Visualizing RAG Chatbots in Education

Raymond Liu and Kevin Wang

Abstract—

1 Introduction

In recent years, instructors have increasingly adopted large language models (LLMs) to assist in teaching, grading, and answering student questions. Retrieval-Augmented Generation (RAG) systems such as ChatEd [14] and Jill Watson [10] represent a promising step forward: they allow educators to integrate their own course materials into LLMs, improving answer accuracy and contextual alignment without requiring advanced programming or machine-learning expertise. For under-resourced teaching teams, these systems offer a scalable and cost-effective way to provide round-the-clock learning support for students.

However, instructors often lack visibility into how these systems operate internally. Most are non-experts in LLM architecture or prompt design, and current interfaces provide little guidance on how retrieval parameters or prompt changes affect the model's behavior. As a result, instructors struggle to understand why a chatbot produces certain answers—or why it fails to reference the correct course documents. This lack of transparency limits instructors' ability to diagnose errors, adapt prompts, and iteratively improve their course bots.

Opaque retrieval logic. Instructors cannot easily see which documents influence an answer, nor how the model balances retrieved content with general background knowledge.

Iteration blind spots. When instructors modify system prompts (e.g., "be concise," "simplify for beginners"), they must manually query the chatbot to observe changes. This hinders rapid iterations and improvements.

Limited explainability. Existing RAG interfaces show only raw document lists or similarity scores, providing little sense of how retrieved content influences the generated text. There are no intuitive mechanisms for tracing document–sentence relationships or for visualizing the impact of parameter adjustments.

To address these challenges, we propose a visual analytics suite that makes RAG behavior interpretable and actionable for instructors. Our goal is to transform opaque retrieval pipelines into transparent, manipulable visual representations that support diagnosis, comparison, and refinement of educational chatbots.

Our proposed tool adopts a two-tier design: an overview view summarizing student-chatbot interactions across the course, and a detail view that juxtaposes responses under different configurations (A/B comparison). Instructors can identify low-quality answers, inspect how document retrieval and generation differ between model settings, and receive actionable nudges—such as suggesting a new course document upload or adjusting top K values.

By combining RAG explainability with instructor-centered visualization design, this project aims to bridge the gap between powerful LLM infrastructure and the practical needs of educators.

1.1 Personal Expertise

Kevin has experinces in designing and evaluating AI systems for education, with focus on Retrieval-Augmented Generation (RAG), instructor tooling, and explainable human–AI interfaces. He was the lead developer of the HelpMe platform [12], an AI-enhanced help-seeking system deployed across multiple departments at the University of British Columbia (UBC). HelpMe integrates human and AI assistance into a unified interface, enabling instructors to customize RAG configurations (e.g., chunking, top-k, and retrieval thresholds), maintain course-specific knowledge bases, and monitor student queries in real time.

In promoting and maintaining these systems, Kevin has conducted extensive interviews and informal evaluations with instructors, uncovering recurring problems that motivate this visualization project:

- Instructors struggle to understand how the chatbot retrieves certain materials, and which documents most influence its answers.
- Almost all lack intuition for adjusting RAG parameters (top K, retrieval thresholds, or model choice).
- Instructors want to understand why some questions are answered wrong (the false positive rate is essential)

Raymond has also worked with RAG systems and has conducted research on evaluating LLMs.

2 Related Work

2.1 Large Language Models (LLMs) and Education

Large Language Models (LLMs) have evolved rapidly, showing promise across numerous domains [3, 4]. However, general-purpose LLMs often lack the contextual awareness and pedagogical appropriateness crucial for diverse classroom contexts. Agentic frameworks further extend these capabilities through techniques like Chain-of-Thought (CoT) prompting, enabling LLMs to handle multi-step tasks by making their intermediate reasoning explicit [11, 16].

However, a fundamental challenge arises when applying LLMs in education: their training data is broad but not course-specific. Most LLMs rely on large-scale, publicly available datasets that may not include domain-specific, up-to-date, or institutional knowledge [3, 8]. This limitation is particularly problematic in higher education, where course specifics vary across institutions, instructors, and student needs [5].

2.2 Educational AI Assistant

LLM and Retrieval-Augmented Generation (RAG) methods address this gap by combining LLM capabilities with targeted knowledge retrieval [7], enhancing both precision and relevance of AI-generated responses in education. The later iterations, especially in-prompt RAG systems, increase contextual awareness of AI-generated responses and adaptability as updating knowledge sources is much easier. A survey [6] shows that lack of training data is the biggest problem in developing a chatbot in education. Most AI assistant systems primarily rely on pre-existing instructional content, such as course materials and structured knowledge bases. Several LLM-based educational tools have been developed to assist students and instructors in courses with minimal overhead. Examples include Jill Watson [5], an AI-powered teaching assistant designed to answer student queries in online courses, and a chatbot-based system for assisting students with academic inquiries [14]. Prior work on integrating student help systems and AI chatbots [2] shows promise, although there is a need to evaluate AI effectiveness in question answering and supporting continual content updates. While effective for answering frequently asked questions, these

- *Raymond Liu is with the University of British Columbia. Email: raymliu@cs.ubc.ca*
- *Kevin Wang is with the University of British Columbia. Email: kevinsk@student.ubc.ca*

approaches miss a critical aspect of education: real-time, student-driven interactions. Students often ask novel, context-dependent questions that are not explicitly covered in course materials.

2.3 Visualization for RAG

New research on visual analytics for LLM ecosystems has begun to address the transparency and interpretability challenges inherent in retrieval-augmented generation (RAG) systems. While much of the early work focuses on prompt engineering or plain LLMs, a small but growing body of work explicitly targets the retrieval + generation pipeline.

One of the earliest efforts in visualization for prompt-driven LLM workflows is the work by Strobelt et al. [9] which presents a system for interactive and visual prompt engineering. Their approach enables on-the-fly parameter tuning, side-by-side outcome comparisons, and rapid visual feedback, though it does not incorporate an explicit retrieval component.

Extending into the retrieval space, Wang et al. [15] introduce RAGVIZ, a visual system designed to help users diagnose how retrieved passages contribute to generated answers. Their interface visualizes token and document-level attentiveness, supports toggling context inclusion, and assists debugging of hallucinations and retrieval failures. More recently, Wang et al. [13] propose XGraphRAG, which leverages graph-based representations of retrieval for specifically graph knowledge retrievals. Arawjo et al. [1] present ChainForge, a visual toolkit for prompt engineering and hypothesis testing with LLMs. Though not explicitly RAG-oriented, ChainForge's comparative small-multiple views and experiment workflows suggest strong design patterns for comparing variants (e.g., prompt v1 vs prompt v2), which we may adapt and extend in our work.

Together, these systems illustrate several recurring design idioms: side-by-side comparisons, provenance tracing, visual attention/coverage indicators, and multi-view pipelines. And that informs our instructor-facing "cockpit" design. Our work differentiates itself by focusing on: 1. course-specific corpora, 2. student-question histories, 3. and by embedding actionable nudges for instructors (such as "Considering add new document about x; decrease top k to 3"). By and large, we aim to bridge the gap between general RAG diagnostics tools and the unique operational needs of instructors who deploy chatbots in programming and theory-heavy courses.

3 Data and Task Abstraction

3.1 Data Abstraction

On the highest level, our data consists of parameters within the educational RAG chatbot system, as well as student questions, chatbot answers, and the documents retrieved by the RAG.

More specifically, the first table consists of the chatbot configurations, which are adjustable by instructors. This includes:

- Base LLM: The base LLM used for the chatbot (for example GPT4o, qwen, gemma, etc.), abstracted as a categorical variable with approximately 5 options available to the instructor.
- Course documents; the documents uploaded by instructors to be retrieved by the RAG system, abstracted as a set of categorical variables (free-text documents).
  - Note that these documents are chunked (split into sub-documents) by the system. We can consider the chunking locations within the documents (i.e. start positions, end positions) as ordinal variables.
- Top K: the number of top documents retrieved at each query, abstracted as a categorical variable (with options 3, 5, and 10 available to the instructor)
- Retrieval threshold: the minimum similarity score a document should have to be considered in the RAG ranking, abstracted as a categorical variable (with 100 options, ranging from 0.01 to 1.00, available to the instructor)
- System prompt: the text used in every prompt providing (for example, "Please answer the following student question using the provided documents."). This is abstracted as a free-text categorical variable.

We plan to focus our visualization on the base LLM and course documents, as we believe instructors will be more inclined to alter those parameters, compared to the remaining three options.

The next table documents the usage of the chatbot in a course. This contains the questions asked by students and the answers provided by the system, given a specific set of model parameters. More specifically, the attributes are as follows:

- Student question text: The question text given by the student, abstracted as a free-text categorical variable.
- Chatbot-generated answer: The response generated as a chatbot, abstracted as a free-text categorical variable.
  - Optionally: we can also consider displaying reasoning traces for reasoning models, which is another free-text categorical variable.
- Retrieved document chunks: The list of document chunks used by the chatbot to generate the answer to the student question, abstracted as a set of categorical variables.

  We can also consider data derived from this:

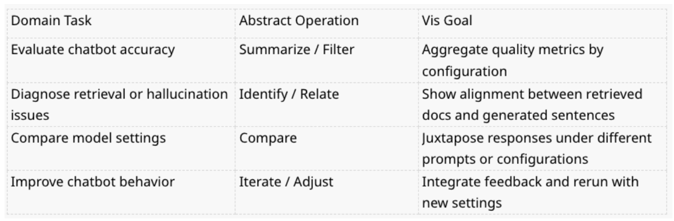
  - Document similarities: the semantic similarity of each document with the question, as determined by the RAG system. This is abstracted as a list of ordinal variables, in which the ordering of the list matches the ordering of the document chunks.
  - Relevant substring positions: The positions (start, end) of the relevant substring(s) within the document chunks used by the chatbot to generate the answer. This can be obtained using an NLP-based similarity metric after string splitting the document; it can be abstracted as a set of ordinal variables.
- Chatbot configurations: The chatbot settings that were used to generated the answer. This consists of all the information in the first table of data listed above.
- Chatbot answer quality: Some measure of the quality of the chatbot answer. We consider two possible metrics:
  - Human feedback: the system provides mechanisms for students and instructors to evaluate the quality of the chatbot-generated answer. This would most likely be a binary evaluation in approximately three categories (helpfulness, correctness, relevance). This can be abstracted as a set of categorical variables.
  - NLP-generated metrics: computational evaluations of the chatbot-generated answer can be derived using methods rooted in NLP. One potential method is semantic similarity with ground truth; another is using an LLM-as-a-judge approach, using a different LLM. Both systems would produce an ordinal variable (ranging from 0.00 to 1.00), representing a numerical score measuring the quality of the chatbot answer.

Note that our tool is designed to work with datasets of different cardinalities, ranging from 1 student question/chatbot answer to over 100. However, for the purposes of testing, we will work with a custom dataset with approximately 25 student questions/chatbot answers.

3.2 Task abstraction

The instructors' goals in using our system are to:

1. evaluate the overall ability of the chatbot to answer student questions, in terms of correctness;



Domain Task	Abstract Operation	Vis Goal
Evaluate chatbot accuracy	Summarize / Filter	Aggregate quality metrics by configuration
Diagnose retrieval or hallucination issues	Identify / Relate	Show alignment between retrieved docs and generated sentences
Compare model settings	Compare	Juxtapose responses under different prompts or configurations
Improve chatbot behavior	Iterate / Adjust	Integrate feedback and rerun with new settings

Fig. 1: Task abstraction summary

2. identify and diagnose the chatbot's problems, specifically questions that were answered incorrectly or unhelpfully; and,
3. fix the problems, by adjusting model settings or uploaded course materials and evaluating the new chatbot on these problematic questions.

Due to time and resource constraints, we will most likely focus on a subset of these tasks for the project. A summary of task abstraction is in Fig. 1.

4 Proposed Solution

The system's general design is based on two visualization displays, which roughly breaks down into an overview page and a detail page. The overview page will display a full history of chatbot questions and answers for overall evaluation and identification of poorly answered questions; the detail page will juxtapose the chatbot answers for different model configurations, allowing instructors to find their preferred configuration for these poorly answered questions.

To provide an scenario, imagine a UBC CPSC447 instructor is using this chatbot in their course. They've decided to use the default model configurations, and they've uploaded all the documents pertaining to the course, including the course website (as an HTML file). It is now one month into the course, and they are receiving complaints in office hours about the chatbot providing incorrect answers regarding the assignment requirements and deadlines, as well as using harsh language.

To investigate this, the instructor can first navigate to the overview page. As shown in Fig. 2, the overview page consists of a list of student questions/chatbot answers. Each question/answer pair is associated with an entire set of chatbot configurations; to display those configurations concisely, only the base model configuration will be displayed using a colour-coded mark on this overview page, as the other configurations are either overly complex or not immediately relevant. Furthermore, each pair is associated with answer quality score(s), given by humans and/or NLP algorithms. Different points and/or lines will be used to visualize these aggregated scores, using 2D size, colour, and/or positions on a common scale to express the differences in scores.

The CPSC447 instructor can immediately identify specific questions in which the chatbot answers were marked as incorrect or unhelpful, by students and our algorithm.

To focus on specific questions, the instructor can select that row, which leads to the detail view. The detail view will show the full set of model configurations used to generated that chatbot answer. Notably, the retrieved document chunks and relevant document text can be... Similarly, the document,

In our scenario, the instructor discovers that the chatbot was giving incorrect information because the relevant document texts all pertain to an old assignment. The instructor posted new assignment instructions on their course website, but they failed to upload newer versions of the website into the RAG system. The instructor also decides to switch their base model from GPT4o to qwen, as they heard from a colleague that qwen has a more affirmative tone.

The instructor then makes those changes to the chatbot configurations in the visualization, and runs a new version of the model on the same student question. The visualization, shown in Fig. 3 juxtaposes the changes between the old and new runs using colour and positioning, and size. Once the run is complete, they can see that the new chatbot outputs the correct information, and retrieves the correct documents!

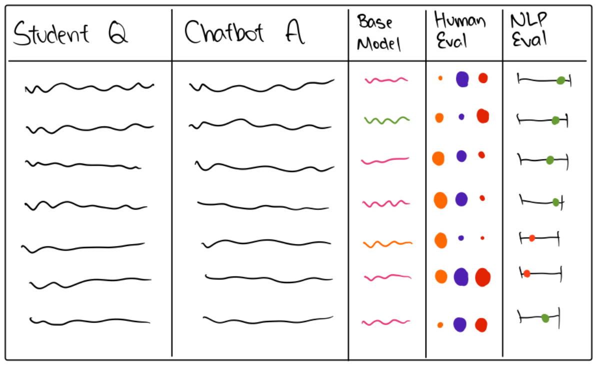


Fig. 2: A mockup of the overview page. We will consider other layouts, beyond a table, that highlight answers with poor human or NLP evaluation scores.

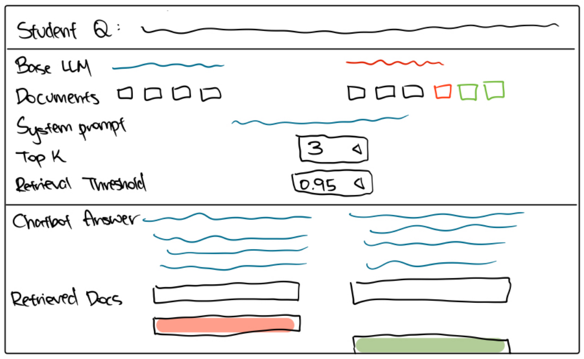


Fig. 3: A mockup of the detail page, juxtaposing an existing chatbot query with a different version with amended documents and a different base LLM. We will consider different methods to highlight the differences between different versions; this mockup uses single-dimensional positioning and colour, but there are many other possible channels.

The instructor then chooses to permanently update the chatbot configuration, meaning that they successfully used our system to identify, diagnose, and fix the problem.

This tool will be developed as a full-stack web application. The frontend will use React/JavaScript (particularly in D3.js), and the backend will be developed Node.js and PostgreSQL.

5 Milestones and Work Allocation

The major milestones are as follows:

By **the end of October**, we aim to have completed a basic, crude prototype of the system, with the basic functionality implemented but perhaps missing the specific visual encodings. Kevin will work on the backend setup (particularly with the RAG chatbots) as well as obtaining/creating data. Raymond will work on the frontend design of the system. In this phase, we also might revise some of the goals based on feasibility of certain goals and also potentially add different components.

By **mid-November** (in particular the update document deadline), we aim to refine the system, implementing and iterating on the specific visual encodings. Kevin will focus on the overview page, while Raymond will focus on the detail page.

By **the end of November**, we aim to have a final design that largely fulfills the goals outlined in this proposal. We will work together more closely in this phase, fleshing out any remaining details and making final refinements to the visual encodings.

6 Discussion, Future Work, and Conclusion

References

[1] I. Arawjo, C. Swoopes, P. Vaithilingam, M. Wattenberg, and E. L. Glassman. Chainforge: A visual toolkit for prompt engineering and llm hypothesis testing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pp. 1–18, 2024. 2

[2] N. Basit, M. Floryan, J. R. Hott, A. Huo, J. Le, and I. Zheng. Asci: Ai-smart classroom initiative. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSETS 2025, 7 pages, p. 81–87. Association for Computing Machinery, New York, NY, USA, 2025. doi: 10.1145/3641554.3701957 1

[3] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 1

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[5] S. Kakar, P. Maiti, K. Taneja, A. Nandula, G. Nguyen, A. Zhao, V. Nandan, and A. Goel. Jill Watson: Scaling and Deploying an AI Conversational Agent in Online Classrooms. In *International Conference on Intelligent Tutoring Systems*, pp. 78–90. Springer, 2024. 1

[6] M. A. Kuhail, N. Alturki, S. Alramlawi, and K. Alhejori. Interacting with educational chatbots: A systematic review. *Education and Information Technologies*, 28(1):973–1018, 2023. 1

[7] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, 2021. 1

[8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1

[9] H. Strobelt, A. Webson, V. Sanh, B. Hoover, J. Beyer, H. Pfister, and A. M. Rush. Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. *IEEE transactions on visualization and computer graphics*, 29(1):1146–1156, 2022. 2

[10] K. Taneja, P. Maiti, S. Kakar, P. Guruprasad, S. Rao, and A. K. Goel. Jill watson: A virtual teaching assistant powered by chatgpt. In *International Conference on Artificial Intelligence in Education*, pp. 324–337. Springer, 2024. 1

[11] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*, 2022. 1

[12] K. Wang and R. Lawrence. HelpMe: Student Help Seeking using Office Hours and Email. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education - Volume 1*. ACM, 2024. doi: 10.1145/3626252.3630867 1

[13] K. Wang, B. Pan, Y. Feng, Y. Wu, J. Chen, M. Zhu, and W. Chen. Xgraphrag: Interactive visual analysis for graph-based retrieval-augmented generation. In *2025 IEEE 18th Pacific Visualization Conference (PacificVis)*, pp. 1–11. IEEE, 2025. 2

[14] K. Wang, J. Ramos, and R. Lawrence. ChatEd: A Chatbot Leveraging ChatGPT for an Enhanced Learning Experience in Higher Education, 2023. https://arxiv.org/abs/2401.00052. 1

[15] T. Wang, J. He, and C. Xiong. Ragviz: Diagnose and visualize retrieval-augmented generation. *arXiv preprint arXiv:2411.01751*, 2024. 2

[16] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 1