains in the sample.

Provider anchoring. On provider-constrained queries, the hybrid ranking tends to fill lower positions with unrelated programs from the queried institution. For instance, a query for *Socionomprogrammet* at a specific university returns six unrelated programs from that institution (teacher education, social psychiatry) at positions 5–10 in the hybrid ranking. The BM25 component scores them highly because they match the provider name, regardless of whether the program is relevant. The reranker, which scores each query–document pair as a whole, replaces all six with Socionomprogrammet offerings from other universities, raising Precision@10 from 0.40 to 1.00.

8.4 Human Evaluation Results

This section reports results from the blinded human A/B study described in Section 7.7, addressing RQ3. Six respondents (three career counselors and three students) each rated both system outputs for all 12 evaluation questions, yielding 72 paired observations. Given the small sample, the analysis is primarily descriptive; we report effect magnitudes and consistency patterns rather than inferential significance tests.

8.4.1 Overall Results

Across all respondents and questions, the RAG system receives consistently higher ratings than the baseline on the combined helpfulness–correctness scale. The baseline achieves a mean score of 3.40 and the RAG system a mean of 4.22, corresponding to a mean paired improvement of +0.82 Likert points. Figure 8.1 illustrates the overall comparison.

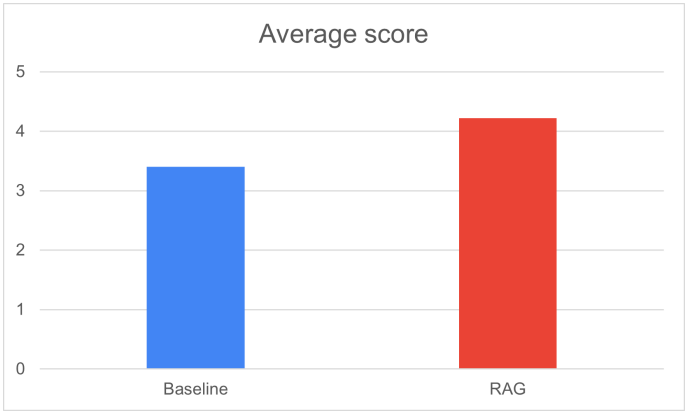


Figure 8.1: Overall average rating (1–5) for the baseline and RAG systems across all respondents and questions ($n = 6$ respondents, 12 questions).

The direction of the difference is consistent: every respondent assigns a higher mean score to the RAG system than to the baseline. While the small sample precludes strong inferential claims, the uniformity of the preference across all six respondents suggests that the observed difference is unlikely to be an artefact of individual rater variation.

8.4.2 Per-Question Analysis

Across the 12 evaluation questions, the RAG system outperforms the baseline on 9 questions, ties on one, and is rated lower on two (Figure 8.2). The largest gains occur on questions requiring precise factual grounding or constraint-aware guidance, such as admission requirements, program availability, and step-by-step planning based on academic merit.

The two questions where the baseline does slightly better are open-ended recommendation tasks where many reasonable answers exist. On these, the RAG system's reliance on retrieved candidates may have narrowed the response compared to what the baseline could produce without that constraint.

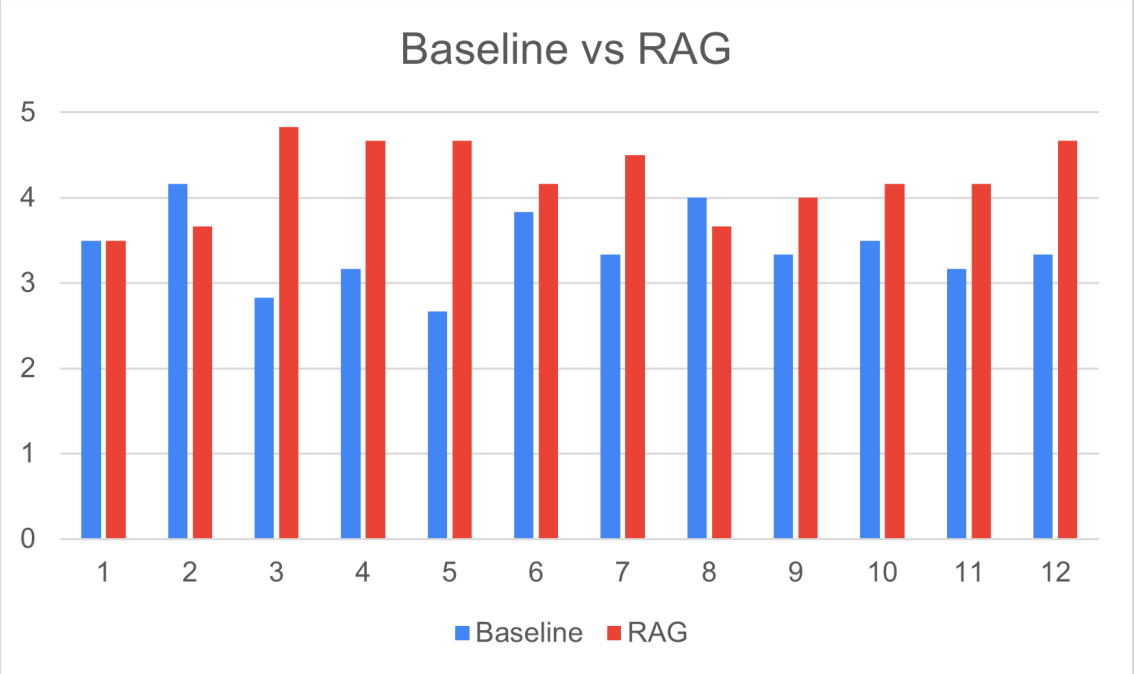


Figure 8.2: Mean ratings per question for the baseline and RAG systems. Each bar represents the average score across all respondents for a given question.

8.4.3 Factual vs. Reasoning Questions

Despite the two open-ended questions where the baseline scores higher, grouping all 12 questions by type shows that RAG improves ratings on both factual and reasoning/planning questions overall. Figure 8.3 summarises the results.

However, contrary to the initial expectation that retrieval would primarily benefit factual questions, the relative improvement is larger for reasoning and planning questions (approximately +1.0 Likert points) than for factual questions (approximately +0.7 points).

One possible explanation is that the single-item rating scale conflates helpfulness and correctness. Several student respondents reported difficulty determining which answer was factually correct when the two systems disagreed, which may have compressed rating differences on factual questions. By contrast, reasoning and planning questions emphasise structure, coherence, and actionable guidance—qualities that are more directly observable by respondents regardless of domain expertise. Retrieval-augmented grounding appears to support more concrete, step-by-step responses on these questions, producing a quality difference that is easier for respondents to detect and reward.

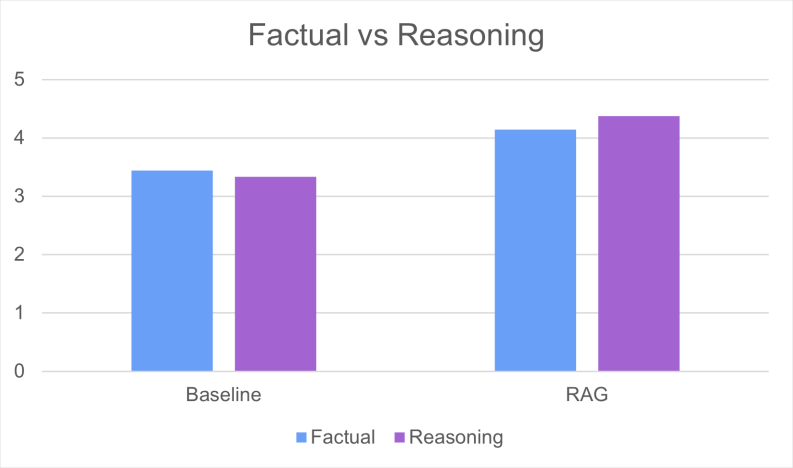


Figure 8.3: Average ratings by question type (factual vs. reasoning/planning) for the baseline and RAG systems.

8.4.4 Students vs. Career Counselors

Ratings were stratified by respondent role to assess whether the observed preference is consistent across rater groups (Figure 8.4).

Both groups prefer the RAG system. Career counselors assign notably lower scores to baseline answers than students do, particularly on questions involving factual correctness and realistic admissions guidance. Their ratings of the RAG system are consistently high, suggesting strong alignment with professional expectations of specificity and accuracy. Students also rate the RAG system higher, though their ratings show less variation between systems.

The consistency of the RAG preference across both domain experts and end users strengthens the directional evidence for RQ3, even though the sub-group sizes ($n = 3$ per group) are too small to support formal between-group comparisons.

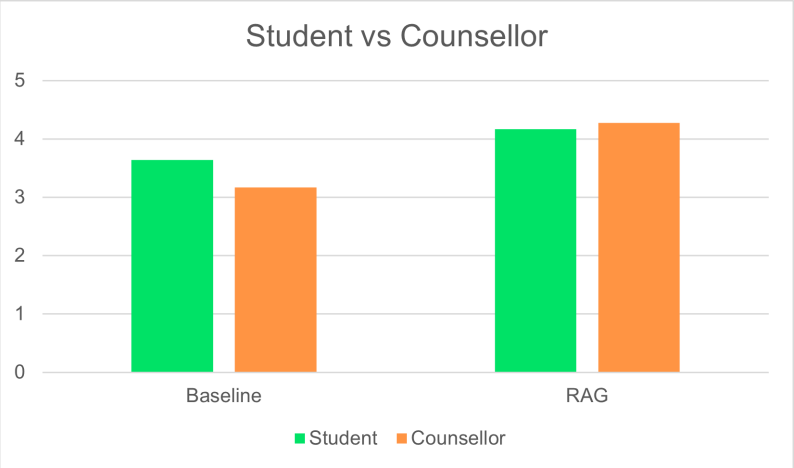


Figure 8.4: Average ratings by respondent role (students vs. career counselors) for the baseline and RAG systems.

8.5 Discussion

The three evaluations target different aspects of the pipeline: alignment with the recommendation cards, retrieval relevance, and perceived answer quality. Yet they point in the same direction. This section draws out the connections and flags where the evidence is weaker.

ERA and the human ratings complement each other. ERA checks whether the explanation stays within the displayed recommendation set. The human study checks whether respondents find the answer helpful and correct. These are not the same thing: a system could score well on ERA by faithfully describing irrelevant recommendations, and a system could get high human ratings with fluent text that references programs not on screen. That both metrics favour the RAG system makes the overall case stronger than either would alone.

The two evaluations used different prompt sets (30 for ERA, 12 for the human study), so whether they correlate at the individual question level is an open question. Running both on the same prompts would be a natural next step.

Why reasoning questions benefit more than factual ones. The human evaluation shows a larger improvement on reasoning and planning questions (+1.0 Likert points) than on factual ones (+0.7). This was unexpected, you would think retrieval matters most when the answer needs to cite specific facts. A likely explanation is that the retrieved context does not just supply facts; it gives the model concrete names, eligibility details, and admission thresholds to build a plan around. Without that, the model falls back on generic advice. Career counselors, who are better positioned to judge plan quality, show a larger preference gap than students, which supports this.

The two questions where the baseline wins. Both are open-ended recommendation tasks where many reasonable answers exist. When the retrieval set does not cover the full space of relevant options, the RAG system is constrained to what it retrieved, while

the baseline can draw on whatever the model knows. This is a known trade-off [6] and could be addressed by having the router treat exploratory queries differently, injecting retrieved context as background rather than as a hard boundary on what the model can say.

Reranker confidence. Across all 100 queries, the mean top-1 reranker score is 0.938. Queries scoring above 0.95 tend to have high retrieval precision, scores below 0.80 tend to coincide with retrieval problems. This was not tested as a runtime signal in the deployed system, but it suggests a low-cost way to flag uncertain responses without changing the pipeline architecture.

Annotation limitations. Both the ERA evaluation and the reranking relevance evaluation were annotated by a single person. The categories are simple enough that disagreement would likely be rare, but without a second annotator there is no way to verify that. This matters most for ERA, since it is a thesis contribution: a reader has no way to check whether the +0.235 improvement would hold under independent annotation.

8.6 Summary of Findings

Table 8.7 consolidates the key results across the three evaluation methods.

RQ	Evaluation	Key result	Evidence strength
RQ1	ERA (30 prompts)	ERA +0.235; false mentions ↓ 94%	Moderate (single annotator)
RQ2	Retrieval (100 queries)	MRR 0.92 vs. 0.26; $p < 0.001$	Strong (paired test, $n = 50$)
RQ2	Reranking (10 annotated)	nDCG +0.251; latency ≈ 231 ms	Moderate (small subset)
RQ3	Human A/B ($n = 6$)	+0.82 Likert; consistent direction	Preliminary (pilot scale)

Table 8.7: Summary of key findings by research question and evaluation method.

Across all three dimensions, the direction of the RAG advantage is consistent. The retrieval evaluation provides the strongest statistical evidence, ERA provides the most direct measure of UI-level alignment, and the human study confirms that the technical improvements translate into perceived quality gains.

Chapter 9
Threats to Validity

This chapter discusses limitations of the study using Shadish, Cook, and Campbell's validity typology as organised by Anglin et al. [1]: internal validity, statistical conclusion validity, construct validity, and external validity.

Internal validity. The main concern is whether the observed preference for RAG answers reflects the architectural change or something else. Some of the 12 evaluation questions may inherently favour retrieval-grounded responses, which would inflate the RAG advantage. Fatigue effects are possible since each respondent rated 24 answers in one session. And the two systems differ in more than just retrieval: prompt structure, response length, and formatting also changed, any of which could influence ratings independently.

Both systems were run on the same frozen data snapshot and the same question set. The A/B labels were randomised across questions and respondents to reduce position bias. But the evaluation is tied to one specific configuration, and causal claims should be read with that in mind.

Statistical conclusion validity. Six respondents is enough to see a direction but not enough to estimate effect sizes precisely. The paired design helps, since each respondent rates both systems on the same question, but the study is likely underpowered to detect small differences. The Likert scale adds measurement noise on top of that. The results are best read as a consistent signal rather than a precise estimate.

Construct validity. The human evaluation uses a single rating item that combines helpfulness and correctness. These are different things, and collapsing them into one score makes it impossible to tell which one is driving the preference. Several student respondents reported difficulty judging factual correctness when the two systems disagreed, which suggests the ratings may reflect clarity or plausibility more than verified accuracy.

ERA, evaluated separately, captures whether the response stays within the displayed card set. It does not check whether the stated attributes (e.g. eligibility requirements, locations)

are correct for each program, and it may undercount programs that are referenced indirectly rather than by name.

External validity. All three evaluations are scoped to the Swedish higher-education domain: 12 questions rated by six people for the human study, 100 queries for the retrieval comparison, and 30 prompts for ERA. The results may not hold for other national contexts, other educational domains, larger or more diverse user groups, or live multi-turn conversations where students interact with the system directly.

The system was also evaluated under a fixed retrieval configuration and data snapshot. A different corpus, embedding model, or LLM version could change the results. The findings should be read as evidence that the architecture works under these specific conditions, not as a general guarantee.

Chapter 10
Ethical Considerations

The system described in this thesis is deployed in schools, used by students making decisions about their futures. This raises questions about what the system can reasonably be trusted to do and where its limitations need to be made explicit.

The EU AI Act classifies AI systems used in educational and vocational training as high-risk, requiring attention to transparency, human oversight, and the protection of fundamental rights [3]. The considerations below are shaped by that regulatory framing, but they are also motivated by concrete tensions that arose during the design and evaluation of the system itself.

Information Provision versus Guidance There is an important difference between helping someone discover which programs exist and helping them figure out what to do with their life. Professional study and career counselling (*studie- och yrkesvägledning*) draws on pedagogical theory, psychology, and structured conversational methods. A counselor does not just list options—they support reflection, surface constraints the student may not have articulated, and adapt to the individual's circumstances in ways that require judgement and empathy.

The system in this thesis does none of that. It retrieves program records, packages them into a prompt, and generates a text that describes them. It can tell a student that Socionomprogrammet at Högskolan Väst is six semesters long and requires a specific set of prerequisite courses. It cannot tell a student whether social work is a good fit for them.

This distinction matters because presenting the system as a guidance tool rather than an information tool would misrepresent what it can do. Throughout the design, the system is framed as a complement to the counselor, not a substitute. Its job is to make educational options easier to find and compare, so that the limited time a student has with a counselor can be spent on the questions that actually require human judgement.

Automation Bias A practical worry is that students may trust the system more than it deserves. Research on automation bias shows that people tend to follow AI-generated ad-

vice even when it conflicts with their own assessment, simply because it comes from a system that appears confident and structured [9]. For a 17-year-old with no prior experience evaluating AI output about higher education, a fluent and specific response may look authoritative regardless of whether the underlying retrieval was good.

The RAG architecture partially mitigates this. Because the recommended programs are displayed as structured cards alongside the generated text, a student can check that the programs actually exist, read their eligibility requirements, and follow links to official sources.

But this safeguard has limits. A student may still defer to the system's implicit ranking of which programs matter most, or take the generated text at face value without clicking through to the cards. The ERA evaluation in Chapter 8 shows that the RAG system substantially reduces false mentions (programs named in the text but absent from the cards), which removes one source of misplaced trust. It does not eliminate the broader risk that students treat the system's framing as more definitive than it is.

During the human evaluation, one pattern was telling: several student respondents reported difficulty judging which of two answers was factually correct when the systems disagreed (Section 8.4.3). If respondents in a controlled evaluation struggle to verify correctness, students using the system in a classroom setting (without a second answer to compare against) are in an even weaker position. This reinforces the case for keeping a counselor in the loop rather than treating the system as self-sufficient.

What the System Cannot Do It is worth being explicit about what falls outside the system's competence, because the boundary is easy to cross in a conversational interface.

The system has no access to a student's motivation, family situation, financial constraints, or emotional state. It cannot weigh whether a student who says they want to study nursing is expressing a genuine interest or repeating something a parent suggested. It cannot recognise when a student is anxious, disengaged, or making choices under social pressure. These are exactly the situations where professional counselling is most valuable, and where an information retrieval system is least equipped to help.

Traceability as an Ethical Design Choice If the system cannot avoid all the risks above, it can at least make its reasoning inspectable. The core architectural decision of this thesis—grounding generation in retrieved records and displaying those same records as recommendation cards—is as much an ethical choice as a technical one.

In the baseline system, the generated text and the recommended programs came from separate processes with no guaranteed connection. A student reading the explanation had no reliable way to check whether the claims in the text corresponded to the programs on screen. The RAG redesign closes that gap: every program the model discusses is drawn from the same candidate set shown in the cards, and the ERA evaluation confirms that false mentions are nearly eliminated.

This traceability does not make the system trustworthy in any deep sense. But it makes it auditable. A counselor reviewing a student's saved conversation can see which programs the system retrieved, which ones it chose to highlight, and whether the explanation matches the cards.

Summary The system described in this thesis is an informational interface, not a counselor. It can lower barriers to discovering educational options, but it cannot replace the

human judgement required to contextualise those options within a student's life. Ethical deployment depends on maintaining that boundary clearly.

Chapter 11
Conclusion

11.1 Summary

The baseline system studied in this thesis had a straightforward problem: the chatbot wrote its answer first and retrieved programs second, so the explanation and the recommendation cards were produced independently. In practice, this meant the model could describe admission requirements, locations, or program characteristics that had nothing to do with the programs actually shown on screen. For a system deployed in schools, where students may not have the background to notice the mismatch, that is not an acceptable failure mode.

The RAG pipeline redesign addresses this by reversing the order of operations. Programs are retrieved and reranked before the model sees the prompt, then serialised into a schema-stable context payload so the model writes its answer with the actual candidates in front of it. A query router decides per turn whether retrieval is needed at all, and a filter-construction step narrows the candidate set based on constraints extracted from the conversation.

The evaluation provides evidence that this works, though with caveats about scale and annotation reliability.

On **RQ1**, ERA roughly doubles from **0.266** to **0.501**. The most striking change is in false mentions: the baseline averages **0.533** references per answer to programs not present in the UI cards, while the RAG system drops this to **0.033**. Mention recall rises from **0.313** to **0.518**, meaning the model discusses roughly half the recommended programs rather than all of them. This is arguably appropriate: listing every candidate would produce a catalogue rather than a conversational response, and the remaining programs are still visible as recommendation cards in the UI. The more important change is that the model almost never invents programs that are not there. This is a single-annotator result on 30 prompts, so the exact numbers should be read with that limitation in mind, but the direction and magnitude of the improvement are clear.

On **RQ2**, hybrid retrieval places the correct provider at rank 1 in 90% of provider-constrained queries versus 12% for dense-only, with no degradation on semantic queries. Cross-encoder

reranking adds ~250 ms and reshuffles about 60% of the top-10 composition, with consistent relevance gains. Despite the added pipeline stages, the RAG system is faster end-to-end (median TTVA 5.98 s vs. 6.66 s; p95 6.74 s vs. 8.99 s), owing to the efficiency gains discussed in Chapter 6.

On **RQ3**, the human A/B evaluation shows a consistent preference for RAG answers, with a mean improvement of +0.82 Likert points. Both students and career counselors prefer the RAG system, and the gap is largest on reasoning and planning questions, where the grounding context enables more concrete, step-by-step responses. Two open-ended recommendation questions go slightly in the baseline's favour, a pattern consistent with the retrieval bottleneck discussed in Section 8.5: when the retrieval set does not cover the full space of reasonable answers, grounding can constrain rather than help. The sample is too small for inferential claims, but the uniformity of direction across all six respondents makes it unlikely that the preference is noise.

Taken together, the results suggest that the architectural change from generate-then-retrieve to retrieve-then-generate produces measurable improvements in alignment, relevance, and perceived quality without breaking the latency budget. Whether those improvements are large enough to matter for real student outcomes is a question this thesis cannot answer. What it can say is that the baseline failure mode, where the chatbot confidently describes programs that do not match what the student sees, is largely eliminated.

11.2 Future Work

This thesis scopes the RAG pipeline to Swedish higher-education program records. The most useful extensions would broaden the types of questions the system can answer with grounded evidence rather than ungrounded generation.

11.2.1 Expanding the Knowledge Base Beyond Program Records

Application process guidance. Students frequently ask not about which program to choose but about how to apply: deadlines on Antagning.se, how merit points are calculated, how the aptitude test (*högskoleprovet*) factors into admission. The current system leaves these questions to the LLM's parametric knowledge, which is a hallucination risk for anything involving specific dates, cutoff scores, or procedural rules. Indexing official guidance documents from UHR (Universitets- och högskolerådet) as a secondary retrieval corpus, routed to when the query router classifies intent as `application_process`, would ground these answers without requiring fine-tuning.

Eligibility bridging through Komvux. A gap that came up repeatedly during evaluation is the handling of students who do not yet qualify for their target program. The system can retrieve and display eligibility requirements, but it cannot reason about how to bridge the gap. In the Swedish system, municipal adult education (Komvux) exists precisely for this purpose: students can take individual courses to obtain missing prerequisite grades. Incorporating Skolverket's regulatory framework for Komvux as retrievable content would

let the system suggest concrete next steps, such as which subjects to complete and where, rather than stopping at "you do not meet the requirements."

QA-style context for guidance questions. When the query router decides that no program retrieval is needed, the system currently falls back to ungrounded LLM generation. A curated corpus of question–answer pairs covering recurring topics like student finance (CSN loans), grade-point conversion, and campus life would give the system something to retrieve for these common queries. This is a well-understood pattern, sometimes called FAQ-augmented generation [6], and it fits naturally into the existing routing architecture.

11.2.2 Larger-Scale Human Evaluation

The human evaluation in this thesis involved six respondents and twelve questions, which was sufficient to identify a consistent directional preference but too small to estimate effect sizes precisely or to detect differences between question types with confidence. A natural next step would be a larger study with more respondents across multiple schools, using separate rating items for helpfulness and factual correctness rather than a single combined scale. Conducting the evaluation as a live deployment study, where students interact with the system directly rather than rating screenshot pairs, would also better capture how grounding affects real usage patterns over time.

Appendices

Appendix A
Human Evaluation Questions

The 12 questions used in the human A/B evaluation are listed below, grouped by type. All questions were presented in Swedish and evaluated via screenshot pairs as described in Section 7.7.3.

Factual program information.

1. Hur lång är Socionomprogrammet vid Högskolan Väst och på vilken ort ges det?
2. Har Uppsala universitet civilingenjörsprogrammet datateknik?
3. Vad är antagningsgränsen för systemvetenskap på Högskolan i Skövde?
4. Vilka förkunskapskrav krävs för att söka till högskoleingenjör?

Constrained recommendations.

5. Jag vill läsa något tekniskt efter gymnasiet i Göteborgsområdet, men inte civilingenjör. Vad finns det för alternativ?
6. Rekommendera fem ekonomirelaterade utbildningar i norra Sverige som är på helfart.
7. Kan du hitta yrkesförberedande utbildningar inom vård som inte kräver en högskoleförberedande gymnasieexamen?
8. Vilka program kan man läsa som leder till jobb som systemutvecklare och på vilka orter ges de?
9. Jag gillar programmering men inte matte. Vilka utbildningar tror du passar mig?

Guidance and planning.

10. Jag går i tvåan på gymnasiet och har ingen aning om vad jag vill bli. Ge mig en konkret plan för vad jag kan plugga utifrån mina betyg.

11. Jag går på gymnasiet och vill bli socionom men har just nu 11.64 i merit. Hur kan jag på ett realistiskt sätt ta mig dit efter gymnasiet baserat på mina betyg?

12. Jag har barn och kan inte flytta från min hemort. Hur kan jag på ett smart sätt planera för att bli systemutvecklare med distansutbildningar och andra alternativ, steg för steg?

Appendix B
ERA and Latency Prompts

The 30 prompts used for the ERA evaluation (Section 7.4.2) and the latency comparison (Chapter 6). All prompts were designed to trigger retrieval in both architectures.

1. Jag vill arbeta som sjuksköterska men vill helst studera i västra Sverige. Vilka alternativ finns?
2. Finns det juristprogram i norra Sverige som jag kan söka?
3. Jag är intresserad av datavetenskap men vill läsa på distans. Vad finns det för utbildningar?
4. Jag har cirka 18.0 i meritvärde. Vilka tekniska utbildningar är realistiska för mig?
5. Jag saknar Fysik 2. Finns det ingenjörsutbildningar jag ändå kan söka?
6. Jag vill läsa till lärare för årskurs 4–6. Var i Sverige erbjuds det?
7. Jag kan bara studera på halvfart. Finns det utbildningar inom psykologi som fungerar?
8. Jag är intresserad av kriminologi. Vilka program finns och var ges de?
9. Jag vill läsa något inom hållbar utveckling. Vilka program kan passa?
10. Jag vill arbeta inom HR i framtiden. Vilka utbildningar borde jag titta på?
11. Jag har 15.8 i meritvärde och är intresserad av ekonomi. Vad kan vara realistiskt?
12. Finns det masterprogram inom teknik som ges på engelska?
13. Jag vill arbeta med barn och unga. Vilka utbildningar kan leda dit?
14. Jag kan inte flytta från Malmö. Vad kan jag läsa där inom samhällsvetenskap?

15. Jag vill bli arbetsterapeut. Var kan jag läsa det?
16. Jag är intresserad av biomedicin men är osäker på behörigheten. Vad krävs?
17. Jag vill jobba inom IT-säkerhet. Vilka utbildningar kan vara relevanta?
18. Jag kan bara läsa 75% och behöver distans. Finns det utbildningar inom administration som passar?
19. Jag funderar på att läsa vidare efter min kandidat i ekonomi. Vilka masteralternativ finns?
20. Jag vill arbeta internationellt inom företagsekonomi. Vilka program kan ge den möjligheten?
21. Jag är intresserad av miljöfrågor och teknik. Finns det program som kombinerar det?
22. Jag saknar Naturkunskap 2. Finns det vårdutbildningar jag ändå kan söka?
23. Jag vill bli civilingenjör men vill inte studera i Stockholm. Vilka alternativ finns?
24. Jag har ungefär 17 i meritvärde och är intresserad av juridik. Vad kan vara realistiskt?
25. Jag vill arbeta inom kommunikation eller media. Vilka program kan passa?
26. Jag måste bo kvar i Uppsala. Vad finns det för utbildningar inom teknik där?
27. Jag är intresserad av beteendevetenskap och vill kunna arbeta inom offentlig sektor. Vad kan jag läsa?
28. Jag vill läsa något inom hälsa men inte bli läkare. Vilka alternativ finns?
29. Finns det utbildningar inom programmering som ges helt på distans?
30. Jag vill studera ekonomi men är osäker på skillnaden mellan kandidat och civilekonomprogram. Vad finns det för alternativ?

Appendix C
Retrieval Evaluation Queries

The 100 queries used in the retrieval strategy evaluation (Section 7.5.1), grouped by category.

Category A: Provider-constrained ($n = 50$)

A1: Explicit university name ($n = 20$).

1. Sjuksköterskeprogrammet Göteborgs universitet
2. Juristprogrammet Umeå universitet
3. Psykologprogrammet Uppsala universitet
4. Socionomprogrammet Högskolan Väst
5. Biomedicinprogrammet Karolinska institutet
6. Civilekonomprogrammet Linnéuniversitetet
7. Läkarprogrammet Linköpings universitet
8. Tandläkarprogrammet Malmö universitet
9. Fysioterapeutprogrammet Lunds universitet
10. Sjuksköterskeprogrammet Karolinska institutet
11. Datavetenskapligt program Stockholms universitet
12. Arkitektprogrammet Chalmers tekniska högskola
13. Personalvetarprogrammet Göteborgs universitet

14. Statsvetarprogrammet Stockholms universitet
15. Logopedprogrammet Uppsala universitet
16. Ämneslärarprogrammet Örebro universitet
17. Biologiprogrammet Karlstads universitet
18. Socionomprogrammet Mittuniversitetet
19. Systemvetarprogrammet Luleå tekniska universitet
20. Kandidatprogram i kriminologi Stockholms universitet

A2: City name only ($n = 20$).

1. Sjuksköterskeprogrammet Göteborg
2. Kriminologiprogrammet Malmö
3. Ekonomie kandidatprogram Lund
4. Civilekonomprogrammet Jönköping
5. Civilingenjör datateknik Linköping
6. Psykologprogrammet Stockholm
7. Juristprogrammet Uppsala
8. Läkarprogrammet Göteborg
9. Socionomprogrammet Örebro
10. Tandläkarprogrammet Göteborg
11. Fysioterapeutprogrammet Stockholm
12. Sjuksköterskeprogrammet Malmö
13. Personalvetarprogrammet Lund
14. Civilingenjör industriell ekonomi Linköping
15. Arbetsterapeutprogrammet Umeå
16. Grundlärarprogrammet Karlstad
17. Högskoleingenjör datateknik Jönköping
18. Kandidatprogram i filmvetenskap Stockholm
19. Medicinprogrammet Umeå
20. Kriminologiprogrammet Växjö

A3: Smaller or regional institutions ($n = 10$).

1. Sjuksköterskeprogrammet Högskolan Dalarna
2. Socionomprogrammet Högskolan i Gävle
3. Sjuksköterskeprogrammet Högskolan Kristianstad
4. Byggteknik Högskolan i Borås
5. Ekonomprogrammet Högskolan i Skövde
6. Sjuksköterskeprogrammet Mälardalens universitet
7. Folkhälsovetenskap Mittuniversitetet
8. Nätverksteknik Högskolan Väst
9. Idrottsvetenskapligt program Malmö universitet
10. Dataingenjör Blekinge tekniska högskola

Category B: Topic-only ($n = 25$)

1. Grundlärarprogrammet årskurs 4–6
2. Arbetsterapeutprogrammet
3. Högskoleingenjör maskinteknik
4. Medie- och kommunikationsvetenskap
5. Cybersäkerhet och nätverksadministration
6. Miljövetarprogrammet
7. Barnmorskeprogrammet
8. Civilingenjör kemiteknik
9. Kandidatprogram i företagsekonomi
10. Tandhygienistprogrammet
11. Röntgensjuksköterskeprogrammet
12. Dietistprogrammet
13. Högskoleingenjör elektroteknik
14. Kandidatprogram i statistik
15. Landskapsarkitektprogrammet

16. Optikerprogrammet
17. Audionomprogrammet
18. Civilingenjör bioteknik
19. Kandidatprogram i lingvistik
20. Fastighetsmäklarprogrammet
21. Receptarieprogrammet
22. Brandingenjör
23. Kandidatprogram i konstvetenskap
24. Civilingenjör teknisk fysik
25. Folkhälsovetenskapliga programmet

Category C: Exploratory ($n = 25$)

1. Jag vill jobba med fotboll och ekonomi
2. Jag gillar att programmera men vill inte räkna matte
3. Utbildning för den som vill hjälpa människor i kris
4. Jag vill jobba utomlands med affärer
5. Kreativ utbildning där man jobbar med ljud och musik
6. Något inom hälsa men inte bli läkare
7. Jag vill förstå varför människor begår brott
8. Jobba med barn och familjer i utsatta områden
9. Utbildning som kombinerar teknik och miljö
10. Jag vill jobba med HR och personalfrågor
11. Något inom data och AI på distans
12. Jag vill bygga hus och rita byggnader
13. Utbildning som leder till jobb inom offentlig förvaltning
14. Jag är intresserad av djur och natur
15. Jag vill jobba med sport och ledarskap
16. Jag vill hjälpa äldre människor

17. Utbildning för att jobba med klimatfrågor
18. Jag är intresserad av rymden och raketer
19. Vill jobba med marknadsföring på sociala medier
20. Jag vill jobba på museum eller galleri
21. Utbildning där man lär sig om pengar och investeringar
22. Jag vill utveckla mobilappar
23. Något som handlar om mat och hälsa
24. Jag vill jobba med mänskliga rättigheter internationellt
25. Utbildning för att starta eget företag

References

[1] Kylie Anglin, Qing Liu, and Vivian C. Wong. A primer on the validity typology and threats to validity in education research. *Asia Pacific Education Review*, 25:557–574, 2024.

[2] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation, 2025.

[3] European Parliament and Council of the European Union. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 june 2024 laying down harmonised rules on artificial intelligence (artificial intelligence act). Technical Report L 2024/1689, Official Journal of the European Union, 2024. Annex III, point 3(a): AI systems used in education and vocational training classified as high-risk. `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L_202401689`.

[4] Yue Feng, Shuchang Liu, Zhenghai Xue, Qingpeng Cai, Lantao Hu, Peng Jiang, Kun Gai, and Fei Sun. A large language model enhanced conversational recommender system, 2023.

[5] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels, 2022.

[6] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023. v5, 27 Mar 2024.

[7] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*, 2020. arXiv:2004.04906.

[8] Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of SIGIR*, 2020. arXiv:2004.12832.

[9] Artur Klingbeil, Cassandra Grützner, and Philipp Schreck. Trust and reliance on ai — an experimental study on the extent and costs of overreliance on ai. *Computers in Human Behavior*, 160:108352, 2024.

[10] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401v4*, May 2020.

[11] Zongxi Li, Zijian Wang, Weiming Wang, Kevin Hung, Haoran Xie, and Fu Lee Wang. Retrieval-augmented generation for educational application: A systematic survey. *Computers and Education: Artificial Intelligence*, 8:100417, 2025.

[12] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023.

[13] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert, 2019.

[14] Regeringskansliet. Arbetslösheten i sverige – så ser den ut. `https://www.regeringen.se/regeringens-politik/arbetsmarknad/arbetslosheten-i-sverige--sa-ser-den-ut/`, 2025. Accessed 1 December 2025.

[15] Alan Said. On explaining recommendations with large language models: a review. *Frontiers in Big Data*, 7, January 2025.

[16] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021.

[17] Xiaolei Wang, Xinyu Tang, Xin Zhao, Jingyuan Wang, and Ji-Rong Wen. Rethinking the evaluation for conversational recommendation in the era of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, page 10052–10065. Association for Computational Linguistics, 2023.

[18] Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K. Qiu, and Lili Qiu. Retrieval augmented generation (rag) and beyond: A comprehensive survey on how to make your llms use external data more wisely, 2024.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2026-04-10

EXAMENSARBETE Retrieval-Augmented Generation for Educational Exploration
STUDENT Ludvig Nyström
HANDLEDARE Peng Kuang (LTH)
EXAMINATOR Pierre Nugues (LTH)

Från gissningar till fakta: En chattbot som ger pålitliga studieråd

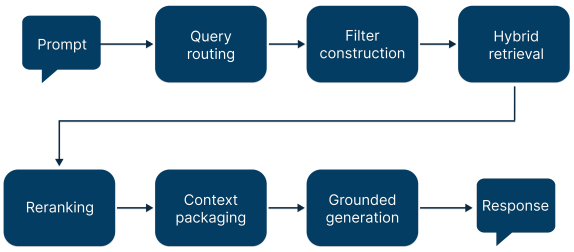
POPULÄRVETENSKAPLIG SAMMANFATTNING **Ludvig Nyström**

När en språkmodell svarar på frågor om utbildningar kan den låta övertygande även när den har fel – den kan blanda ihop behörighetskrav eller rekommendera program som inte existerar. Det här examensarbetet bygger om den underliggande arkitekturen så att AI:n hämtar verklig programdata innan den skriver sitt svar. För en elev som väljer utbildning utifrån chattbotens svar spelar det roll att informationen stämmer.

Sverige har över 3 500 utbildningsprogram fördelade på närmare 300 lärosäten, och studie- och yrkesvägledare på gymnasiet ansvarar ofta för flera hundra elever. Vanligtvis träffar en elev sin SYV endast en gång, resten av tiden sker sökandet på egen hand via webbplatser och antagningsportaler av varierande kvalitet.

I det ursprungliga systemet genererades chattbotens svar och programrekommendationerna oberoende av varandra. Vi byggde om arkitekturen så att systemet först söker i en databas med programdata och matar de mest relevanta träffarna till modellen innan den skriver sitt svar. Modellen baserar alltså sitt svar på verifierad data i stället för att förlita sig på vad den lärt sig under träningen. Tillvägagångssättet kallas Retrieval-Augmented Generation, eller RAG.

Skillnaden är tydlig. Det nya systemet eliminerade nästan helt fabricerade programreferenser: felaktiga omnämnanden minskade med 94 %. I en utvärdering där studie- och yrkesvägledare samt elever bedömde svar utan att veta vilket system som hade producerat dem föredrog samtliga den RAG-baserade varianten.



Något vi inte hade förväntat oss var att det nya systemet dessutom är snabbare, trots att det gör mer. Den gamla lösningen slog upp data vid flera tillfällen; den nya samlar allt i ett steg, vilket sänkte svarstiden med ungefär 10 %.

Poängen är inte att AI kan ersätta studie- och yrkesvägledare. Det kan den inte. Men när förklaringen och rekommendationerna kommer från samma källa kan användaren direkt se vad systemet baserade sitt svar på. Det förvandlar chattboten från en svart låda till något en skola rimligen kan lita på.