SLMs Meet GraphRAG: A Structured Approach to Context-Aware Cybersecurity Hint Generation

Ishan Abraham[1], Jens Mache[1], Taylor Wolff[2], Jack Cook[2], Richard Weiss[2] and Justin Wang[3]
[1]Lewis & Clark College, Portland, Oregon, USA
[2]The Evergreen State College, Olympia, Washington, USA
[3]Northeastern University, Boston, Massachusetts, USA

ishanabraham@lclark.edu
jmache@lclark.edu
taylor.wolff@evergreen.edu
cookjackc@gmail.com
weissr@evergreen.edu
hs.wang@northeastern.edu

Abstract: Generating hints for learners who are engaged in hands-on cybersecurity exercises is the goal of our research. Learners sometimes get stuck or frustrated, they head in the wrong direction or are missing information that is necessary for solving an exercise. While using large language models (LLMs) is an option, LLMs typically require the sharing of student data with third-party AI providers. In order to improve privacy and minimize cost and computational overhead, previous research has explored using locally deployed small language models (SLMs) with retrieval-augmented generation (RAG). However, while RAG has been shown to enhance SLM capabilities without the need to fine tune, it falls short when answering open-ended or multi-step questions that require reasoning across interconnected concepts. This limitation is particularly evident in cybersecurity education, where students often need help understanding how threats, tools, and strategies relate to one another. The cybersecurity hint system EDUHints (Wolff et al, 2025) currently relies on a standard RAG pipeline. In classroom testing, students were unsure whether generated hints meaningfully answered their questions. To address this challenge, we present a custom GraphRAG approach that builds on a proposed cybersecurity education focused ontology and knowledge graph called AISecKG[1]. We extend the ontology to let us incorporate natural language-to-bash command mappings, a valuable feature as students tend to ask questions regarding command-line use. Graph data is extracted using multiple methods and semantically scored to prioritize only the most relevant results. Our pipeline currently employs Microsoft's Phi-3-mini-4k-instruct SLM, integrates LangChain for modular orchestration, and uses Neo4j as the graph database. We survey cybersecurity instructors to rate responses generated by the EDUHints and our GraphRAG system. Results show that hints generated using a GraphRAG are preferred almost three times more by cybersecurity instructors. This suggests that an SLM's educational hint generation abilities can be improved through our GraphRAG architecture.

Keywords: Cybersecurity education, Small language model (SLM), Retrieval-Augmented generation (RAG), Knowledge graph, Human-in-the-Loop

1. Introduction

The rise of LLM chatbots within the educational sphere comes with good reason. These bots offer a great resource for students to receive feedback at times when instructors are not present. However language models may need additional topic and contextual information as they may not have been trained extensively on domain specific data. To mitigate this, researchers and educators have experimented with Retrieval-Augmented Generation[2] (RAG), which allows models to access relevant external information improving answer results. It does this by creating a system where embedded data can be retrieved efficiently and then injected into the model's prompt (Lewis et al, 2020). This serves as a cost effective and scalable way to give the model access to information it was not trained on. RAG as well offers a reliable solution to bolstering the performance of small language models (SLMs) within an educational setting (Liu et al, 2024 and Yu et al, 2025). SLMs also have the benefit of not requiring GPU's and can be run in a fully containerized environment.

With EDURange, a cybersecurity education platform, students learn the fundamentals of command-line cybersecurity while participating in live exercises, letting them apply concepts learned in the classroom to real

[1]Agrawal, G. et al (2023) "AISecKG: Knowledge Graph Dataset for Cybersecurity Education", Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE), https://par.nsf.gov/servlets/purl/10401616

[2]Lewis, P. et al (2020) "Retrieval-augmented generation for knowledge-intensive NLP tasks", Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS), pp 9459–9474, https://dl.acm.org/doi/abs/10.5555/3495724.3496517

applications (Weiss et al, 2017). Previously we introduced EDUHints[3], a human-in-the-loop hint generation system for EDURange, to speed up the instructor feedback process. EDUHints is a system for the instructor that can generate a hint based on a student's most recent bash commands, their questions to instructors, and knowledge check up responses. Instructors also have the ability to manually edit hints before sending them to students ensuring quality and relevancy of hints.

The EDUHint system has thus far been evaluated via three classroom tests, garnering mixed results from students surveyed on the perceived helpfulness and the quality of hints they received (Wolff et al, 2025). These results indicated a clear need for improvements to the system's hint generation process. One such area of improvement that we explore in this work is implementing a more sophisticated retrieval-augmented generation system to improve the capabilities of our SLM.

GraphRAG[4] (Edge et al, 2024) has been shown to be a promising alternative to RAG which allows for our model to understand not just the cybersecurity data, but the relationships between them as well. This tackles a known issue in RAG which is its inability to handle open-ended or multi-step questions (Harjono, 2025). In EDURange, students are required to carry out complex tasks involving knowledge over multiple concepts. Our goal with utilizing a GraphRAG is to accurately convey these relations, providing students more accurate and educational hints.

This paper's main contributions are as follows:

- **Graph-Enhanced Retrieval for SLMs**: We demonstrate that small language models (SLMs) can benefit from a graph-augmented retrieval architecture. By leveraging structured knowledge graphs, our system compensates for the limited reasoning and domain knowledge capabilities typical of SLMs, particularly in cybersecurity education.
- **A Custom GraphRAG Pipeline for Cybersecurity Education**: We design and implement a modular GraphRAG pipeline tailored to EDURange, a real-world cybersecurity learning platform. Our system integrates Phi-3-mini-4k-instruct, Neo4j, and LangChain, and includes custom mechanisms for entity extraction, multi-modal graph querying, vector-based path scoring, and context formatting.

2. Related Work

Recent work highlights that traditional RAG struggles with multi-step reasoning and questions aimed at a broad scope of knowledge. Edge et al. (2024) discussed how RAG only retrieves locally relevant passages without capturing relationships among concepts. Edge also proposes GraphRAG, a graph-based approach that leverages LLM powered community-level summaries and entity-relationship style data. This strategy outperformed standard vector RAG on queries focused on a large corpus of data. Within the education domain, RAG has been shown in tandem with Small Language Models (SLMs) showing promise as a question answering chatbot for introductory computer science (Liu et al, 2024 and Yu et al, 2025) and as a way to improve an SLMs hint generation capabilities in cybersecurity (Wolff et al, 2025). SLMs are used as opposed to LLMs in order to gain control over privacy, cost, as well as local deployment, however their lack of general knowledge leads researchers to find ways to bolster up their domain specific capabilities.

To bring more structured context into retrieval, several groups have focused on domain specific knowledge graphs. Agrawal et al (2023) propose AISecKG, a defined ontology, knowledge graph, and custom entity classification model for the cybersecurity education field. The ontology describes how cybersecurity data can be classified into roles, applications, or concepts as well as further subsections within them. This form of graph centric data enables reasoning entities and their relationships.

Evaluations of RAG-style systems remains an open question. A prominent metric, Ragas[5] (Es et al, 2024) scores how retrieved passages contribute to generated outputs. While this method works well to determine the effectiveness of your knowledge base data, when it comes to educational hints, automated frameworks such as

[3]Wolff, T. et al (2025) "EDUHints: A Human-in-the-Loop Small Language Model System for the Generation of Cybersecurity Hints", Proceedings of the 24th European Conference on Cyber Warfare and Security (ECCWS), https://papers.academic-conferences.org/index.php/eccws/article/view/3659/3360

[4]Edge, D. et al (2024) "From Local to Global: A Graph RAG Approach to Query-Focused Summarization", ArXiv, https://arxiv.org/abs/2404.16130

[5]Es, S. et al (2024) "Ragas: Automated Evaluation of Retrieval Augmented Generation", Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, https://arxiv.org/pdf/2309.15217

this fail to examine the usefulness of a generated response. Similar to past work (Wolff et al, 2025), we survey instructors directly on the preference of hints generated rather than consulting frameworks or relying on LLMs to ensure accurate assessment of the quality of hints.

3. Data/Knowledge Graph

We used an existing cybersecurity education ontology and knowledge graph AISecKG (Agrawal et al, 2023) as the basis for our design. The ontology describes three main categories in which all cybersecurity concepts are categorized in: application, concept, and role. Within each category there are subcategories (12 total) which further define what the data is. The data was created based on 6 upper level cybersecurity lab manuals which outlined many important tools, frameworks, concepts, and how they are used. Data is stored in entity relationship triples making it ideal for a graph database.

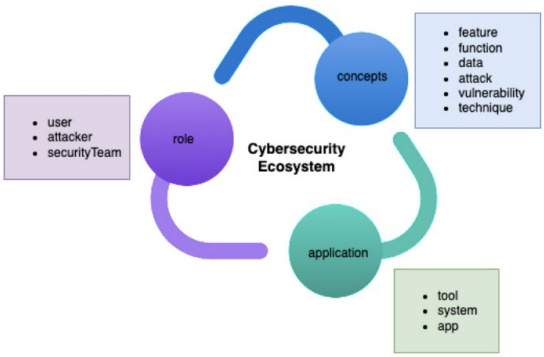

Figure 1: Ontology from AISecKG [Agrawal et al.]

An important piece of data we supplant this ontology with is a newly created natural language to bash commands dataset, NL2SH-ALFA[6] (Westenfelder et al, 2025) . We extend our ontology by adding a category called nl_bash which has two types of data, natural language and bash commands, as well as the relation has_command. We do this to match the triples format to keep data consistent with the rest of the ontology and also to define the connection between the natural language and bash. In total our graph has about 80,100 nodes and 41,200 relationships, with 79,000 of the nodes being either natural language or bash commands. All data has a label, type, category, as well as a vector embedding. We embed all nodes and relationships using the all-MiniLM-L6-v2 embedding model from the sentence transformers library. In the future we hope to improve this by mapping new data to the AISeckKG ontology since the grand majority of our current database is natural language to bash and because we would like more EDURange specific data to be available to our system.

4. Setup/ Implementation

LangChain is an open source framework that allows developers to build applications utilizing all types of language models. LangChain's built-in compatibility with graph databases such as Neo4j, vector databases, and llama-cpp makes it a practical choice for our system. Although LangChain has built in methods for creating a GraphRAG system we found that creating a custom chain was significantly more effective in querying optimal graph data.

[6]https://huggingface.co/datasets/westenfelder/NL2SH-ALFA

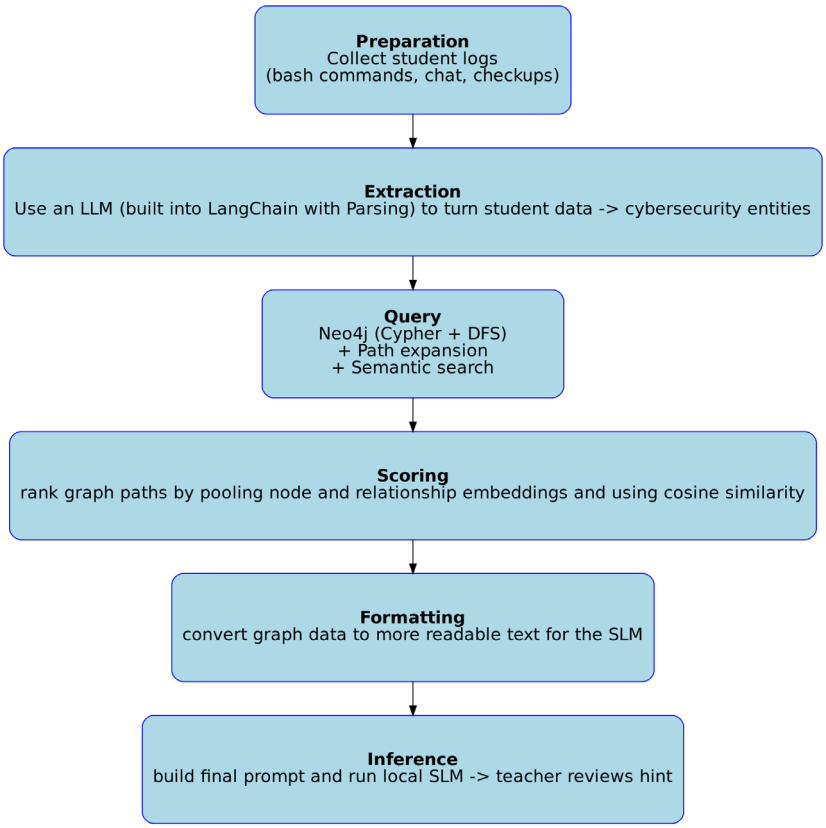

Figure 2: GraphRAG System Design

4.1 Preparation

Unlike question answering systems, the industry standard for educational chatbots, EDUHints and our GraphRAG system work semi-autonomously to collect a student's recent history and use this to generate a hint for the teacher to then review. This human-in-the-loop approach allows the incorporation of artificial intelligence while preserving organic student-instructor interaction. In our preparation step we parse student logs generated by EDURange and store student bash commands, chat questions, and the scenario's check up answers in a dictionary.

4.2 Extraction

Just like other GraphRAG systems, we use an LLM (gpt-3.5-turbo) to process our input and break it into its important cybersecurity relevant entities. We ensure correct formatting by using LangChain's built in PydanticOutputParser and defining a schema for an output, guaranteeing consistency with the LLM output. For example if a student asked the question "What does a firewall do in a network?", the output of our extraction would be ['firewall', 'network'] since these are the relevant cybersecurity terms. We also append the full question as an entity after extraction because much of our natural language to bash command data is in sentence forms.

4.3 Query

We use three separate strategies for obtaining data from our graph while tracking which algorithm was used to query each data point, letting us observe the impacts of each method.

1. **Initial Connection (IC):** We attempt to locate direct paths between two entities in our list via a standard depth first search graph traversal, allowing a max depth of three. IC generally resulted in the fewest amount of queried data for any particular student log. This makes sense as this algorithm requires a connection (of at most three "hops" away) between at least two entities. This strict requirement can often not return any data at all. However, data points found from this method often scored *significantly higher* than the other algorithms as by its nature these data points are most rich with relevant information. We choose this algorithm first to ensure any data captured here is always scored as we limit the total amount of queried data to 20.

2. **Neighborhood Expansion:** Store a two depth "neighbourhood" of each entity. This had shown to consistently output relevant data points, often scoring highest when a learner asks questions about concepts within cybersecurity but Initial Connection returned little to no data. This information provides our SLM with contextual information about the specific concepts a learner is asking about.
3. **Semantic Search:** Since we already have node and relationship embeddings stored in our graph, we embed each entity (which includes the full questions) and use sklearn's NearestNeighbors function to find the top 5 closest matching nodes in our graph to EACH entity we extracted. We then run Neighborhood Expansion on this set of nodes. This approach was shown to excel when students asked questions about bash commands, very likely due to our large amount of natural language to bash command data mappings.

This tri-query system ultimately aims to **minimize retrieval bias**, allowing us to capture the most relevant and useful data that a particular algorithm may miss. However, since the methods run independently, we also must explicitly deduplicate to ensure data is unique.

4.4 Scoring

We limit our querying methods to cap at 20 data points per method, more than enough data for most queries. After combining and deduplicating data we implement a method of ranking these data points so that the most relevant graph data shows up first. We utilize our already existing vector embeddings for all nodes and relationships by taking the mean of all node and relationship embeddings within a path, resulting in a vector which semantically represents a full data point. We can then compare these pooled embeddings to a learner's chat questions via cosine similarity to find the closest matching data points. This showed to be incredibly effective at identifying which graph data was the most important in answering a learner's question. Since our small language model was tested to be volatile with longer context windows we only choose the top 4 highest scoring data points. One notable edge case is the situation where a student has not asked any questions yet chooses to generate a hint. In this case, we skip scoring altogether as in order for our scoring mechanism to work, we require at least one question in order to have a comparison for our queried graph data.

4.5 Formatting

We organize graph data into an ordered list with the most relevant data point at the top. We as well format our data to be significantly more context window friendly while maintaining all information. This is done by converting Neo4j's dictionary data points into a more human readable string format.

4.6 Inference

Once we have our formatted data we can build our final prompt which includes student bash commands, chat history, check up answers, as well as the graph data. Our system directly mimics the prompt engineering of EDUHints to ensure consistency with results. Once the prompt has been built, hint generation can be run locally.

Live Updates

Optionally, we have the ability to classify entities on the fly if they are new to the system, which is done before querying our graph database. We accomplished this by using few-shot prompting with our small language model to classify an entity into our ontology by having it group the entity into its type and subtypes. This process showed promise but when compared to the speed and consistency of using a third party large model such as Openai's gpt-3.5.turbo, it was clear more work would be needed to make this more reliable within our system. We discuss limitations of SLMs in this manner further within section 7.1. When evaluating the performance of the GraphRAG we turn off the option to run live updates.

5. Evaluations/Metrics

We evaluated the two systems, EDUHints and GraphRAG by generating hints for the same set of 25 student logs randomly sampled from classroom data. Each log contained the student's recent bash commands, 3 most recent questions asked, and knowledge-check responses. Knowledge-check responses are the answers to practice questions throughout the exercise. For each log, we produced two hints: one with EDUHints and one with GraphRAG. These paired hints were presented blindly and in random order to a panel of four cybersecurity instructors to minimize bias toward either system.

Each instructor was asked to vote for the hint that they believed was more educationally useful for the student. If both hints were deemed equally good, instructors could select "Similar." In total, this process yielded 100 individual votes (25 logs × 4 voters).

To better understand system performance, we aggregated the votes in two ways:

- **Votes:** the raw number of instructor votes for each system across all 25 logs.
- **Wins:** the number of logs for which a system received the majority of votes (i.e., at least 3 out of 4 votes for that log). If neither system held a majority or if "Similar" was the majority choice, that log was counted as a win for Similar.

6. Results

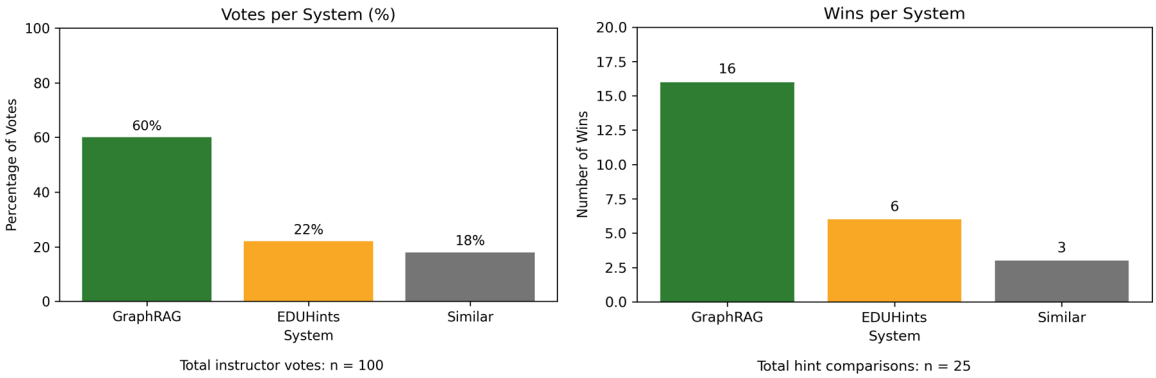

Figure 3: GraphRAG vs EDUHints Results: Figure 3 compares instructor preferences across EDUHints, GraphRAG, and instances where both systems were judged similarly

These results show that our GraphRAG architecture consistently outperformed EDUHints. In both the votes per system and wins per system, a majority of instructors preferred GraphRAG generated hints. While our system was shown to produce more beneficial hints for students, the data there are still areas for improvement.

6.1 Examples

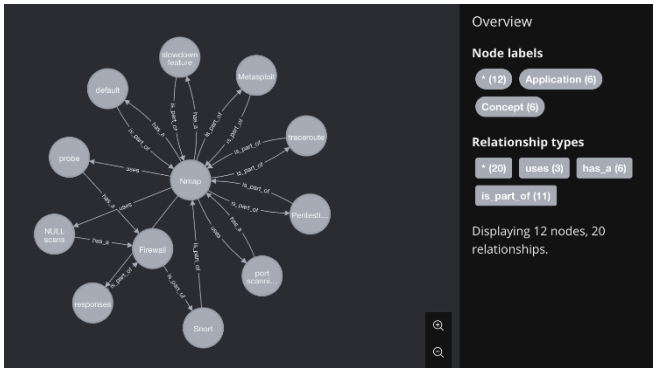

Figure 4: Graphical Output within Neo4j

```
MATCH path = (n)-[*1..2]->(m)
    WHERE
    n.name in ["Firewall", "Nmap"]
    AND
    m.name in ["Firewall", "Nmap"]
RETURN DISTINCT path
LIMIT 10
```

Figure 5: Example Cypher query

Above and to the right, we display an example and a graphical output of a simple Cypher query which finds all distinct paths where either the first or last entity within 2 hops is named "Firewall" or "Nmap".

Below, we show an example of graph data that is injected into our small models context along with a display of which algorithm resulted in that point of data. In this example the student's recent questions asked about viewing a text file and using the ls command.

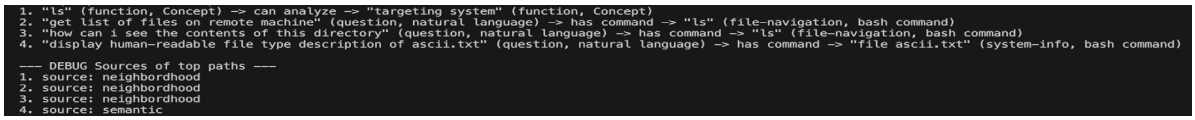

Figure 6: Formatted graph data and sources display

7. Conclusion and Future Work

In this paper we've demonstrated how SLMs with GraphRAG can substantially enhance the quality of hints in cybersecurity education. By leveraging domain specific knowledge graphs, multi-step querying, and semantic scoring, we show that even general purpose lightweight models can learn fundamental concepts of domain specific knowledge. These results underscore the potential for graph-centric retrieval methods that are scalable, privacy-focused, and close the gap between SLMs and LLMs in technical domains.

7.1 Limitations

While instructors clearly favored GraphRAG produced hints, we acknowledge that the small sample size of 25 student logs and 4 cybersecurity instructors limits the overall statistical of our conclusions. Our evaluation strategy as well does not measure the tangible usefulness of hints, rather it aims to measure the perceived usefulness. A possible solution would be to have instructors assess hints, track lesson statistics, and also survey or live prompt students to see which hints they prefer.

Our GraphRAG system is also not entirely local. We utilize a third party LLM to extract cybersecurity relevant entities from student questions or the data we give it. While internal testing was done to use our SLM for entity extraction, this process turned out to be unreliable, resulting in either a reduced amount or irrelevant set of graph data. Right now this was identified as a technical point of limitation but as SLMs continue to improve in the realm of cybersecurity, it is reasonable to suspect the use of third party LLM's will no longer be necessary.

Another limitation we observed was the timeliness in generating hints for the GraphRAG system. Despite practically identical prompt structures for both the EDUHints and GraphRAG, the additional four points of graph data resulted in a significant increase in hint generation time (~10x compared to the EDUHints system). This is likely because the increased context introduced a layer of complexity that the SLM could not handle efficiently. To verify this hypothesis, we as well independently measured the time to query the graph database to identify if this was a bottleneck in our system. Over the 25 hints generated, we observed that querying our database took an average of 3.2 seconds with little deviation, meaning this was not the root cause of the extended time to generate a hint.

7.2 Future Work

In the future we plan to research the following areas:

- **Expanding the instructor-survey study** with a larger panel and more varied student logs to strengthen reliability.
- **Examining cost savings** relative to time saved: Do generated hints result in tangible benefits for teachers, such as time savings .
- **Investigating student-learning metrics** (task-completion time, error-rate reduction) to complement instructor preferences.
- **Refining entity extraction using SLMs**: explore prompt-engineering and guardrails to eventually replace the LLM dependency while maintaining accuracy.
- **Broadening our data**: integrate instructor-authored slides, scenario-specific artifacts, and future EDURange exercises for richer context.

In summary, this work has demonstrated the value of combining SLMs with graph-enhanced retrieval to improve domain-specific hint generation. The progress so far shows that these tools can be made both accessible and effective, and the next phase of research will aim to make them more pedagogically aligned, better validated, and truly end-to-end local, broadening their impact in cybersecurity education and potentially in other technical learning domains

Acknowledgements

This work was partially supported by the National Science Foundation under Awards 2216485 and 2216492.

AI Declaration: AI tools were used to generate the contents of some prototypal scenario context files, to assist in the detailing of figures, and minimal support with code.

Ethics Declaration: The Institutional Review Board of Lewis & Clark College has determined that this project is exempt under exemption code 45 CFR 46.104(d)(1) and will not require further review or approval from the committee.

References

Agrawal, G. et al (2023) "AISecKG: Knowledge Graph Dataset for Cybersecurity Education", Proceedings of the AAAI 2023 Spring Symposium on Challenges Requiring the Combination of Machine Learning and Knowledge Engineering (AAAI-MAKE), https://par.nsf.gov/servlets/purl/10401616

Chetri, C. (2024) "Exploring Large Language Model-Powered Pedagogical Approaches to Cybersecurity Education" Arxiv, 163-66, https://dl.acm.org/doi/pdf/10.1145/3686852.3686887

Edge, D. et al (2024) "From Local to Global: A Graph RAG Approach to Query-Focused Summarization", ArXiv, https://arxiv.org/abs/2404.16130

Es, S. et al (2024) "Ragas: Automated Evaluation of Retrieval Augmented Generation", Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, https://arxiv.org/pdf/2309.15217

Harjono, K. (2025) "From Embeddings to Entities: A Comparative Analysis of RAG Architectures in Academic Domains", Thesis, https://open.library.ubc.ca/media/stream/pdf/52966/1.0448869/5

Lewis, P. et al (2020) "Retrieval-augmented generation for knowledge-intensive NLP tasks", Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS), pp 9459–9474, https://dl.acm.org/doi/abs/10.5555/3495724.3496517

Liffiton, M. et al (2023) "CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes", Proceedings of the 23rd Koli Calling International Conference on Computing Education Research (Koli Calling), pp 1–11, https://doi.org/10.1145/3631802.3631830

Liu, S. et al (2024) "Can Small Language Models With Retrieval-Augmented Generation Replace Large Language Models When Learning Computer Science?", Proceedings Innovation and Technology in Computer Science Education (ITiCSE), pp 388–393, https://doi.org/10.1145/3649217.3653554

Lu, Z. (2025) "Small Language Models: Survey, Measurements, and Insights", https://arxiv.org/abs/2409.15790

Pasquale, R. D. and Represa, S. (2024) "Empowering Domain-Specific Language Models with Graph-Oriented Databases: A Paradigm Shift in Performance and Model Maintenance", https://arxiv.org/pdf/2410.03867

Sahoo, P. et al (2021) "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications", https://arxiv.org/pdf/2402.07927

Sarmah, B. at al (2024) "HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction", Proceedings of the 5th ACM International Conference on AI in Finance (ICAIF), https://doi.org/10.1145/3677052.3698671

Westenfelder, F. et al (2025) "LLM-Supported Natural Language to Bash Translation", Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, https://doi.org/10.18653/v1/2025.naacl-long.555

Wolff, T. et al (2025) "EDUHints: A Human-in-the-Loop Small Language Model System for the Generation of Cybersecurity Hints", Proceedings of the 24th European Conference on Cyber Warfare and Security (ECCWS)