Benchmarking RAG Models in Undergraduate CS Education

Zhixuan Chen
The University of Michigan
Ann Arbor, United States
shczx@umich.edu

Téa Hajratwala
The University of Michigan
Ann Arbor, United States
hajratwt@umich.edu

Junsen Huang
The University of Michigan
Ann Arbor, United States
junsen@umich.edu

Naihe Xiao
The University of Michigan
Ann Arbor, United States
naihe@umich.edu

1 Introduction

1.1 Motivation and Problem Statement

The student-to-instructor ratio in EECS 280, the introductory programming and data structures course at the University of Michigan, is approximately 32 to 1 – more than double the rest of the university, at 15 to 1. However, lowering this ratio is not easy. Hiring more instructors and TAs is costly, making this approach difficult to scale for large courses. As a result of this higher student-to-instructor ratio, students may be drawn to sources of help other than instructors. One popular choice is ChatGPT, a Large Language Model (LLM) which can answer questions on any topic in a matter of seconds. The problem with LLMs such as ChatGPT is that they have a tendency confidently state falsehoods as fact (hallucinate). A student using ChatGPT may take its response at face value, especially while they are learning new material. These hallucinations confuse students at their best and imbue them with a false sense of confidence at their worst. Retrieval-Augmented Generation (RAG) is a popular approach to reduce hallucinations in existing LLMs. Our project aims to find out whether RAG is useful in a computer science classroom setting by benchmarking RAG models against EECS 280 course document Q&A. We hope this study can provide valuable insight on future development of an EECS 280 chatbot.[1]

1.2 Background

RAG utilizes external data, such as a textbook or course notes, to provide the model with relevant background information, reducing the chance that a hallucination occurs. The traditional RAG pipeline has four general steps [4]:

1. **Pre-Retrieval**. Involves breaking down source documents into vectors, typically using a process called indexing. The vectors are stored into a database.
2. **Retrieval**. Involves searching through the database and ranking the documents in terms of relevance to the user's query.
3. **Post-Retrieval**. Involves filtering out low-quality or irrelevant documents, and re-ranking documents to find the best match to the query.
4. **Generation**. Involves integrating the documents retrieved into the original query as additional information, then calls the API of the LLM using the modified query to produce an accurate response.

[1]This section is modified. We shifted project task under guidance from Prof. Bondi-Kelly and GSI James Boggs

1.3 Overview of Proposed Work

In our project, we will focus on evaluating the performance of two graph-based RAG methods (i.e., LightRAG[2] and HippoRAG [3]) when they are asked to answer questions related to EECS 280 projects. LightRAG enhances RAG systems by integrating graph-based indexing with dual-level retrieval. It constructs a knowledge graph (KG) from text chunks, where nodes (entities) and edges (relationships) are enriched with key-value summaries. A vector search first retrieves relevant nodes, followed by graph traversal to expand context. HippoRAG also leverages KGs to improve retrieval accuracy. External documents are converted into KGs, and a customized page rank algorithm is applied to rank and retrieve the most relevant content. Besides LightRAG and HippoRAG, we will also examine the naive GPT model without any RAG enhancement as our baseline.

We will use the project specifications on the EECS 280 website as the knowledge base and prompt the RAGs with real questions found on EECS 280's Piazza. We will manually rate the responses from all the methods and then conduct a comparative analysis of the results.[2]

2 Related Work

2.1 RAG Methods

2.1.1 Model: LightRAG [2]. LightRAG is a RAG framework that features graph-based indexing and a dual-level retrieval paradigm. Given that traditional RAG systems use flat data representations in the indexing stage, their ability to capture complex interdependencies between information is always restricted. To mitigate this limitation, LightRAG uses an LLM to extract entities and relationships from the external data source and then constructs a knowledge graph where nodes represent entities and edges capture their connections. In addition, all nodes and edges in the graph are enriched with key-value pairs (e.g., summaries or descriptions) for efficient retrieval. When a query arrives, LightRAG employs a two-tier retrieval strategy: low-level retrieval targets specific entities and relationships, while high-level retrieval addresses broader themes related to the query. For both retrieval strategies, the system first extracts keywords and then matches the corresponding entities and relationships in the graph by comparing their vector similarities. After that, LightRAG will perform a graph traversal to retrieve all the key-value pairs for each node and edge traversed, which are

[2]This section is modified to accomodate new LightRAG model we use.

used as the contextual information of the query. Experiments on several datasets demonstrate its superiority in generating diverse and contextually rich responses. However, it is unknown whether LightRAG can be generalized to other domains, because it is only evaluated on datasets with limited domains.[3]

2.1.2 Model: HippoRAG. Gutiérrez et al. [3] proposed a RAG system called HippoRAG, which was inspired by how the neocortex and the hippocampus interact in human brains to store and retrieve long-term memory. HippoRAG consists of two key components: 1) Offline indexing, and 2) Online retrieval. Offline indexing is the stage of constructing a knowledge graph from the knowledge base, which is similar in LightRAG. In online retrieval, the LLM first extracts named entities from the query, which are linked to nodes in the knowledge graph. Then, the system will run a personalized page ranking algorithm which performs a graph search starting from the query nodes and distributes the probabilities to related nodes in the graph. Finally, the rank score of each passage will be computed by aggregating corresponding node probabilities, and the passages with high rank scores will be provided to the LLM as the context of the query. HippoRAG is evalauted on several multi-hop QA benchmarks where the answers need to be collected from different documents. The results show that HippoRAG outperforms previous methods on both accuracy and efficiency, especially for the path-finding questions. However, it is also mentioned that the named entity recognition sometimes leads to errors and further validation is needed to ensure its scalability as the size of the knowledge graph grows and the knowledge fields shift.

2.1.3 Weakness Addressing. In our work, we will evaluate how the known weaknesses of LightRAG and HippoRAG manifest in the context of EECS 280. By benchmarking both models on real classroom data, we can assess whether these limitations significantly impact response quality in an educational setting—something prior work has not explored. [4]

2.2 RAG Evaluation

Chen et al. [1] put forward four fundamental abilities required for effective RAG:

- **Noise Robustness:** The ability of LLMs to extract useful information from noisy documents.
- **Negative Rejection:** The ability of LLMs to reject answering questions when the required knowledge is not present in any retrieved document.
- **Information Integration:** The ability of LLMs to answer complex questions that require integrating information from multiple documents.
- **Counterfactual Robustness:** The ability of LLMs to identify and handle factual errors in retrieved documents.

To evaluate LLMs upon these abilities, Chen et al. [1] created a benchmark called Retrieval-Augmented Generation Benchmark (RGB), which was constructed using the latest news articles and external documents retrieved from search engines. They then evaluated six LLMs with this benchmark. The evaluation results reveal that while LLMs exhibit noise robustness to some extent, they struggle significantly with the other three abilities. However, they only evaluated RAG systems without enhancement methods, meaning that the external documents are just stored in a vector database for retrieval purpose. We would expect that methods like LightRAG or HippoRAG can boost the performance, which is already demonstrated in the results of the two methods.

[3]This subsection is newly added because we changed one of the RAG systems that we originally planned to use into LightRAG.
[4]This subsection is newly added to address previous critics.

3 Methodology and Evaluation

3.1 Data Preparation

3.1.1 Q&A dataset. We begin by constructing a Q&A dataset from EECS 280 EdStem Q&A in a structured format of question and answer.

3.1.2 Course Related Documents. We will also need related documents with regards to the question. In this case it will be course related materials from EECS 280. The document will need to be in Markdown format for better processing.

3.2 Model [2][3]

We select HippoRAG and LightRAG to be the two RAG models that we will be benchmarking. These two models both uses Knowledge Graph, which should be more effective with structured documents like EECS 280 course notes and project specs. We expect this to improve accuracy in cases where understanding depends on connecting different concepts. GPT-4o-mini is used across all models for its affordability and fast response.
Both model include API for easier programming. We utilize the HippoRAG API and LightRAG API to:

- Index course specifications.
- Parse and extract questions from the dataset QA dataset.
- Submit these questions to the RAG pipeline and collect the generated answers.

The outputs are stored in a structured format for further evaluation. We include a plain GPT-4o-mini without RAG to see if RAG actually improves performance. For this, we simply provide the model with the document and the question, and record its response.

3.3 Evaluation and Benchmarking [1]

To evaluate both LLM-only and RAG-enhanced answers, we adopt three key criteria: **1. Noise Robustness:** Does the answer refer to course specific information out of long and noisy course materials? **2. Negative Rejection:** Does the model refuse to answer if student's question have no answer in the provided materials? **3. Information Integration:** Does the answer works with knowledge on different parts of the document? These criteria were chosen to reflect real-world student needs in complex CS courses where hallucinations is unacceptable. We also use the staff response for each question to compare with each model's output. Each LLM response is manually evaluated by a course staff member. Answers are scored on a 3-point scale:

- **1 – Poor**: Incorrect or irrelevant response.
- **2 – Fair**: Partially correct.
- **3 – Excellent**: Comparable to instructor feedback.

We then aggregate performance by counting the number of responses in each score category per model.

4 Preliminary Results

4.0.1 Experiment Details. We selected the following datasets for our preliminary experiment:

- Q&A Dataset: EECS 280 EdStem Q&A on Project 3 from Spring 2024
- Course Document: EECS 280 Project 3 Specification

The Q&A dataset consists of 21 student questions and corresponding instructor answers. These questions are focused on a narrow topic (Project 3), making it easier to evaluate whether RAG improves the model's ability to retrieve relevant information from the specification. Additionally, Project 3 requires substantial domain knowledge (i.e., Euchre rules specific to this EECS 280 project), making it a suitable test case for evaluating RAG effectiveness. Instructor answers serve as the benchmark (ground truth) for rating. The dataset size (21 Q&A pairs) is manageable for manual evaluation, yet large enough to yield meaningful preliminary insights. We submitted the same set of questions to the following three models:

(1) **GPT-4o-mini (No RAG)**: We prompt the model as follows: ""You are an Instructional Aide (IA) for EECS 280 at the University of Michigan. You will help students by answering questions based on the project specs."
(2) **HippoRAG**: GPT-4o-mini as the LLM and Facebook's *Contriever* as the embedding model
(3) **LightRAG**: GPT-4o-mini as the LLM and LightRAG API *openai embed* as embedding model

We then compared the outputs of these models to the instructor-provided answers using the rating methodology described in the Methodology section.

4.0.2 Experimental Results.

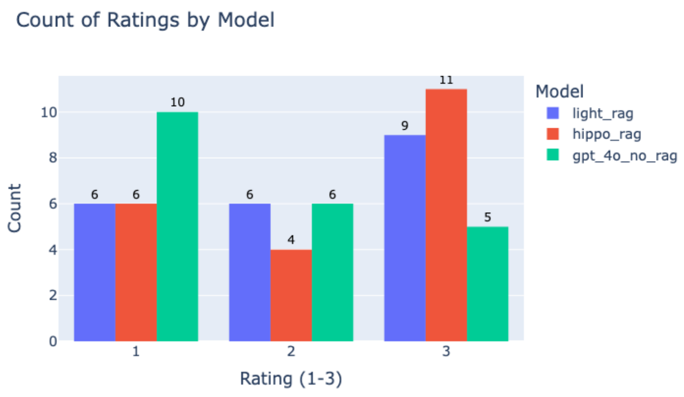

Figure 1: Histogram showing distribution of ratings (1, 2, 3) by model

4.0.3 Analysis. We analyze the performance of the three models based on the distribution of their output ratings across 21 questions.

- **Poor (score 1):** GPT-4o-mini received the most low ratings (10), suggesting difficulty and high error rate in generating domain-specific responses.
- **Fair (score 2):** The distribution of the scores between the three models was relatively balanced, indicating minimal difference in the borderline outputs. This shows no signs of the RAG model's advantage.
- **Excellent (score 3):** Both HippoRAG (11) and LightRAG (9) outperformed GPT-4o-mini (5), showing that RAG methods are generally better in producing correct and high quality domain-specific responses.

GPT-4o-mini exhibited a downward trend in ratings, while the RAG-based models showed an upward trend. This suggests that RAG contributes positively to generating helpful answers in domain-specific contexts. Unfortunately, due to the size constraint of the preliminary results, it is unknown which model is more superior in generating EECS 280 answers because the current results are not statistically significant.

5 Future Milestones

The table below outlines our upcoming milestones from now until the end of the semester, denoted by weekly subtasks:

Week	Milestone(s)
3/31 - 4/6	Add UM's Maizey to evaluate for performance against other models
4/7 - 4/13	Add project starter code and/or solutions to knowledge base; Safeguard model's responses against revealing answers to students
4/14 - 4/21	Benchmark all four models against data from W25; Create and present presentation

Table 1: Future Milestones for the project

6 Conclusion

We evaluated LightRAG and HippoRAG against a baseline LLM using the Spring 2024 Q&A dataset. Surprisingly, implementing RAG showed no statistically significant improvement, likely due to the small sample size. To increase performance, we plan to add project starter code and solutions while ensuring solution security in the KG. We will also reevaluate our RAG models with a larger Winter 2025 dataset and test UM's Maizey as another potential RAG model. Our final goal is to select the best model for use in EECS 280 in Fall 2026.

Acknowledgments

We used OpenAI's ChatGPT (March 2025 version) to assist with grammar and latex during the writing of this paper.

References

[1] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2023. Benchmarking Large Language Models in Retrieval-Augmented Generation. arXiv:2309.01431 [cs.CL] https://arxiv.org/abs/2309.01431
[2] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. arXiv:2410.05779 [cs.IR] https://arxiv.org/abs/2410.05779
[3] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2025. HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. arXiv:2405.14831 [cs.CL] https://arxiv.org/abs/2405.14831

[4] Yizeheng Huang and Jimmy X. Huang. 2024. The Survey of Retrieval-Augmented Text Generation in Large Language Models. arXiv:2404.10981 [cs.CL] https://arxiv.org/pdf/2404.10981