In Search of a Better Search: Using RAG on Cornell Virtual Workshop Content

Zilu Wang
Cornell University
Center for Advanced Computing
zw427@cornell.edu

Susan Mehringer
Cornell University
Center for Advanced Computing
shm7@cornell.edu

Christopher J. Cameron
Cornell University
Center for Advanced Computing
cjc73@cornell.edu

ABSTRACT

The Cornell Virtual Workshop (CVW) is a continually updated HPC training platform with about forty roadmaps and more than two thousand pages, organized in a three-level hierarchy of roadmaps, topics, and pages. The CVW's traditional keyword search works poorly for users who lack domain terminology or want navigational guidance rather than a specific page. We describe a RAG-based chatbot that routes each query to a retriever matched to the appropriate level of the content hierarchy, ensuring that navigational queries return information about CVW structure and factual queries return specific page content. The chatbot is built with LangGraph and uses a hybrid BM25 and FAISS retrieval pipeline with cross-encoder reranking. A fixed-path pipeline provides prompt-injection defense, query rewriting, post-generation URL verification, and graceful handling of out-of-scope queries.

KEYWORDS

RAG, LLM, HPC, AI chatbot, Hybrid Retrieval, Hierarchical Retrieval, Query Routing, HPC Training, Workforce Development

1 INTRODUCTION

The Cornell Virtual Workshop [4] (CVW) is an online asynchronous training platform for high-performance computing developed by the Cornell Center for Advanced Computing, currently containing about forty roadmaps, with about 300K page views per year. We help people find the right content by organizing material into roadmaps, topics, and pages; adding keyword tags to each topic; and providing two types of traditional searches. In our observation, the quality of results from traditional site-specific searches depends on using specific terminology and keywords that may be unfamiliar to new learners.

We developed a Retrieval-Augmented Generation (RAG) chatbot based on CVW content to answer natural language queries without requiring CVW-specific terminology. The chatbot is intended to direct learners to the CVW materials, not replace them, so it provides a single, accurate answer that includes both the answer and a link to the in-context information.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page, along with the DOI information below. To copy otherwise, republish, post on servers or redistribute to lists requires prior specific permission from the first author.

© 2026 Journal of Computational Science Education
https://doi.org/10.22369/issn.2153-4136/17/2/6

We chose to use a RAG solution because it allowed us to use a general-purpose large language model (LLM) to answer queries using only CVW content and reduce AI hallucinations [6].

Our chatbot contributes the following:

- a hierarchy-aware RAG design that routes each query to a level-filtered retriever (roadmap, topic, or page), ensuring that navigational queries return structured overviews and factual queries return specific page content
- a fixed-path LangGraph pipeline with explicit nodes for prompt-injection defense, query rewriting, multi-level retrieval, post-generation URL verification, and graceful handling of vague and out-of-scope queries

2 RELATED WORK

Several recent efforts use LLMs to support HPC education and training. Within the HPC-ED project, Morsy et al. [8] demonstrate the potential of using LLMs as a quality assurance tool to review HPC training catalog metadata at scale.

Other works have focused on adding a retrieval layer between the LLM and HPC data. Miyashita et al. [7] extend RAG by allowing the system to execute a set of predefined commands and feed the output back as context for user queries.

Closest to our work, Bondapalli et al. [1] present AskHPC, a RAG-based chatbot for HPC users that uses facility user guides and scheduler and programming documentation as the corpus. In contrast, our chatbot is built around CVW, a hierarchically structured corpus, to route queries to roadmap-, topic-, and page-level retrievers.

3 SYSTEM DESIGN

CVW users ask questions at all three levels of its content hierarchy: some want a learning path, others a specific page, and others a topic overview. This section describes how we encode that structure in the chatbot's ingestion, retrieval, and routing components.

3.1 Corpus: Cornell Virtual Workshop

Cornell Virtual Workshop content is organized in a three-level hierarchy. At the top, a *roadmap* is a learning sequence on a single subject area, consisting of 5–10 topics listed in a suggested order. Each *topic* contains a set of individual *pages* that cover specific concepts, examples, or exercises. Each roadmap and topic includes structured metadata, such as a description, learning objectives, and prerequisites; roadmaps also specify system requirements. CVW pages follow the URL structure `cvw.cac.cornell.edu/<roadmap>/<topic>/<page>`.

3.2 Ingestion and Chunking

CVW pages are written as ASP.NET Razor pages with JSON, C#, and a database back end. The files used to build the corpus contain a mix of HTML markup and Razor directives. Before any file is processed, we use Python to strip out these elements, except for `<pre><code>` blocks, which are preserved verbatim. We do so by replacing each code block with a sentinel placeholder, removing the HTML tags and Razor directives, and then restoring the code blocks.

We then split pages with a recursive character splitter into 512-character chunks with 64-character overlap, treating each as an indexable document. We chose 512 characters because CVW pages are prose-heavy, and smaller chunks generally improve semantic retrieval when matching against user queries. Future work may evaluate other chunk sizes.

In addition to the chunked page documents, we construct a single *synthetic summary document* for each roadmap and each topic. A roadmap summary contains the roadmap's description, learning objectives, prerequisites, and system requirements, followed by a flat list of its topics with brief descriptions. A topic summary contains the topic's description, objectives, prerequisites, and keyword list, followed by a flat list of its pages by title. These documents are indexed unchunked because splitting them would sever the relationship between labeled fields and their values, and would distribute the child listing across separate chunks. If the retriever did not return all the chunks, the LLM could not accurately summarize roadmap and topic contents. The summaries are compact by design — typically a few hundred to roughly a thousand characters — so the cost of keeping them whole is low. Because each retriever is filtered to a single corpus level (roadmap, topic, or page), a generation node always receives either summary documents or page chunks.

Every chunk and summary document also carries structured metadata identifying its source, such as the roadmap and topic it belongs to, its CVW URL, and the topic description (or, for roadmap-level documents, the roadmap description).

3.3 Embeddings and Vector Store

In the embedding phase, all documents — both page chunks and synthetic summaries — are encoded using the `text-embedding-ada-002` embedding model from OpenAI. The resulting vectors are stored in a single FAISS (Facebook AI Similarity Search) index [3] shared across all three corpus levels. Level-filtered queries at retrieval time make this functionally equivalent to maintaining separate indexes per level, while simplifying index maintenance. The index is rebuilt from source only when a new rebuild token is set via an environment variable; otherwise it is loaded from disk on startup.

3.4 Pipeline

When a user sends a query, it passes through a pipeline implemented in LangGraph [5]. All queries follow a fixed sequence of nodes (Figure 1): Guard, Understand-and-Classify, one of Roadmap, Topic, Page, or Clarify, then URL-Verification, optionally Rate, then Log. Each node is a function that reads from and writes to a shared State object.

The initial Guard node strips invisible or non-printing Unicode characters from the input. It then checks the sanitized text with regular expressions against a set of common prompt-injection phrases, and also checks whether any base64 or hexadecimal runs in the query contain injection patterns when decoded. Queries shorter than 3 or longer than 2000 characters are also rejected. If any check fails, the pipeline exits immediately and returns a refusal without calling an LLM API.

The Understand node first rewrites the user query to produce a cleaned query that corrects grammatical or spelling errors and uses past conversations to form a complete question. For example, if a user first asks about Linux and then follows up with "How do I learn it?", the Understand node infers that "it" refers to Linux and produces a standalone query "How do I learn Linux?". The Understand node also generates a list of retrieval keywords, the main subjects of the query plus synonyms and related technical terms, that are likely to appear in relevant documents.

The Classify node routes the cleaned query to one of four nodes: Roadmap, Topic, Page, or Clarify. An LLM call at temperature 0 makes the routing decision on the cleaned query. The prompt defines the four routes: *roadmaps* for queries seeking a learning sequence or structured path; *topic* for queries that explicitly reference a CVW topic or ask for a topic-level overview; *page* for concrete how-to and reference questions, which is also the default when neither of the other labels applies; and *clarify* for out-of-scope, code-generation, and vague requests. Although originally implemented as two separate nodes, Understand and Classify were merged into a single LLM call to reduce round-trip latency.

The Clarify node handles queries that are out of scope or too vague to retrieve against. If the query is still within the HPC scope but vague, the node responds with a brief clarifying question; if the query is entirely out of scope, it returns a one-sentence refusal.

In the Roadmap, Topic, and Page nodes, the retrieval keywords generated by the Understand node serve as the search query; if no keywords are produced, the cleaned query is used instead. The query is passed to the corresponding retriever described in Section 3.5, and the top-n reranked documents are spliced into the answer generation prompt. We developed the routing prompt through iterative manual inspection of the 30-case internal example set described in Section 5. The selected node sends the system prompt, the top-n reranked documents, and the cleaned query to the LLM, which produces an answer. If no documents are retrieved, a fixed message states that the CVW does not have the requested topic, without calling the LLM.

Before returning the answer to the user, the URL-Verification node normalizes URLs and eliminates hallucinated URLs by comparing against a URL allowlist generated from the indexed CVW corpus. It strips invalid URLs and returns the final answer to the user.

After generating an answer, the Rate node optionally scores the response against a rubric, and the Log node stores the exchange in a local database.

3.5 Retrieval and Generation

As described in the pipeline section (Section 3.4), relevant queries are routed to one of three retrievers: roadmap, topic, or page. Each

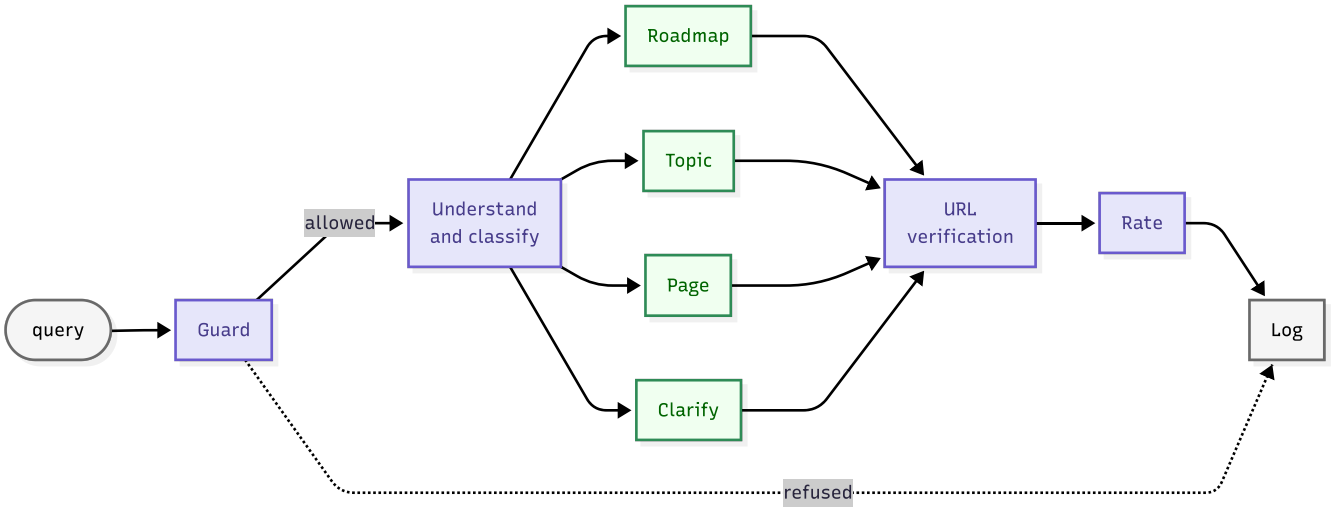


Figure 1: The CVW Chatbot pipeline. Queries come from users into Guard, where they are either allowed for further understanding and classification, or refused. Allowed queries are routed to one of four nodes, which all converge on URL verification before rating and logging.

retriever operates on the single FAISS index built in Section 3.3 and applies a `level` filter to ensure the returned documents remain within its scope. For example, the roadmap retriever returns only documents whose `level` field equals `"roadmap"`. Using multiple retrievers allows the chatbot to respond to roadmap- and topic-level queries because it hides details that would otherwise distract the LLM; the roadmap and topic retrievers operate on the synthetic summary documents, whereas the page retriever operates on the chunked page documents.

BM25 (Best Matching 25) [9] and FAISS are then run on the filtered documents, and a weighted ensemble combines their results, with BM25 at 0.4 and FAISS at 0.6. BM25 is a lexical ranking function that matches on exact terms, whereas FAISS retrieves documents whose embeddings are semantically closest to the query. The result is passed to FlashRank [2], which reranks the documents and returns the top n documents. For the roadmap retriever, we fetch 20 documents and rerank to the top 10; for the topic retriever, 30 and 20; for the page retriever, 30 and 20.

We group the reranked documents by their roadmap and topic, and prepend each group's metadata, such as description, objectives, prerequisites, and requirements, above the chunks. We then splice the grouped documents into a level-specific answer-generation prompt and pass it to OpenAI's GPT-5.4 at a temperature of 0.2. Each retrieved document includes a `Source URL:` line that we stored in metadata, which the generation prompt copies verbatim into the answer. A post-generation check then removes any URL not found in the indexed corpus, so every URL cited in a response points to a page in the indexed corpus.

3.6 Prompts

Each of the Understand, Classify, Roadmap, Topic, Page, Clarify, and Rate nodes includes a prompt that guides the LLM to perform the node-specific task. These prompts define much of the pipeline's behavior. Below, we highlight several prompt choices that shape the chatbot's answers.

- In the Roadmap, Topic, and Page nodes, the prompt instructs the LLM to append the URL from the source document's `Source URL:` to any answer it generates as a markdown-formatted URL with the page title as the link text. Together with the post-generation URL check, this ensures that a reference always exists to validate the answer.
- The prompt in the Page node also limits its answer to one or two sentences with CVW URLs, presented as a high-level teaser to encourage users to visit the link, rather than relying on the generated answer. We want to encourage users to discover and engage with the CVW material, rather than using the chatbot as a substitute.
- Several prompts also define out-of-scope behavior. Code-generation requests, for instance, are either routed to the Clarify node or refused outright when the query falls clearly outside CVW's HPC scope.

4 DEPLOYMENT

The chatbot is deployed as a search agent on the CVW website https://cvw.cac.cornell.edu. Depending on operating costs, we may impose access limits in the future by restricting access to logged-in users or imposing a per-session rate limit.

LLMs are accessed through the Cornell AI Gateway, which provides the GPT-5.4 model from OpenAI and applies Cornell-managed content safety guardrails, including prompt-attack and harmful-content filters applied to answer-generation calls. The chatbot's LLM usage is billed per token, and operators can monitor usage via the Cornell AI Gateway's LiteLLM dashboard. Using test data, we estimate that the cost will be approximately $0.05 per query.

The chatbot code runs on an on-premises CAC Red Cloud instance. The conversation history between users and the chatbot is

redacted of personally identifiable information before being stored in a database. It is used only for multi-turn conversations and development purposes.

We have observed limited user engagement. Initially, we restricted access to logged-in users and observed that users opened the agent chat page but did not interact with it or log in. We are reviewing the impact of both a login requirement and a privacy notice as potential barriers to adoption.

5 EVALUATION

To evaluate the chatbot, we constructed an internal test set of 30 cases, each targeting a specific case type: page-level queries, roadmap-level queries, topic-level queries, vague queries, out-of-scope queries, code requests, prompt injections, and content not covered by the CVW. We sampled these cases from the conversation log collected during development.

The Rate node evaluates chatbot responses on a 1–5 scale using the rubric below. The evaluator LLM first identifies which of three legitimate response types the query requires: (A) a cited answer for factual or learning-path queries within CVW's scope; (B) a clarifying question for vague or ambiguous in-scope queries; or (C) a one-sentence scope refusal for out-of-scope or adversarial queries. It then scores the match between the required type and the actual response, penalizing type mismatches and execution failures:

- **5 — Excellent.** Response type matches the query's need and execution is clean: a cited answer is accurate, brief, and includes a real CVW URL as a teaser; a clarifying question is focused and narrows intent; a scope refusal is one clean sentence with no hedging.
- **4 — Good.** Right type, minor issues: slight verbosity, a citation pointing to a related but not exact page, or a clarifying question broader than ideal.
- **3 — Mixed.** Right type but weak execution: tutorial-style prose rather than a teaser, a weak or generic citation, a hedged refusal that implies the bot might know elsewhere, or a graceful recovery from a routing error.
- **2 — Poor.** Wrong type for the query: a clarifying question when the query was directly answerable, answering an out-of-scope query as in-scope, or refusing an answerable in-scope query.
- **1 — Unacceptable.** Hallucinated CVW content, fabricated URLs, system prompt leakage, capitulation to prompt injection, or clearly unsafe output.

The bot generally handles in-scope queries well, producing concise answers with valid CVW URLs as citations:

> **User:** "What is the best way to transfer files to Vista?"
> **Bot:** "For Vista, CVW points to the standard secure transfer tools—**scp, sftp, and rsync**—as the main ways to move files from your local or remote machine, with the best choice depending on your file size and workflow. It also notes that transfers should follow TACC good-conduct guidelines.
> *Source:* Getting Started on Vista: Storing and Moving Data"

An example of roadmap recommendations:

> **User:** "I'm just getting started with HPC, what do I need to know?"
> **Bot:** "Here are a few good starting points for getting oriented to HPC and parallel computing:
> - Parallel Programming Concepts and High Performance Computing gives a broad introduction to HPC fundamentals, parallel terminology, and how to think about designing code for many-core systems.
> - Getting Started on Frontera fits if you want a practical overview of how an HPC system works, including environment setup, storage, compiling, and running jobs.
> - Data Transfer is useful for learning one of the most common HPC workflow tasks: moving code and data between your local machine, cloud storage, and remote systems.
>
> *Source:* Parallel Programming Concepts and High Performance Computing
> *Source:* Getting Started on Frontera
> *Source:* Data Transfer"

An example of an out-of-scope query:

> **User:** "Give me a recipe for burritos?"
> **Bot:** "That question is outside Cornell Virtual Workshop's scope — CVW focuses on HPC, parallel programming, MPI, CUDA, Slurm, Linux, Python performance, storage, and related computing topics."

The examples above are representative of the 30-case test set. In the test set, all injection attempts were refused. Twenty-four of the twenty-six queries with an expected route were routed correctly. The errors were vague queries routed to the roadmap retriever rather than the clarification node. The self-evaluation average is 3.6/5 for rateable responses. The LLM tended to evaluate itself more harshly than the human rater, and since the LLM-as-rater was not aware of the CVW content, it tended to give low ratings to refusals to answer out-of-scope questions that were plausible in-scope. A human review of the test query responses found that 25/30 responses were reasonably helpful; the human-identified error cases involved the LLM misunderstanding a question about the latest published material (2 cases), failing to fully cite the source (1 case), failing to return complete results (1 case), and failing to return results (1 case).

Once we have collected a meaningful volume of user queries, we will conduct a formal evaluation. Quantitatively, we will measure answer quality on a 1–5 scale using the LLM-as-judge method, and measure latency and cost per query as reported by the Cornell AI Gateway; we will measure routing accuracy separately by cross-checking answers with a human expert. Qualitatively, we will manually review low-rated answers identified by the Rate node, examine multi-turn conversations to assess behavior in a realistic setting, and use the results to identify pipeline weaknesses. The LLM-as-judge method is known to have several flaws, such as verbosity bias (LLMs favor longer responses) and self-enhancement bias (LLMs favor responses generated by the same model) [10]. A fully reliable evaluation will require manual review by a human expert.

6 FUTURE WORK

Future work falls into two areas. First, the chatbot is deployed but has seen limited user engagement to date. We plan to investigate barriers to adoption — including whether requiring a login or the presence of a privacy notice discourages use — and to improve chatbot response through prompt refinement based on real user queries. We plan to assess routing accuracy using a human-labeled test set as part of the post-deployment evaluation.

Second, while current configurations and parameters produce acceptable results, we chose the retrieval parameters and prompts heuristically rather than tuning them. We selected the BM25 weight (0.4) and FAISS weight (0.6) to prioritize semantic matches while still allowing keyword search. We chose the reranker model by comparing three FlashRank cross-encoders of different sizes (`TinyBERT-L-2`, `MiniLM-L-12`, and `MultiBERT-L-12`) on informal tests. The larger models introduced unacceptable response latency, so we selected `TinyBERT-L-2`. We set the `fetch_k` and `top_n` values based on the size of the CVW corpus at each level. A smaller candidate set is appropriate for the smaller roadmap corpus (about 40 documents). In comparison, the topic and page corpora are larger and benefit from a larger candidate set at the reranking stage. Future work will refine these configurations and parameters to reduce the cost per query and the response latency.

ACKNOWLEDGMENTS

This work was conducted with the support of the Cornell University Center for Advanced Computing and the Cornell AI Gateway.

REFERENCES

[1] Akhilesh Bondapalli, Huihuo Zheng, Oluwaseun T. Ajayi, Murat Keceli, Haritha Siddabathuni Som, Taylor Childers, Lisa Childers, Yasaman Ghadar, Michael Papka, Venkatram Vishwanath, and Rong Ge. 2025. AskHPC: A ChatBot for High Performance Computing User Support. In *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC Workshops '25)*. Association for Computing Machinery, New York, NY, USA, 727–739. https://doi.org/10.1145/3731599.3767433

[2] Prithiviraj Damodaran. 2023. FlashRank, Lightest and Fastest 2nd Stage Reranker for search pipelines. https://doi.org/10.5281/zenodo.10426927

[3] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:cs.LG/2401.08281

[4] Cornell Center for Advanced Computing. 2026. Cornell Virtual Workshop. Retrieved May 18, 2026 from https://cvw.cac.cornell.edu/

[5] LangChain AI. 2024. LangGraph. https://github.com/langchain-ai/langgraph. GitHub repository.

[6] 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. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:cs.CL/2005.11401 https://arxiv.org/abs/2005.11401

[7] Yusuke Miyashita, Patrick Kin Man Tung, and Johan Barthélemy. 2024. LLM as HPC Expert: Extending RAG Architecture for HPC Data. arXiv:cs.DC/2501.14733 https://arxiv.org/abs/2501.14733

[8] Habiba Morsy, Essence Toone, Charlie Dey, Zilu Wang, Mary Thomas, and David Joiner. 2026. HPC-ED: Testing Automated Agents to Assess the Quality of Training Resource Metadata. *The Journal of Computational Science Education* 17 (March 2026), 34–41. Issue 1. https://doi.org/10.22369/issn.2153-4136/17/1/5

[9] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3 (09 2009), 333–389. https://doi.org/10.1561/1500000019

[10] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv:cs.CL/2306.05685 https://arxiv.org/abs/2306.05685