An Agentic AI Approach for Hands-on Networking Education

Akhil Gorthi Bala Sai
University of South Carolina
Columbia, SC, USA
akhilgbs@sc.edu

Ali Mazloum
University of South Carolina
Columbia, SC, USA
amazloum@email.sc.edu

Jose Gomez
Fort Lewis College
Durango, CO, USA
jagomez@fortlewis.edu

Samia Choueiri
University of South Carolina
Columbia, SC, USA
choueiri@email.sc.edu

Amith GSPN
University of South Carolina
Columbia, SC, USA
amithgspn@sc.edu

Elie Kfoury
University of South Carolina
Columbia, SC, USA
ekfoury@email.sc.edu

Jorge Crichigno
University of South Carolina
Columbia, SC, USA
jcrichigno@cec.sc.edu

ABSTRACT

Hands-on networking education relies on laboratories where learners configure networked systems, observe behavior, and debug failures. Virtual labs make this experience scalable, but support remains difficult because useful guidance often requires knowing both the intended procedure and the current state of a learner's environment. A lab manual describes the topology, setup steps, commands, and expected outputs, but it cannot inspect whether a learner has missed a step, configured the wrong interface, or produced an inconsistent routing state. General-purpose Large Language Models (LLMs) can explain networking concepts, but they are not grounded in a specific lab manual or in the live state of a running lab instance. They may also provide final commands too early, reducing the opportunity for learners to reason through the debugging process.

This paper presents an agentic AI approach for hands-on networking education. The system is built around an IPv6 configuration and routing training sequence, with a static-routing lab used as the running example, while the design targets virtual networking labs more broadly. The agent builds a lab-specific knowledge base from manuals and structured notes, uses Retrieval-Augmented Generation (RAG) to retrieve procedural and diagnostic context, and uses read-only live-state tools to inspect the learner's lab instance when an answer depends on current state. A human-audited evaluation over 100 lab-specific questions compares three conditions: LLM-only, document-only RAG, and the full agentic system. The full system answers 91 questions correctly and 9 partially, compared with 79 correct and 9 partial answers for document-only RAG and 34 correct and 17 partial answers for LLM-only. The results suggest that agentic AI can offer a promising direction for scalable support in hands-on networking education.

CCS CONCEPTS

• **Applied computing** → **Interactive learning environments**; *Computer-assisted instruction*; • **Computing methodologies** → *Artificial intelligence*; • **Information systems** → *Retrieval models and ranking*;

KEYWORDS

Agentic AI, networking education, virtual laboratories, retrieval-augmented generation, large language models, educational scaffolding

1 INTRODUCTION

Hands-on networking education offers laboratories where learners configure networked systems and observe the effects of their decisions. Virtual laboratories make this experience scalable without requiring dedicated physical infrastructure. In these environments, learners can build topologies, configure hosts and routers, test reachability, and collect measurements in a repeatable setting [3, 5].

A typical lab is accompanied by a manual that guides the learner through the exercise. The manual describes the topology or setup, the steps to follow, the commands to run, the outputs to inspect, and the expected behavior. However,

Acknowledgment: This material is based upon work supported by the National Science Foundation under Grants Nos. 2400729 and 2417823.

the manual alone cannot diagnose what is happening in a learner's environment. For example, if a connectivity test fails, the cause may be a missing route, an incorrect interface, an incomplete setup step, a command copied from the wrong part of the manual, or a conceptual misunderstanding. Human teaching assistants handle these cases by asking for evidence, checking the current state, and giving targeted guidance. Providing that level of support is not practical in self-paced or remote labs.

LLMs are attractive for this setting because they can answer questions in natural language and explain general networking concepts. However, a generic LLM is not grounded in the specific lab manual, topology, commands, expected outputs, or current lab state. As a result, it may suggest commands or explanations that sound reasonable but are very generic and do not apply to the learner's exercise. It can also provide final answers too early, before the learner has inspected the relevant evidence or reasoned through the problem. This creates an assistance dilemma: too little help leaves learners blocked, while too much help can reduce the opportunity to learn [13, 22].

This paper presents an agentic AI approach for hands-on networking education. The approach is designed for virtual networking labs broadly. An IPv6 routing and configuration lab is used as a running example throughout the paper. The agent uses Retrieval-Augmented Generation (RAG) to ground responses in lab documentation and uses read-only diagnostic tools to collect information from the running lab environment. The documentation provides the intended procedure, while the diagnostic tools inspect what is happening in the learner's lab instance. The goal is to provide guidance that is grounded in the lab context without immediately giving final solutions.

The main contributions of this paper are as follows:

- An agentic AI system for hands-on networking education that combines lab-specific documentation with read-only inspection of the running lab environment.
- A Retrieval-Augmented Generation (RAG) workflow that uses lab manuals and supplemental lab notes to provide context for conceptual, procedural, and diagnostic questions.
- A guidance strategy that provides concise, evidence-based help without immediately giving complete solutions.
- A preliminary evaluation showing that the system retrieves relevant lab-specific evidence for most test questions and that live-state inspection reveals diagnostic information unavailable from static lab documents alone.

The remainder of the paper is organized as follows. Section 2 provides background on virtual networking laboratories, agentic AI systems, retrieval-augmented generation, and educational scaffolding. Section 3 presents the proposed system. Section 4 presents the evaluation methodology and results. Section 5 reviews related work. Section 6 outlines future work. Section 7 concludes the paper.

2 BACKGROUND

This section provides background on virtual networking laboratories, agentic AI systems, retrieval-augmented generation, and educational scaffolding.

2.1 Networking Laboratories

Virtual laboratories make hands-on networking education practical by providing realistic systems without requiring dedicated physical infrastructure. Learners can configure hosts, switches, routers, links, routing behavior, traffic generators, and measurement tools, then observe how protocol and configuration decisions affect reachability and performance. A virtual learning environment provides each learner with a dedicated virtual machine or lab instance. Inside that instance, tools such as Mininet can emulate network topologies using lightweight OS-level virtualization [14]. Container-based emulation can serve a similar role, supporting reproducible experiments with resource isolation, monitoring, and repeatable scripts [8]. This layered setup lets learners work with realistic network behavior while the surrounding platform handles isolation, reset, and remote access.

Recently, research testbeds are also becoming part of the educational landscape. Platforms such as FABRIC [2] support instructional use by allowing instructors to create courses, enroll learners, and provide access to programmable infrastructure. These platforms extend the range of environments available for networking education, from course-managed virtual machines to shared experimental infrastructure. Faculty are leveraging such infrastructures to provide guided instructions to students on various networking topics [18].

Another common delivery model is the self-paced lab. In these labs, learners often follow step-by-step instructions without immediate human guidance. Support is challenging because questions may involve both conceptual understanding and the current state of the lab environment. A missing route, incorrect interface name, incomplete setup step, or misinterpreted measurement can block progress even when the learner understands the general networking idea. Useful guidance may therefore require knowing both what the lab manual asked the learner to do and what is currently happening inside the learner's lab instance.

Regardless of the training type, there is a support challenge: useful guidance may require knowing both what the

lab manual asked the learner to do and what is currently happening inside the learner's lab instance.

2.2 Agentic AI Systems

Agentic AI refers to systems that can make decisions, gather information, and take actions toward a task goal with some degree of autonomy. In recent LLM-based agents, a language model often serves as the reasoning component that decides when additional context is needed and how to obtain it. This context may be obtained by calling tools, querying APIs, running commands, or inspecting external environments before producing a response [17, 24]. This pattern is useful when the answer depends on a state that is not contained in the prompt itself.

2.3 Retrieval-Augmented Generation

Large Language Models can answer many general questions, but they do not automatically know the task-specific context needed for a particular use case. Retrieval-Augmented Generation (RAG) addresses this limitation by leveraging an external document collection and conditioning the model's answer on retrieved evidence [15]. In networking education, this context may include a lab manual, setup instructions, topology descriptions, command references, expected outputs, and notes about common failure cases. Grounding responses in these materials helps keep answers aligned with the actual exercise rather than the model's general memory.

Retrieving relevant evidence can be implemented in several ways. BM25 is a lexical search method that works well when a query contains exact words or identifiers that also appear in the documents [16]. This property is important in networking labs because strings such as interface names, route prefixes, host labels, and configuration paths can be decisive. Dense retrieval uses learned vector representations to find passages with similar meaning even when the wording differs [9]. Hybrid approaches combine lexical and dense retrieval, and Reciprocal Rank Fusion (RRF) is a simple way to merge their ranked lists [4].

2.4 Educational Scaffolding

Educational support should help learners make progress while preserving the opportunity to think through the lab task. In hands-on networking education, this includes interpreting observations, connecting those observations to networking concepts, and deciding what evidence to inspect next. Prior work on computer-supported instruction has studied how systems can provide feedback and hints at an appropriate level of detail [19]. Scaffolding refers to temporary support that helps learners perform tasks that would otherwise be difficult to complete alone [22].

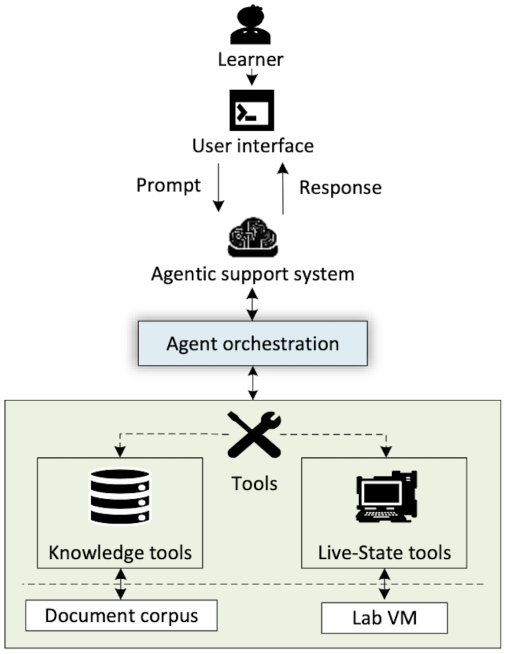

Figure 1: Architecture of the proposed agentic support system. The agent consults lab documentation and obtains observations from the running lab instance before responding to the learner.

A related challenge is the assistance dilemma: too little help can leave the learner stuck, while too much help can reduce the learner's engagement with the underlying problem [13]. Networking labs make this dilemma concrete. Providing the final command may help a learner complete a step, but it can also hamper the process of interpreting symptoms, checking evidence, and understanding why the command is needed.

3 PROPOSED SYSTEM

3.1 System Overview

The proposed system is an agentic support workflow for hands-on networking labs. Figure 1 shows the architecture. A learner asks a question about the lab, and depending on the question, the agent consults lab documentation, obtains observations from the running lab instance, or combines both sources. The documentation provides procedural and conceptual context, while live-state observations show what is currently happening in the learner's environment. The agent uses this evidence to produce a concise response for the learner.

3.2 Knowledge Tools

The knowledge tools provide access to lab-specific documentation. The lab manual is the primary source of documentation: it defines the exercise by describing the topology, setup steps, commands, expected outputs, and learning goals. However, manuals often omit context that is useful during support, such as why a command matters, what an intermediate state should look like, which failures are common, and how observed outputs should be interpreted.

For each lab exercise, the system performs an offline preparation step before learner interaction. During this step, the agent completes the exercise in the lab environment and documents support-relevant information. This preparation is done once when the lab is added to the system, so learners do not wait for the agent to exercise the lab during a support session. The documented information includes:

- The topology, lab steps, commands, and expected outputs described by the manual;
- Intermediate and final states observed while completing the exercise, such as configured interfaces, expected routes, service state, and reachability results;
- Representative command outputs and measurements, such as routing output, address configuration, ping results, or traceroute results;
- Common diagnostic cases, including failures that are not fully explained in the manual.

The resulting supplemental notes are organized as structured Markdown and aligned with the corresponding manual sections and lab steps. They do not replace the manual; they make implicit lab knowledge explicit so that the agent can retrieve context that is closer to what an experienced instructor or teaching assistant would know after running the lab. During support, the knowledge tools retrieve from this documentation rather than relying only on the model's general knowledge. Figure 2 shows a simplified excerpt from the IPv6 running example. The excerpt describes an asymmetric reachability problem: traffic from h1 may reach h2, but the reply cannot return because one router is missing the reverse static route. The excerpt records the symptom, the route-table evidence to inspect on both routers, and the hint sequence used to guide the learner from comparing routes to identifying the missing return path.

The system exposes two knowledge tools to the agent:

- `lookup_lab_docs(question)` retrieves relevant passages from the lab manual and supplemental notes.
- `get_lab_step(section, step)` retrieves a specific numbered lab step when the learner refers to the manual by section or step number.

The manual is split into sections, subsections, and steps, while the Markdown notes are split by headings. This structure keeps commands, observations, and explanations close to the lab context in which they are used. The retrieval process supports several configurations, including BM25-only, dense-only, and hybrid. BM25 helps preserve exact networking identifiers such as IPv6 prefixes, interface names, paths, and routing-service names. Dense retrieval helps when a learner uses different wording than the manual or supplemental notes.

```
### Configured the static route in the wrong direction

Symptom:
- Asymmetric ping: h1 to h2 works in one direction, but the reply
    never comes back.
- From h1's view: 100% packet loss, but no Destination Unreachable.

Detect:
- m r1 /tmp/vt-show-r1.sh shows S>* for :2::/64.
- m r2 /tmp/vt-show-r2.sh does not show S>* for :1::/64, or vice
    versa.

Hint progression:
- "Routing is one-way per router. If h1 can reach h2 but h2 cannot
    reach h1, one router is missing its reverse route."
- "Check show ipv6 route on r2: do you see a static route to
    2001:db8:abcd:1::/64?"
-   "On   r2,   configure   ipv6   route   2001:db8:abcd:1::/64
    2001:db8:abcd:12::1."
```

Figure 2: Simplified example of supplemental Markdown documentation used by the knowledge tools.

The knowledge tool also supports expected-state questions, such as which routes, addresses, or connectivity results should appear after a lab step. These questions require exact lab-specific facts, but they still arise from the lab documentation rather than from the learner's live environment. The system therefore treats structured expected-state records as a specialized part of the knowledge base. Document retrieval handles the general case, while the structured records are used when the question asks for an exact expected state or a numbered lab step. Figure 3 summarizes this workflow.

The step lookup tool is deterministic rather than fuzzy. This matters because lab manuals often reuse step numbers across sections; searching only for "Step 8" may return the wrong procedure. Together, the knowledge tools support conceptual questions, procedural questions, and questions about expected lab state.

3.3 Live-State Tools

In addition to retrieving lab documentation, the agent can obtain observations from the running lab instance. This capability distinguishes the system from a document-only LLM interface. Documentation can describe the expected state of the lab, but live-state tools can show whether the learner's current environment actually matches that state.

In the IPv6 running example, these observations are collected through SSH-backed tool interfaces. These interfaces

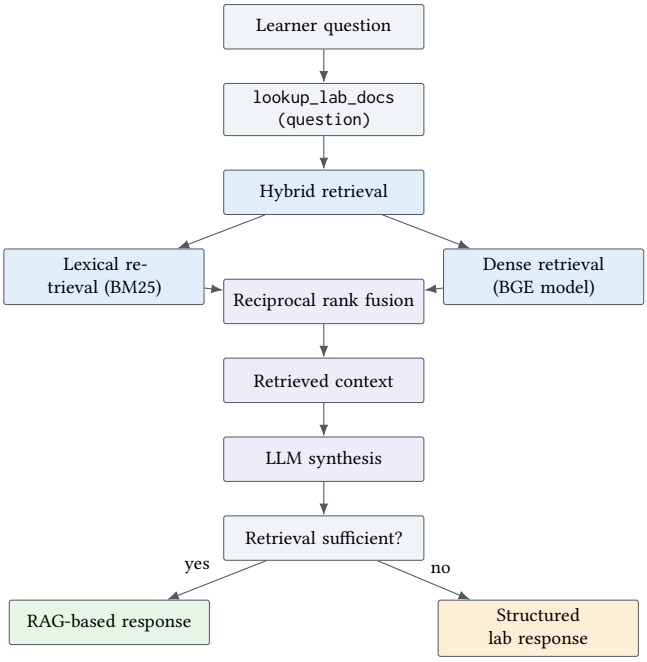

Figure 3: Knowledge lookup workflow. The agent first searches lab documentation using hybrid retrieval. If retrieved documentation is not enough for an expected-state question, the system can fall back internally to a structured description of the lab's expected state.

are not commands typed directly by learners; they are controlled system tools that wrap the commands needed to inspect the lab. For example, `vtysh_command` can query router state through `vtysh`, `ping6_from` can run an IPv6 reachability test from a selected host, and `recent_vtysh_history` can inspect recent router CLI history. A broader snapshot tool collects process state, interface and address state, forwarding status, and route output. Some lab artifacts, such as router startup files, may be readable only through elevated privileges; the tool layer exposes these cases through a whitelisted read-only file interface rather than general root command execution.

These tools allow the agent to compare the expected and observed states. For example, the expected lab state may include a static route, but the running lab instance may show that the route is missing, a routing service is inactive, or reachability fails in only one direction. Each tool call is audited so that later analysis can see what evidence supported the response.

Table 1 summarizes the tools used in the running-example IPv6 lab. Other labs can expose different tools while keeping the same architecture. For example, a performance measurement lab might include traffic-generation or packet-capture tools, while a routing lab may include route-table and reachability tools.

Table 1: Tool families used in the running-example IPv6 lab.

Family	Tools used in the IPv6 example
Knowledge	`lookup_lab_docs`, `get_lab_step`
Live state	`lab_state_snapshot`, `vtysh_command`, `ping6_from`, `recent_vtysh_history`
Filesystem	`list_directory`, `read_file`, `read_privileged_lab_file`, `create_directory`, `create_file`, `delete_file`

3.4 Agent Orchestration

The orchestration layer coordinates the interaction among the learner, the language model, and the tool interfaces. For each question, it determines whether the available conversation context is sufficient or whether the agent should first gather evidence from the lab documentation, the running lab instance, or both. Retrieved passages and live-state observations are returned to the model as evidence before it produces the learner-facing response.

The orchestration layer also maintains session context. Follow-up questions often depend on earlier observations: a learner may ask why an IPv6 reachability test fails, then ask what a route-table entry means, then ask what should be checked next. The maintenance of session state allows the agent to connect these turns without treating each question as an isolated request.

Finally, orchestration provides the policy boundary around tool use. Read-only diagnostic tools can be used to inspect the lab state, while operations that could modify files or the lab environment require human approval. Tool invocations are recorded with the tool name, arguments, success status, exit code, and output preview, making it possible to audit whether a response was grounded in documentation, live-state observations, or neither.

3.5 Instructional Constraints

The system is designed to provide support without turning the lab into an answer-generation interface. Its responses are constrained to address the learner's immediate question, remain concise, and preserve opportunities for the learner to reason through the task. To distinguish guidance from complete solutions, the agent is constrained to provide step-by-step guidance. When a learner reports a problem, the agent first identifies the likely issue and the evidence behind it, rather than immediately providing every command

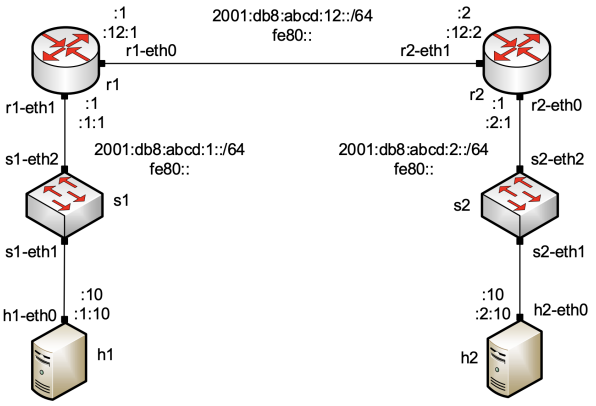

Figure 4: Topology used in the IPv6 example.

needed to complete the exercise. Then it asks whether the learner wants help before giving the corrective steps. If the learner agrees, the agent begins by providing hints that direct attention to the relevant lab step or observed state. The complete command sequences are reserved for the cases where the learner explicitly asks for them, remains stuck after guidance, or authorizes the agent to apply a fix.

The system also treats lab-specific facts as evidence-based claims. If a response depends on the lab manual, topology, router configuration, expected output, or current lab state, the agent is expected to use the relevant knowledge or live-state tool rather than answer from general networking knowledge alone. This serves as a hallucination-mitigation mechanism. When the learner's questions are ambiguous or underspecified, the agent is instructed to ask for clarification or inspect the relevant lab state before answering. This constraint reduces the risk of incorrect advice, which is especially important because imprecise questions are common in interactive lab settings.

These constraints are enforced through both prompts and software structure. Prompt instructions define the desired response style, while the orchestration layer controls which tools are available, when tool outputs are returned to the model, and which actions require approval. This design keeps educational guidance, evidence grounding, and operational safety as explicit parts of the system rather than optional behaviors left entirely to the language model.

4 EVALUATION

The evaluation analyzes whether combining lab documentation with live-state inspection improves support for lab-specific learner questions. The full agentic system is compared against two baselines: a generic LLM and a document-only RAG system. Results are reported for generated-answer accuracy and retrieval quality.

Table 2: Question categories used in the evaluation dataset.

Category	Example question
Conceptual	Why are IPv6 addresses alone not enough for `h1` and `h2` to communicate through `r1` and `r2`?
Procedural	What `ipv6 route` command should be entered on `r1` to reach the `h2` LAN?
Expected state	After the `r2` route is configured, what static route should appear for the `h1` LAN?
Diagnostic	If traffic reaches `h2` but replies never return to `h1`, what route is likely missing?
Identifier	What is the difference between `/root/.history_frr` and `/home/frr/.mininet_history`?

4.1 Experimental Setup

The running example is a basic IPv6 static routing lab whose topology is shown in Figure 4. The lab includes a step-by-step manual, supplemental structured notes created while completing the lab, and an active VM running a Mininet topology with hosts `h1`/`h2` and routers `r1`/`r2`. In the full agentic condition, SSH-backed read-only tools can inspect process state, interface and address state, routing state, IPv6 forwarding state, reachability, selected lab files, and recent router command history.

The evaluation dataset contains 100 lab-specific learner questions from the lab. The questions are organized into five categories with 20 questions per category, as shown in Table 2. The categories are informed by prior work on question asking during tutoring [7]. The categories are adapted to hands-on networking labs by separating conceptual, procedural, expected-state, diagnostic, and identifier-heavy questions. The identifier-heavy category captures a networking-specific challenge: many support requests depend on exact commands, paths, interfaces, prefixes, or daemon names.

4.2 Answer Accuracy

Each generated answer is labeled as correct, partially correct, or incorrect using a rubric that checks whether the answer is technically correct, grounded in the lab, and responsive to the learner's question. All three answer-generation conditions use OpenAI GPT-OSS [1]. For grading, a separate rubric-based judge, also using OpenAI GPT-OSS, first compares each answer against manually identified reference evidence from the lab documentation. A human reviewer then inspects the generated answer, the judge rationale, and the reference evidence. The final labels reported reflect this manual audit rather than the automated judge output alone.

Table 3: End-to-end answer accuracy over 100 lab-specific questions. Score counts correct answers as 1 and partial answers as 0.5.

Condition	Corr.	Part.	Inc.	Score
LLM only	34	17	49	0.425
Document-only RAG	79	9	12	0.835
Full agentic system	91	9	0	0.955

The LLM-only condition answers without lab-specific retrieved context. The document-only RAG condition answers using retrieved passages from the lab corpus but does not inspect the running VM. The full agentic system can use both retrieved documentation and live-state tools.

Table 3 shows that grounding is essential for this lab-specific setting. The LLM-only baseline answers only 34 of 100 questions correctly and performs especially poorly on procedural and expected-state questions that require exact commands, addresses, routes, or file paths. Document-only RAG improves strict correctness to 79 of 100 answers. The full agentic system answers 91 questions correctly, has no answers judged incorrect after manual inspection, and obtains the highest partial-credit score. This pattern suggests that tool orchestration improves lab-specific answering when the agent can combine retrieved documentation with targeted observations from the running environment. Table 4 illustrates this pattern through representative questions: the LLM-only baseline often answers from general networking knowledge, document-only RAG improves when the needed fact is present in the corpus, and the full agentic system is strongest when the answer depends on exact files or current VM state.

4.3 Retrieval Results

For the retrieval analysis, each question is associated with one or more manually identified reference passages from the lab documentation. A query is counted as a Hit@5 if any reference passage appears among the top five retrieved chunks. Hit@5 is reported because the deployed system passes five retrieved chunks to the language model. Mean Reciprocal Rank (MRR) is also reported because it rewards systems that rank the first relevant passage closer to the top of the retrieved list.

The retrieval comparison includes three configurations. The BM25 baseline uses lexical matching with identifier-aware tokenization. The dense baseline uses BGE-small embeddings [23]. The deployed configuration uses hybrid retrieval, which combines BM25 and dense retrieval with reciprocal rank fusion.

Table 5 shows that BM25 achieves the highest aggregate Hit@5, retrieving a reference passage for 90 of the 100 questions. This result reflects the importance of exact technical strings in networking labs, including command names, file paths, interface labels, daemon names, and IPv6 prefixes. Dense retrieval is weaker overall on this benchmark, with 69/100 Hit@5, which suggests that semantic similarity alone is not sufficient when the answer depends on exact lab artifacts.

The deployed hybrid configuration shows its clearest benefit on procedural questions, where it retrieves a reference passage for 18 of 20 questions compared with 15 of 20 for BM25. This category often includes learner phrasing that differs from the wording of the manual, such as asking how to complete, check, or recover from a lab action rather than naming the exact step text. In these cases, the dense component can recover semantically related passages that lexical matching alone may miss.

The categories where BM25 is stronger are those with more exact matching requirements. Expected-state questions often depend on literal lab artifacts, such as step numbers, route prefixes, interface names, host labels, or expected command outputs. Identifier-heavy questions have the same property by design. In these cases, lexical matching can rank the exact passage higher than hybrid retrieval, where the dense component may promote semantically related but less exact chunks.

The current implementation uses hybrid retrieval as the deployed configuration because it improves procedural retrieval while preserving competitive overall performance; future versions could route different question types to different retrieval strategies.

5 RELATED WORK

Related work spans LLMs for network configuration and troubleshooting, document-grounded and tool-augmented assistants, and LLM-based educational support.

5.1 LLMs for Network Configuration and Troubleshooting

Recent studies have evaluated LLMs on networking tasks. NetConfEval examines network-configuration problems and shows both the potential and limitations of language models for configuration assistance [20]. NIKA studies LLM-driven agents for network troubleshooting and finds that agents can detect some network issues but still struggle with fault localization and root-cause identification [21]. These systems are closely related because they examine LLMs in technical networking contexts.

The proposed approach differs because the objective is educational support rather than operational repair. Network

Table 4: Representative qualitative examples from the answer-accuracy evaluation. Cells show short excerpts from generated answers.

Question	LLM only	Document-only RAG	Full agentic system
Which configuration file does `zebra` load on `r1`, and which interfaces does it configure?	I don't have the exact details of your lab topology, so I can't tell you the precise file name or the exact list of interfaces.	The provided excerpts only show that the zebra daemon is started on `r1` with the command: `zebra`. They do not specify which configuration file zebra reads.	Zebra on `r1` reads the file `/etc/routers/r1/zebra.conf` … it configures … `r1-eth1` … `r1-eth0`.
What IPv6 addresses and MAC address should `r1-eth1` have after `zebra` reads its configuration?	I'm not able to give the exact IPv6 address(es) or MAC address for `r1-eth1` without seeing the configuration.	After zebra reads … `/etc/routers/r1/zebra.conf` it assigns two IPv6 addresses … `fe80::1:1/64` … `2001:db8:abcd:1::1/64`. The MAC address … is not listed.	IPv6 addresses on `r1-eth1`: … `fe80::1:1/64` … `2001:db8:abcd:1::1/64`. MAC address … `00:00:00:00:1:1`.
If traffic reaches `h2` but replies do not return to `h1`, what route is likely missing?	The problem is almost always a missing return route … the device on the far side … does not know how to reach the source network.	For this lab the two required static routes are … on `r1`: `ipv6 route 2001:db8:abcd:2::/64 . . .` … on `r2`: `ipv6 route 2001:db8:abcd:1::/64 . . . .`	The missing piece is the reverse static route–the static route on the opposite router that points back to the source network.

configuration and troubleshooting benchmarks usually evaluate whether the system can produce or identify a correct technical outcome. In self-paced labs, the system must also preserve the learning process. A useful response may need to point the learner to relevant evidence, explain what to inspect next, or provide a hint rather than immediately producing the final command.

5.2 Document-Grounded and Tool-Augmented Assistance

Retrieval-Augmented Generation grounds LLM responses in external documents [15]. For technical labs, grounding is important because answers often depend on lab-specific commands, topology assumptions, expected outputs, and numbered steps. Retrieval methods such as BM25, dense retrieval, and reciprocal-rank fusion provide mechanisms for finding relevant documentation [4, 9, 16]. However, document retrieval alone cannot determine whether the learner's current environment matches the intended lab state.

Table 5: Retrieval results over 100 lab-specific questions. Each category contains 20 questions. Cells report Hit@5.

Question set	BM25	Dense	Hybrid
Conceptual	20/20	11/20	14/20
Procedural	15/20	13/20	18/20
Expected state	17/20	12/20	14/20
Diagnostic	18/20	17/20	18/20
Identifier	20/20	16/20	19/20
Overall	90/100	69/100	83/100

Tool-augmented LLM agents address missing context by allowing models to gather information from external environments [17, 24]. In this paper, tool use is constrained by the educational setting: tools provide read-only inspection of live lab state rather than automatically modifying the environment or completing the task. This makes tool use a way to ground guidance in evidence, not a mechanism for bypassing the learner's reasoning process.

5.3 LLM-based Educational Support

Educational dialogue systems and intelligent tutoring systems have long studied how automated systems can guide learners through problems using feedback, hints, and mixed-initiative interaction [6, 19]. Work on LLMs in education highlights opportunities for natural-language assistance as well as risks such as factual errors, overreliance, privacy concerns, and answers that bypass the learning process [10]. These risks motivate response constraints that help learners reason through a task rather than simply receive final answers.

Most educational dialogue systems operate over a curriculum script, problem statement, or submitted answer. Hands-on networking labs add a state-dependent dimension: a learner's question may depend on files, interfaces, routes, processes, or command outputs inside a running lab instance. The system studied in this paper addresses that setting by combining educational guidance with lab documentation and live-state inspection.

6 LIMITATIONS AND FUTURE WORK

The evaluation remains preliminary. The answer-accuracy benchmark measures lab-specific question answering, but it does not measure learning outcomes, long-term learner

behavior, or whether hint-oriented responses improve debugging skill. The experiments also do not yet measure diagnostic success over a systematic set of injected faults. Future work will evaluate these dimensions and will also study model-size sensitivity and comparisons with additional state-of-the-art LLMs.

Future work will also extend the system from an IPv6-focused training sequence to broader coverage of networking education. The current development targets a set of IPv6 configuration and routing labs, with one static-routing lab used as the detailed running example in this paper. Next steps include support for additional topics such as traffic measurement, congestion-control analysis, programmable networks (P4 switches [12], data processing units and SmartNICs [11]), intrusion detection (Zeek), and others. Expanding the documentation workflow and live-state tool layer would allow the agent to support a wider range of networking concepts and lab workflows.

Future evaluation will also use real learner inputs from deployed educational settings. The virtual learning platform targeted by this work is used broadly in networking education, including courses at hundreds of universities across the United States and hands-on tutorial and workshop events. Subject to appropriate consent, privacy review, and anonymization, future studies will collect learner prompts, tool-use traces, lab progress, environment-state snapshots, and outcome data from both university courses and workshop deployments. Such data would allow the system to be evaluated on naturally occurring questions and lab states, rather than only on a constructed benchmark.

7 CONCLUSION

This paper presented an agentic AI approach for self-paced virtual networking labs. The central idea is that useful support in this setting must combine two forms of evidence: the lab materials that define the intended task and the live lab state that reflects what the learner has actually done. The system uses lab-specific documentation, retrieval over that documentation, read-only diagnostic tools, and response constraints intended to support learning rather than simply provide final commands.

The preliminary evaluation suggests that this direction is promising. The answers' accuracy results indicate that the full agentic system improves over both an LLM-only baseline and a document-only RAG baseline, while the retrieval results show that the system can often find lab-specific evidence. These results support the broader argument that scalable assistance for hands-on networking education should combine lab documentation, live-state inspection, and guidance policies that preserve opportunities for learner reasoning.

REFERENCES

[1] Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925* (2025).

[2] Ilya Baldin, Anita Nikolich, James Griffioen, Indermohan Inder S Monga, Kuang-Ching Wang, Tom Lehman, and Paul Ruth. 2020. Fabric: A national-scale programmable experimental network infrastructure. *IEEE Internet Computing* 23, 6 (2020), 38–47.

[3] Douglas E Comer. 2003. *Hands-on networking with internet technologies*. Prentice-Hall, Inc.

[4] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. 2009. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 758–759. https://doi.org/10.1145/1571941.1572114

[5] Jose Gomez, Elie F Kfoury, Jorge Crichigno, and Gautam Srivastava. 2023. A survey on network simulators, emulators, and testbeds used for research and education. *Computer Networks* 237 (2023), 110054.

[6] Arthur C. Graesser, Patrick Chipman, Brian C. Haynes, and Andrew Olney. 2005. AutoTutor: An Intelligent Tutoring System with Mixed-Initiative Dialogue. *IEEE Transactions on Education* 48, 4 (2005), 612–618. https://doi.org/10.1109/TE.2005.856149

[7] Arthur C. Graesser and Natalie K. Person. 1994. Question Asking During Tutoring. *American Educational Research Journal* 31, 1 (1994), 104–137. https://doi.org/10.3102/00028312031001104

[8] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. 2012. Reproducible Network Experiments Using Container-Based Emulation. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '12)*. Association for Computing Machinery, New York, NY, USA, 253–264. https://doi.org/10.1145/2413176.2413206

[9] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*. Association for Computational Linguistics, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

[10] Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. 2023. ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education. *Learning and Individual Differences* 103 (2023), 102274. https://doi.org/10.1016/j.lindif.2023.102274

[11] Elie F Kfoury, Samia Choueiri, Ali Mazloum, Ali AlSabeh, Jose Gomez, and Jorge Crichigno. 2024. A comprehensive survey on smartnics: Architectures, development models, applications, and research directions. *IEEE Access* 12 (2024), 107297–107336.

[12] Elie F Kfoury, Jorge Crichigno, and Elias Bou-Harb. 2021. An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends. *IEEE access* 9 (2021), 87094–87155.

[13] Kenneth R. Koedinger and Vincent Aleven. 2007. Exploring the Assistance Dilemma in Experiments with Cognitive Tutors. *Educational Psychology Review* 19, 3 (2007), 239–264. https://doi.org/10.1007/s10648-007-9049-0

[14] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets '10)*. Association for Computing Machinery, New York, NY, USA, 19:1–19:6. https://doi.org/10.1145/1868447.1868466

[15] 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. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[16] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389. https://doi.org/10.1561/1500000019

[17] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems*, Vol. 36. https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html

[18] Teaching on Testbeds. 2024. Teaching on Testbeds: Experiential Learning Materials for Computer Networks, Network Security, Cloud Computing, and Related Topics. (2024). https://teaching-on-testbeds.github.io/ Accessed: 2026-06-07.

[19] Kurt VanLehn. 2011. The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist* 46, 4 (2011), 197–221. https://doi.org/10.1080/00461520.2011.611369

[20] Changjie Wang, Mariano Scazzariello, Alireza Farshin, Simone Ferlin, Dejan Kostic, and Marco Chiesa. 2024. NetConfEval: Can LLMs Facilitate Network Configuration? *Proceedings of the ACM on Networking* 2, CoNEXT2 (2024), 1–25. https://doi.org/10.1145/3656296

[21] Zhihao Wang, Alessandro Cornacchia, Alessio Sacco, Franco Galante, Marco Canini, and Dingde Jiang. 2025. A Network Arena for Benchmarking AI Agents on Network Troubleshooting. (2025). arXiv:cs.NI/2512.16381 https://arxiv.org/abs/2512.16381

[22] David Wood, Jerome S. Bruner, and Gail Ross. 1976. The Role of Tutoring in Problem Solving. *Journal of Child Psychology and Psychiatry* 17, 2 (1976), 89–100. https://doi.org/10.1111/j.1469-7610.1976.tb00381.x

[23] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. *arXiv preprint arXiv:2309.07597* (2023).

[24] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*. https://openreview.net/forum?id=WE_vluYUL-X

A ARTIFACT AVAILABILITY

An anonymized artifact is available at: https://tinyurl.com/ai-network-edu-agent.

The artifact includes the agent source code, structured lab documentation, the 100-question evaluation dataset, evaluation scripts, and anonymized result files. It excludes credentials, raw runtime logs, author-identifying metadata, deployment platform identifiers, and the original lab-manual PDF because the manual contains identifying front matter and metadata.

The artifact contains five main parts: (i) agent source code, (ii) evaluation scripts, (iii) structured documentation, (iv) results, and (v) a sanitized configuration template. The retrieval evaluation can be run without an LLM credential or a lab VM:

```
python -m venv .venv
source .venv/bin/activate
pip install -e .
python eval/eval_retrieval_modes.py
```

This script reports Hit@5 and MRR for BM25, dense retrieval, and hybrid retrieval. The anonymous artifact runs over the included structured documentation. The paper's retrieval result file is included under the artifact's results directory. Exact full-corpus reproduction requires restoring the omitted manual PDF after review.

The answer-accuracy evaluation compares three conditions: LLM-only, document-only RAG, and the full agentic system. It requires an LLM credential configured in the artifact environment file. The full agentic condition also requires a running lab VM for live-state tools. The commands are documented in the artifact README.

Generated answers are graded with a rubric-based LLM judge and then manually inspected. The human-audited labels and summary used in the paper are included in the artifact's results directory. Live-state tool checks can be run when a lab VM is available.