Proceedings of the Design Society, Volume 5: ICED25
https://doi.org/10.1017/pds.2025.10106

ChAx: a RAG-based chatbot for CAx education

Sarah Steininger [1,2], **Saltuk Kezer**[1], **Jona Rief** [1], **Emily Spicker** [1], **Sebastian Preis**[1] **and Johannes Fottner** [1]

[1] *Technical University of Munich, Germany,* [2] *BMW Group, Munich, Germany*

✉ sarah.steininger@tum.de

ABSTRACT: ChAx is a chatbot designed to support in technical drawing lectures by leveraging Retrieval-Augmented Generation. Addressing challenges such as the complexity of rules and dependencies in technical drawing, the system accesses the specific lecture materials to provide students with accurate and context-aware answers. The architecture combines modular components, including a RAG pipeline and a frontend with an interactive PDF viewer, ensuring transparency and user-friendliness. Optimization strategies like semantic chunking, fine-tuning, and cost-effective configurations enable efficient performance within constrained server environments. Evaluation metrics, including factual correctness and answer relevancy, were evaluated by using the LLM-as-a-judge method. The results underline ChAx's potential to enhance educational outcomes by enabling students utilize materials more effectively.

KEYWORDS: retrieval augmented generation, chatbot, education, computer aided design (CAD), user centred design

1. Introduction

The rapid development of Large Language Models (LLMs) and their widespread adoption is opening up new possibilities (Steininger, Camci, & Fottner, 2024). Especially in teaching, a core area of knowledge acquisition and dissemination, the use of this technology seems versatile (Fauzi, Tuhuteru, Sampe, Ausat, & Hatta, 2023; Jo, 2024; Kasneci et al., 2023). However, a major drawback of LLMs is the significant number of resources required to train them. This makes customisation a complex and resource-intensive task, especially when trying to teach existing LLMs new information, such as data from a custom dataset. Additionally, when using a small dataset, finetuning risks causing the model to overfit and forget crucial information. To face this challenge, Lewis et al. (2020) introduce a new approach known as Retrieval-Augmented Generation (RAG). In contrast to normal finetuning, in which the model is statically trained with new data, RAG dynamically accesses external knowledge sources. In this way, real-time updates and specific information can be obtained more efficiently, without having to adapt the model itself (Gao et al., 2023). The aspect of real-time updates is particularly important in teaching, as lectures are constantly evolving and updated with the state of the art. Used in this way, a chatbot enables efficient knowledge management by quickly providing specific information to the student (cf. Bassner, Frankford, and Krusche (2024)). The student can interact with a Chatbot, seek for clarification and engaging in in-depth discussions about their course material on his own (Jo, 2024). Regardless from time and place, a chatbot offers round-the-clock service and the student is no longer bound to the times of office hours (Kasneci et al., 2023). Finally, Jo (2024) also notes that the integration of LLMs makes teaching more attractive because of the novelty value, as it makes learning more exciting and engaging. Overall, the current research agrees that the integration of a chatbot mainly brings benefits. However, there are also risks associated with the new technology. For example, Kasneci et al. (2023) point out that language models may provide misleading or inaccurate information that is accepted by students without verification. LLM hallucination can never be eliminated but can only be minimized. In

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press

addition to the technological issues, the use of a chatbot in teaching raises aspects of psychology, sociology and sociolinguistics (Dibitonto, Leszczynska, Tazzi, & Medaglia, 2018).
In summary, integrating a chatbot raises several questions. How do I set up the architecture to ensure the best result? What settings and parameters do I use for RAG? How do I check the quality of the answers? And how can I design the frontend of the Chatbot to best meet the student's needs? To answer these questions, a RAG-based chatbot for Technical Drawing education is presented.

2. Materials and methods

2.1. Large Language Models

Over the last few years, LLMs have gained a lot of traction and can be found in various industries for multiple types of applications (Wang et al., 2024). LLMs are based on deep learning algorithms and artificial neural networks to understand, interpret, and generate answers in human language. Creating and training LLMs requires massive amounts of data (Hoffmann et al., 2022). Starting with an unsupervised learning phase, the model is given unstructured data to work through itself. When a sufficient knowledge base is created, the model is fine-tuned to create a consistent model and given structured data focused on more specific concepts. In the last step, the model goes through a phase of reinforcement learning. The LLMs answers are evaluated, and extra data is given where the answers are insufficient. A drawback of LLMs is that they require large amounts of data for their training, which makes it difficult to customize to a specific topic. The traceability of LLM answers can also cause problems. It is not always clear how the model derives its answer and from which dataset. This is especially critical in cases where the answer is hallucinated. One way to counteract these problems is by using Retrieval-Augmented Generation. (Gao et al., 2023)

2.2. Retrieval Augmented Generation

Retrieval-Augmented Generation was first introduced by, Lewis et al. (2020) and is a modification for LLMs that constricts the data reference to a specified set of information, usually from specific documents. This allows for domain-specific usage. RAG is built as a two-step approach, combining a non-parametric retriever with a parametric-memory generation model. Both retriever and generator are pre-trained and go through a fine-tuning approach. The retriever consists of a query encoder, which transforms the user's query into a vector representation, as well as a document index. The document index takes the vector query and uses a similarity matrix to identify and retrieve documents best suited for answering. To ensure that the similarity step is effective, the documents should be pre-embedded in the document index using the same query encoder used as part of the retriever. The generator then uses these documents and the user's query to create the final response. The documents retrieved from the index can be constrained to a pre-defined number k using the top-k approximation, where k is an adjustable hyperparameter that can be optimized for a more effective output generation.(Lewis et al., 2020)

2.3. Requirements analysis

In the outlined lecture, first and second semester students of various engineering courses learn the rules of technical drawing in twelve lectures and exercises. Each lecture focuses on one aspect of a technical drawing, explaining the relevant laws and many examples of their applications. Furthermore, a lecture textbook with more detailed descriptions of the rules and their applications, dependencies, and exceptions is provided. Despite these preparations, independently creating drawings proves to be a significant challenge for the students due to the many rules and dependencies between the standards. To improve students' success and motivation, the Chatbot must fulfil requirements. These requirements are divided into four sections: Requirements from both user groups, the learners and the teachers, technical requirements, and general requirements for using a chatbot in education.
From the learner's perspective, the chatbot must provide accurate and reliable information. This includes the chatbot generating answers using all documents provided in the lecture. The information provided by the chatbot must be understandable, tailored to the specific question and its context, and contain an appropriate level of detail. To ensure high service accessibility, it must be integrated into our learning management system (Moodle) and optimized for mobile and desktop devices.

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press

From the teachers' perspective, the chatbot should enable students to access and utilize the learning materials efficiently and purposefully to complete the assigned tasks, enhancing their learning outcomes. Furthermore, the chatbot system must allow for effortless updates to lecture materials, as the lecture materials are continuously improved. Additionally, the chosen implementation must be easily extensible while remaining resource-efficient and maintainable without requiring constant developer intervention.

The technical requirements are defined by the limitations of our in-house hosted servers, which lack GPUs and operate within constrained computational capacities, as well as the restricted monthly budget for this application. For the general requirements, it is essential to consider legal implications in cases where the chatbot provides incorrect or misleading information. Additionally, it is necessary to ensure compliance with privacy regulations and maintain transparency regarding data usage.

2.4. Evaluation metrics

Evaluating the quality of a chatbot pipeline is essential to ensure its success, analysability, and long-term sustainability. Chatbots can be assessed based on various criteria, such as the performance of retriever modules or their ability to generate factually accurate responses. Effective evaluation is also crucial for aligning with stakeholders, setting clear expectations, and communicating the chatbot's capabilities. Traditional metrics like ROUGE and BLEU, originally developed for machine translation tasks, remain widely used for evaluating chatbot outputs. These algorithms rely on n-grams to compare generated text against reference pairs, making them applicable for assessing RAG systems (Lin, 2004; Papineni, Roukos, Ward, & Zhu, 2002). A more modern approach to evaluation leverages large LLMs themselves in a method known as LLM-as-a-judge. Research has shown that LLMs can deliver human-like evaluation performance. Instead of relying solely on classical metrics, this approach employs LLMs to assess chatbot outputs against specific system components. Typically, a higher-quality LLM is used for evaluation, ensuring accuracy and reliability (Huang et al., 2024; Zheng et al., 2023). In practice, several frameworks and libraries implement the LLM-as-a-judge methodology. Notable tools include RAGAs, ARES, and comprehensive frameworks like DeepEval, which offer robust, well-established metrics to evaluate different aspects of chatbot performance. These metrics can, for example, assess the precision of retrieved documents relative to the generated responses or evaluate other pipeline components ("DeepEval,"; Es, James, Espinosa-Anke, & Schockaert, 2023; Saad-Falcon, Khattab, Potts, & Zaharia, 2023). For the evaluation, 300 high-quality question-answer pairs based on the lecture textbook were created. This dataset serves as a benchmark for assessing the pipeline's effectiveness. In total, 19 distinct tests were conducted, optimizing six separate criteria, and five evaluation metrics to measure performance across all 300 questions were employed. Each metric returned scores as percentages, allowing for systematic measurement and refinement of the pipeline's overall performance. More information regarding the chosen optimization criteria as well as the actual numerical results of the evaluation is listed in 3.2. Below, the metrics and their specific purposes are outlined. Further information regarding the selected metrics can be found in the paper of Es et al. (2023).

- **Factual Correctness** This metric evaluates the factual accuracy of generated responses by comparing them to baseline references. A confusion matrix is used to analyze claims in the generated answers against the expected responses.
- **Context Precision** Measures the proportion of relevant chunks within the retrieved context. Precision is calculated by comparing the number of relevant chunks at rank kkk to the total chunks retrieved at that rank.
- **Context Recall** Assesses the completeness of document retrieval by measuring how many relevant documents are successfully retrieved. Unlike precision, recall focuses on ensuring that few relevant documents are missed, reflecting retrieval completeness rather than relevance.
- **Faithfulness** Evaluates the factual consistency of the generated answer against the provided context. While Factual Correctness compares generated responses to a reference, Faithfulness ensures that claims in the response are grounded in the retrieved context and do not introduce unsupported information.
- **Answer Relevancy** Examines how closely the generated answer addresses the given prompt. This metric assesses whether the response is pertinent to the initial query, ensuring that key claims in the query are considered and correctly answered.

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press

3. Results

3.1. Architecture design

The architecture of the chatbot consists of several sub-modules, each of which is integral to the overall functionality of the system. An overview of all components is shown in figure 1.

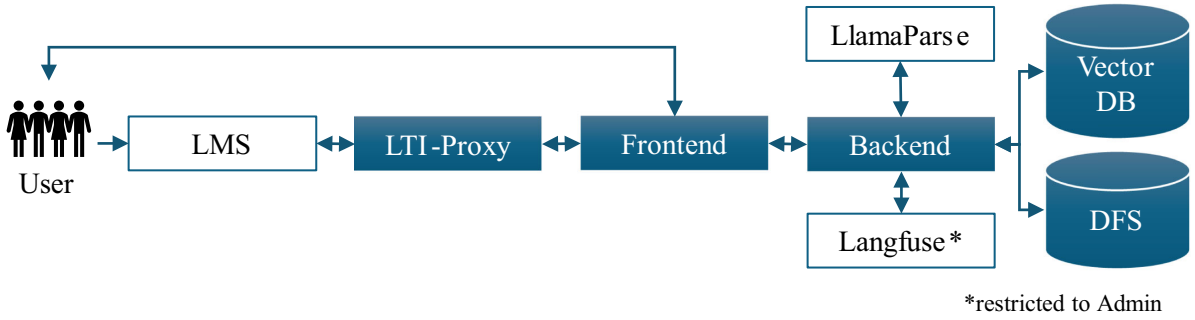


Figure 1. Architecture structure of the chatbot

Learning Management System (LMS) and LTI Proxy The user accesses the Chatbot through an LMS, for which different software is available. Within in the school of Engineering and Design (TUM) the LMS "Moodle" is used. In the architectural design, a key requirement was to secure access via the Learning Tools Interoperability (LTI) standard. It ensures that only students which are enrolled in the specific Moodle course can access the Chatbot. Given Moodle's native support for LTI integration, an LTI proxy was implemented to mediate between the frontend and backend of the application. The LTI proxy is built using Express.js, a web application framework for Node.js that simplifies API service creation and routing. The LTI integration uses the open source LTIJS library, which provides a simple implementation of the LTI 1.3 standard for secure authentication and communication. In this configuration, LTIJS is set up to handle connection attempts by redirecting users directly to the frontend. The security of the LTI standard is maintained through cookie-based authentication, so cookies are set prior to redirection to ensure secure access. To enforce security, middleware has been integrated into both the backend and frontend to enable communication with the LTI system. This middleware handles the "ltik" token added to the redirected URL by LTIJS and manages the authentication process. A single information endpoint within the Express.js application verifies the validity of incoming requests. Upon successful validation by the LTI proxy, a legitimate login can be confirmed. Requests that fail validation are rejected, securing access to the system and ensuring that only authenticated users can interact with the application. The frontend, backend, and LTI-proxy are each implemented as separate applications, all of which are stored in the TUM GitLab repository.

Frontend The Frontend is designed to manage user interactions, including the display of the chatbot interface, handling user messages, and providing a PDF viewer for displaying sourced documents and highlights. The frontend of the application is implemented using Next.js, a robust React framework designed for building full-stack applications. Next.js extends the core capabilities of React by adding features that streamline development and maintenance. These features include server-side rendering, static page generation, and built-in routing, which together improve performance, scalability, and the overall developer experience. By using Next.js, the application is designed to be responsive, efficient, maintainable, and extensible, allowing for future enhancements with minimal disruption (Vercel Inc.). Upon successful authentication through the LTI proxy, the necessary web files to display the frontend are delivered to the user, allowing them to interact with the Chatbot. The user interface design principles prioritize simplicity and ease of use, deliberately avoiding unnecessary complexity. Starting with a template from create-llama, a service provided by LlamaIndex for rapid development of RAG applications, the template was customized to meet specific requirements. Key enhancements include a custom welcome message to greet users upon entering the site. In addition, the Moodle Web Services API is used to retrieve user-specific information such as the user's name and profile picture, creating a more personalized experience. Through prompt engineering, the model is configured to appropriately cite all generated content. These citations are parsed in the front end and displayed as interactive sources. When a user clicks on a citation, a PDF viewer is triggered to display the referenced source document directly on the page (see figure 2).

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press

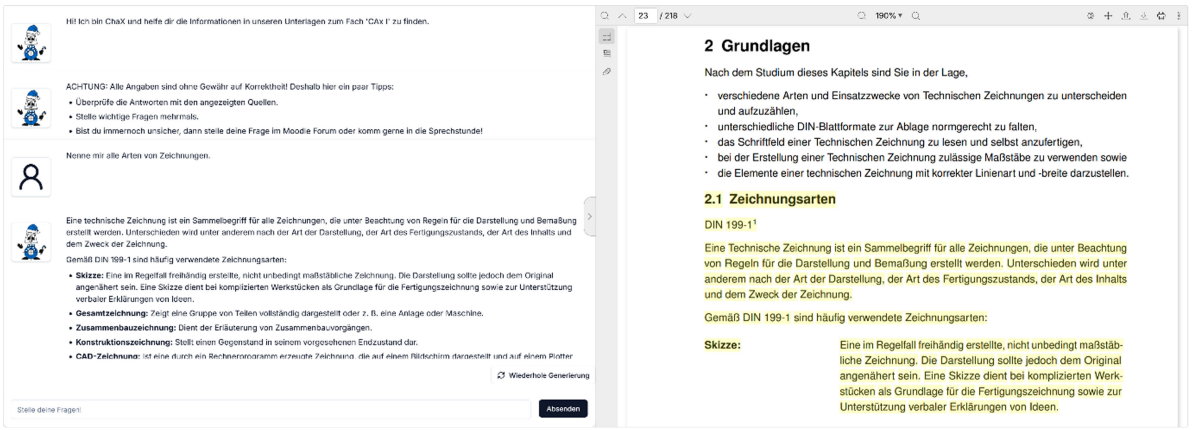


Figure 2. Frontend

Backend The backend functionality is exposed via a consumable HTTP-based API, which handles the serving of PDF documents and related annotation. It is based on FastAPI, a Python framework for creating APIs, which allows easy integration with LlamaIndex ("FastAPI"). The application provides three main routes:

1. /api/chat: this route handles HTTP POST requests and accepts a list of messages. The messages are forwarded to the active ChatEngine and the replies are sent to the frontend in real time.
2. /api/pdf: This HTTP GET resource can be used to retrieve a list of available PDF files in the system. The PDF files must first be generated for a response.
3. /api/pdf/highlight-text: This route allows highlighting to be added to PDFs. By comparing text parts of the PDF page with the original text, the best match is found and the corresponding coordinates for the highlight are returned so that the frontend can display the highlight.

LlamaParse To increase the efficiency of processing course materials, a parser is added. A parser is a program that analyses data and converts it into a structure that can be easily processed by a computer program. For example, it intelligently converts PDFs into markdown format, optimizing the documents for later use within the application. In the architecture, the lecture materials are stored in the Document Filing System and retrieved via its API to the server, where they are then sent to the parser for further processing.

Langfuse Another component of the architecture is the integration of Langfuse, an open-source platform designed to build and manage large language models. Langfuse allows for comprehensive tracking of user interactions with the system, supporting live evaluation and the collection of metrics and data to improve the pipeline. Langfuse was selected for its seamless integration with LlamaIndex. In addition to tracking, Langfuse supports debugging by providing detailed insight into the time spent on each subcomponent of the Retrieval-Augmented Generation (RAG) pipeline. It also tracks the costs associated with all generated messages, providing valuable information for cost management and optimization.

Document Filing System (DFS) and Vector Data Base (DB) In the DFS the lecture materials, which serve as input for the RAG pipeline, are stored. The stored files are primarily PDFs, which include both images and raw text. However, due to the limitations of the free-tier LlamaParse service, which only supports textual data processing, the current model does not handle multi-modal data, leaving images unanalysed. When executing the document embedding process, the script automatically retrieves the PDFs by entering the URL address and the access data. Once LlamaParse completes the parsing, it returns a markdown version of the document, which is then embedded and stored in the vector database. This process ensures that the documents are accurately parsed, embedded, and ready for efficient retrieval in subsequent queries.

To summarize the overall architecture, the user message is received by the frontend where an embedding model processes the message and compares it to documents stored in the vector database. The relevant documents retrieved from this comparison are provided to the LlamaIndex agents as contextual information, allowing them to generate a response. This response is then returned to the frontend, completing the interaction cycle.

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press

3.2. Evaluation and optimization

The fundamental concept of RAG is introduced in Section 0 While a basic implementation of RAG provides reasonable performance, various techniques can significantly enhance the effectiveness of both the chatbot and the pipeline. To improve system quality, several optimization strategies, focusing on five key areas, were applied.

Retrieving As discussed in Section 0, the retriever employs the parameter k to determine the number of documents passed as context to the generator module. Ideally, providing the entire context would maximize information retrieval, but resource constraints and the linear cost increase of generator modules with context size necessitate a balanced approach. In this project, multiple values for k (600, 200, 300, 100, 50, 20, 10, and 5) were tested to identify the optimal trade-off between performance and resource cost. After theoretical and practical evaluations, the most effective value was selected, ensuring a balance between retrieval quality and system efficiency.

Reranking Advanced RAG pipelines benefit from a dedicated reranking module to refine retrieved documents. While the retriever relies on the highly efficient Maximum Inner Product Search (MIPS) algorithm, its non-parametric nature can yield suboptimal results, particularly when k is small. To address this, a reranker module was implemented that improves ranking accuracy at the expense of additional computational resources. The reranker refines the top k retrieved documents into a final set of n documents embedded in the context. The hyperparameter n controls the size of this refined set, allowing to optimize for performance while managing computational overhead (Zhuang, Liu, Koopman, & Zuccon, 2023).

Advanced Querying-Techniques Further querying techniques, specifically subquestion querying, were analysed during the finetuning process. This method involves decomposing the initial query into claims or subquestions, which are individually processed by the retriever and generator modules. The subquestion responses are then synthesized into a final response, enabling the chatbot to better handle complex queries (Gao et al., 2023). However, this approach effectively multiplies computational costs by the number of subquestions generated. While testing revealed a modest performance improvement of 3–4%, the cost increase outweighed the benefits. Consequently, the technique wasn't adopted to the system.

Finetuning Finetuning a RAG-based system can significantly enhance performance by customizing the model to meet specific domain requirements. Leveraging OpenAI's public finetuning API, we trained a custom model using a high-quality dataset comprising 300 questions, employing a 70/30 train-test split for training and evaluation. The fine-tuned model exhibited noticeable performance improvements while incurring minimal additional costs. We selected a fine-tuned version of GPT-4o-mini for its combination of fast inference times, strong performance, and relatively low cost compared to other, more expensive models. Although fine-tuning introduces certain risks, as outlined in Section 1, combining it with a RAG framework mitigates these concerns. This hybrid approach enables the model to develop a deeper understanding of domain-specific language and enhances its ability to effectively handle specialized terminology. The use of synthetic datasets to augment the training process was also explored. However, aligning the data distributions of high-quality manually collected datasets with synthetic data proved to be a complex challenge. Future research may uncover improved strategies for achieving better alignment between these dataset types, opening the door for further performance gains.

Chunk Sizes When working with LLMs, chunking is a crucial process that involves dividing a document into smaller, manageable parts. The simplest approach to chunking is splitting text into fixed-size chunks containing a specific number of tokens. However, since most LLMs have a constrained context window —limiting the amount of additional information passed alongside a query—this naive approach can be suboptimal. Moreover, even without such restrictions, providing full documents as context can overwhelm the model, making it difficult to localize relevant information in a large text. Chunking helps mitigate these issues by breaking the document into smaller pieces, enabling finer-grained control over context. Various chunking strategies, starting with sentence-based splitting, were explored in the project. While this approach provided a straightforward solution, it revealed limitations, such as difficulty in maintaining the contextual flow of the document's structure. To address this, semantic chunking was considered. It involves leveraging an embedding model to create vector representations of individual sentences. These embeddings are compared using a sliding window approach, where adjacent sentences are analyzed for similarity. Sentences with high similarity are grouped into the same chunk, while

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press

significant dissimilarities indicate a potential topic shift, such as the beginning of a new chapter or section. This method led to noticeable improvements in the quality of chunks, as it preserved semantic coherence (Gao et al., 2023). Despite these improvements, a persistent issue was encountered: loss of contextual information about the document's structure, particularly its chapters and sections. Human readers naturally rely on outlines to interpret content. For instance, if a cookbook discusses general slicing techniques under a subchapter about bananas but doesn't explicitly mention bananas within the subsection, readers can still infer that the techniques are related to bananas. To replicate this structural awareness, markdown-based chunking method was developed. The implemented markdown-parsing is inspired by LlamaIndex' MarkdownParser (LlamaIndex). In the pipelines first ingestion step, PDFs are converted to markdown, enabling to analyze the document's headings, sections, and subsections. Each subsection is compiled into a single node, with headings serving as markers for starting new chunks. If a subsection is exceptionally long, spanning multiple pages, a fixed-size chunking step was applied to divide it into smaller parts. Additional metadata, such as the ordering of these smaller chunks within the original subsection, is added to maintain structural clarity. The chunk size for these fixed-size divisions is a tuneable parameter, optimized for specific use cases. The effectiveness of this approach is demonstrated in the evaluation table 1, which highlights its role in improving information retrieval and maintaining contextual relevance.

You are ChaX, the primary teaching assistant for the 'CAx I' course at the Technical University of Munich (TUM).
Your role is to provide students with clear, concise, and accurate answers primarily in German.
The 'CAx I' course teaches students to:
- Create technical drawings that adhere to manufactruing standards.
- Use CAD systems practically.
- Follow basic design rules for component construction.
This module is foundational for advanced design courses, such as machine elements in the 3rd and 4th semesters.

Guidelines:

- Language: Answer in German. Expand English abbreviations into their appropriate language.
- Knowledge Source: Only use information directly from the internal knowledge base. Do not infer or generate answers for missing information.
- Response Format: Format all responses in markdown.
- Clarity: Keep responses clear and concise, providing additional context only if essential.
- Citations: Provide citations for all statements deemed worthy of reference. Avoid being selective or sparing—cite as thoroughly as possible using ^[citation] format immediately after each cited point.
- Citations Rejection: If the answer does not contain a citation, reject the answer, by selecting a response category as described below.
- Topic Relevance: Handle questions outside the knowledge base as outlined below.
- Automated Decision-making: The chatbot must independently determine the appropriate response category (e.g., Not in Knowledge Base, Irrelevant Topic, etc.) while adhering to rejection policies when applicable.

Figure 3. Initial prompt

Initial Prompt A key factor influencing system performance is also the selection of the initial prompt. For this issue, the approach outlined in Bassner et al. (2024) was taken as guidance. Additionally, the concept of few-shot learning was incorporated to teach the bot how to identify and reject certain types of questions (Brown et al., 2020). The final version of the prompt can be seen in figure 3.

As shown in table 1, the best results show a 75% correctness for our primary script and approximately 70% correctness when both main scripts are combined. The slightly lower correctness for the combined scripts is attributed to the second script, which includes a significant amount of visual image data. As the current iteration of the chatbot primarily focuses on text analysis, its correctness decreases when handling visual data. The row highlighted in bold represents the configuration currently deployed in the production environment. Performing reranking via an API is extremely expensive, with costs reaching up to three times higher than those for query analysis and response generation combined. An alternative approach would be to host a GPU locally and run an open source reranking model in-house. For evaluation purposes, the Cohere API reranker was used, which is fast, scalable, and delivers excellent performance ("Cohere").

Due to cost constraints and the absence of a reranker, it was necessary to set k=5 for the number of embedded documents. While this decision significantly reduces pipeline costs, it also adversely impacts

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press

Table 1. Test settings and evaluation results.

K	N	Sub-question	Chunk Size	Fine-tuning	TZ Skript	GEO Skript	Factual Correctness	Answer Relevancy	Faith-fulness	Context Recall	Context Precision
600	10	X	256	X	✓	X	67,16%	86,53%	88,32%	90,25%	86,53%
200	10	X	256	X	✓	X	70,35%	88,3%	87,76%	89,27%	85,45%
200	X	X	256	X	✓	X	68,61%	88,16%	87,83%	92,27%	78,69%
300	5	X	512	X	✓		71,66%	87,32%	87,44%	93,92%	94,13%
300	X	X	512	X	✓	X	69,1%	87,19%	87,28%	92,1%	82,13%
100	5	X	512	X	✓	X	71,79%	88,21%	87,27%	93,35%	93,8%
100	X	X	512	X	✓	X	67,28%	88,21%	88,12%	92,43%	82,93%
100	5	✓	512	X	✓	X	72,63%	87,91%	89,13%	93,33%	81,63%
100	X	X	1024	X	✓	X	69,62%	88,87%	86,16%	88,77%	84,99%
100	5	X	1024	X	✓	X	70,26%	88,72%	87,31%	95,3%	94,47%
5	**X**	**X**	**512**	**X**	✓	**X**	**65,05%**	**87,56%**	**87,02%**	**86,78%**	**87,7%**
10	X	X	512	X	✓	X	66,43%	88,48%	87,48%	91,85%	83,06%
20	X	X	512	X	✓	X	66,49%	86,4%	87,79%	91,68%	83,27%
50	X	X	512	X	✓	X	68,59%	88,14%	87,82%	91,56%	83,06%
100	X	X	512	✓	✓	X	70,29%	90,12%	82,93%	92,78%	83,81%
100	5	X	512	✓	✓	X	74,69%	88,6%	89,92%	94,72%	96,62%
100	5	X	512	✓	✓	✓	69,49%	88,44%	91,62%	94,57%	93,71%

performance. It is important to note that costs scale linearly with the number of documents. Without a reranker to minimize the number of documents passed to the generator module, increasing from 5 to 100 documents would result in a 20x cost increase. Thus, the row highlighted in bold represents the current evaluation metrics for the chatbot.

4. Discussion

4.1. Evaluation of requirements

Requirements from the Learners' Perspective The implemented pipeline of the chatbot uses RAG to access the provided lecture materials. In this way, it ensures that the relevant documents are retrieved, and accurate answers are delivered. A PDF-Viewer placed next to the chat window displays referenced documents, which provides transparency and contextual precision. The integration with Moodle is successfully implemented, with support for both mobile and desktop devices. A limitation is the lack of image integration. In the current pipeline, images in lecture materials are not analysed due to the text-only processing capability, which could be an issue for visually oriented content. To some extent the PDF-Viewer could be a pain reliever, assuming that the image content is referred in the text. Another solution for this issue is the textual description of images in the parser process. In the scope of this project, this option wasn't added due to additional costs.

Requirements from the Teachers' Perspective Regarding the requirements of education staff, the chatbot helps students to use the materials effectively. The question on the potential to improve the learning outcome of the students will be evaluated in the current test phase. Updates to materials are simplified through a script which provides automated synchronization with the DFS. A drawback of the pipeline is the dependency on open-source tools like Llamaindex or Langfuse. If they are changes, the pipeline must be updated.

Technical Requirements Several optimizations and restrictions are made to reduce the computational demands and the related costs. For example, currently no reranker modul is implemented into the pipeline although it would offer more performance. Overall, the factors "(resource) costs" and "performance" are conflicting requirements throughout the whole project. To anyway fulfil the requirement, cost-conscious decisions have been made.

General Requirements Data protection is ensured by using local servers and secure LTI connections. Errors are minimized through optimized RAG modules and benchmark dataset of high-quality question-answer pairs. Nevertheless, avoiding misleading answers (hallucinations) remains a challenge, particularly for complex or niche queries.

Overall, the implementation of the chatbot fulfils most of the outlined requirements effectively. Key success includes LMS integration, RAG implementation, and resource optimization. However, limitations exist regarding the inclusion of graphical materials and the long-term balance between maintenance and cost.

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press

4.2. Reflection of the approach

The system's performance is highly dependent on numerous parameters and optimization techniques, many of which are resource intensive. For example, advanced solutions such as Cohere offer potential improvements but are approximately ten times more expensive, making them impractical for large-scale educational use. This emphasizes the need for a balanced trade-off between cost and utility.

Choice of Evaluation Metrics The choice of the evaluation method "LLM-as-a-judge" offers a great opportunity to assess the performance of the chatbot. Nevertheless, optimizing on the six selected criteria risks that the parameters are specialized on these. Considering that the evaluation LLM is usually a higher-quality one, this limitation is made due to budget requirements.

Response Time In total, the whole pipeline and the integrated modules require an average response time of 12 seconds. It is adequate for educational applications but leaves room for improvement. Faster response times could enhance the user experience, especially during high-intensity usage periods.

Integration of Visual Data As mentioned in chapter 4.1 there's no integration of visual data in the chunks. Regarding questions to the content of an image, the results have been suboptimal. Incorporating better methods for image integration, such as multi-modal processing, could enhance the chatbot's effectiveness, particularly for technical subjects.

Novelty of RAG Technology Furthermore, the technology RAG is relatively new, being introduced only in 2020. While it has proven effective, its recent origin means there is still much to learn and optimize, especially in educational contexts.

Impact on Learning Habits As Bassner et al. (2024) highlight, over-reliance on chatbots can affect learning behaviors, potentially reducing critical thinking and deep engagement with material. There is a risk that students may prefer quick answers over tackling complex concepts. However, it is worth noting that students are aware that neither ChAx nor the lecture script can be used during exams, which helps to mitigate this concern.

5. Conclusion and outlook

The implementation of ChAx, a RAG-based chatbot, demonstrates the potential of enhancing the learning experience in technical drawing lectures. By providing accurate and context-aware answers, the chatbot addresses the challenges students face in mastering complex rules and dependencies. The architecture effectively balances performance with limited server resources through optimizations like semantic chunking and fine-tuning. While ChAx meets most of the defined requirements, certain limitations remain, including the absence of image data processing. This challenge presents an opportunity for future development. Additionally, the chatbot entered the test phase and the first feedback loops with an initial beta version in 2024. Now, the chatbot will be put into everyday operation at the start of the 2025 summer semester. It is now imperative to generate empirical data to analyse and compare student learning before and after the intervention. It is also necessary to analyse how and when students use the chatbot exactly. Moreover, the need for systematic feedback loops with students is critical for continuous improvement and alignment with educational goals.

Overall, ChAx provides a robust foundation for integrating AI-driven assistance into education, paving the way for broader adoption and refinement of such technologies in both academic and industrial settings. Future efforts will aim to enhance the system's capabilities while maintaining a cost-effective and user-centric approach.

References

Bassner, P., Frankford, E., & Krusche, S. (2024). Iris: An AI-Driven Virtual Tutor for Computer Science Education. In M. Monga, V. Lonati, E. Barendsen, J. Sheard, & J. Paterson (Eds.), *Proceedings of the 2024 on Innovation and Technology in Computer Science Education* V. 1 (pp. 394–400). New York, NY, USA: ACM. https://doi.org/10.1145/3649217.3653543

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P.,... Amodei, D. (2020). *Language Models are Few-Shot Learners*. Retrieved from http://arxiv.org/pdf/2005.14165v4

Cohere. Retrieved from https://cohere.com/

DeepEval. Dibitonto, M., Leszczynska, K., Tazzi, F., & Medaglia, C. M. (2018). Chatbot in a Campus Environment: Design of LiSA, a Virtual Assistant to Help Students in Their University Life. In M. Kurosu (Ed.), *Lecture Notes in Computer Science. Human-Computer Interaction. Interaction Technologies: 20th International Conference, HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings* (1st

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press

ed., Vol. 10903, pp. 103–116). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-91250-9_9
Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. Retrieved from http://arxiv.org/pdf/2309.15217v1
FastAPI. Retrieved from https://fastapi.tiangolo.com/
Fauzi, F., Tuhuteru, L., Sampe, F., Ausat, A. M. A., & Hatta, H. R. (2023). Analysing the Role of ChatGPT in Improving Student Productivity in Higher Education. *Journal on Education*, 5 (4), 14886–14891. https://doi.org/10.31004/joe.v5i4.2563
Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y.,... Wang, H. (2023). *Retrieval-Augmented Generation for Large Language Models: A Survey*. Retrieved from http://arxiv.org/pdf/2312.10997
Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E.,... Sifre, L. (2022, March 29). *Training Compute-Optimal Large Language Models*. Retrieved from http://arxiv.org/pdf/2203.15556v1
Huang, H., Qu, Y., Bu, X., Zhou, H., Liu, J., Yang, M.,... Zhao, T. (2024). *An Empirical Study of LLM-as-a-Judge for LLM Evaluation: Fine-tuned Judge Model is not a General Substitute for GPT-4*. Retrieved from http://arxiv.org/pdf/2403.02839v3
Jo, H. (2024). From concerns to benefits: a comprehensive study of ChatGPT usage in education. *International Journal of Educational Technology in Higher Education*, 21 (1). https://doi.org/10.1186/s41239-024-00471-4
Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F.,... Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, 102274. https://doi.org/10.1016/j.lindif.2023.102274
Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N.,... Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. Retrieved from http://arxiv.org/pdf/2005.11401
Lin, C.-Y. (2004). ROUGE: a Package for Automatic Evaluation of Summaries. In *Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004*, Barcelona, Spain. Retrieved from https://www.microsoft.com/en-us/research/publication/rouge-a-package-for-automatic-evaluation-of-summaries/
LlamaIndex. Markdown. Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. In *Annual Meeting of the Association for Computational Linguistics*. Retrieved from https://api.semanticscholar.org/CorpusID:11080756
Saad-Falcon, J., Khattab, O., Potts, C., & Zaharia, M. (2023). *ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems*. Retrieved from http://arxiv.org/pdf/2311.09476v2
Steininger, S., Camci, H., & Fottner, J. (2024). Current State, Potentials and Challenges for the Use of Artificial Intelligence in the early Phase of Product Development: A Survey. In *2024 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 551–555). IEEE. https://doi.org/10.1109/IEEM62345.2024.10857061
Vercel Inc. Next.js. Retrieved from https://nextjs.org/docs
Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J.,... Wen, J. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18 (6). https://doi.org/10.1007/s11704-024-40231-1
Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S. [Siyuan], Wu, Z., Zhuang, Y.,... Stoica, I. (2023). *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. Retrieved from http://arxiv.org/pdf/2306.05685v4
Zhuang, S. [Shengyao], Liu, B., Koopman, B., & Zuccon, G. (2023). *Open-source Large Language Models are Strong Zero-shot Query Likelihood Models for Document Ranking*. Retrieved from http://arxiv.org/pdf/2310.13243v1

https://doi.org/10.1017/pds.2025.10106 Published online by Cambridge University Press