LLM Ability to Answer University Student Questions in CS Trained Exclusively on Auto-Generated QA

Samantha Jaehnig University of Michigan sjaehnig@umich.edu
Yuxuan Jiang University of Michigan jyuxuan@umich.edu
Ann Stone University of Michigan stoneann@umich.edu
Boyuan Zheng University of Michigan boyuann@umich.edu

1 Introduction

The integration of AI into the classroom is inevitable. Using AI chatbots in the classroom has the potential to assist students around-the-clock, making help more accessible. Chatbots and LLMs, though, require immense training. It may be easy to obtain some data for classes that already have QA forums, however not all courses have semesters of student data to lean on to train an LLM. Moreover, gaps in data may lead to gaps in a chatbot's knowledge. Just because a student hasn't asked a specific type of question previously, doesn't mean a future student won't. The persistence of such knowledge gaps can lead to generation of hallucinative responses, as a result of an attempt to answer questions by an LLM with insufficient domain or contextual knowledge.

Our project aims to improve the training of LLMs for CS courses that may not have sufficient data on their own. The main questions we are looking to answer are: Can we use an LLM to generate QA based on a given project/assignment specification? Is that generated QA sufficient to train an LLM that can answer student questions with high accuracy?

2 Related Work

Education is a space that could benefit tremendously from advanced AI tutoring software and is saturated with constant developments. While one-on-one tutoring is provably more effective for students than traditional, larger classroom settings, it can never be feasibly executed by humans alone. AI can be integrated into a number of educational environments, to provide students with personalized feedback and instructors with teaching materials.

For example, some prior work surveys mechanisms for QA systems that generate a variety of question types (ie. fill-in-the-blank, multiple-choice) (Virani et al., 2023; Basu et al., 2023; Riza et al., 2023). Several works utilize the RAG framework to improve the tendency of LLMs to hallucinate information by grounding them in a specified knowledge base (Barron et al., 2024; Meyur et al., 2024). Other work, specifically in the realm of CS chat agents attempts tutoring with no-code responses (Kazemitabaar et al., 2024). The discoveries of these earlier works will inform our project as we base our exploration of the generative QA space in CS course assignment specifications, and tie QA to student questions and course projects.

3 Dataset

We are using Piazza data from previous semesters of EECS courses, focusing on the questions and answers exchanged. This data serves as our primary dataset, capturing a variety of interactions that provide insight into student and instructor engagement, and question types. We will use them for model training and testing.

4 Proposed Approach

To provide robust, adaptive QA support for courses with limited existing material, we propose a *two-step system.*

4.1 LM for Project Specification to QA Generation

This stage aims to develop an LM that generates a comprehensive set of QA pairs from project specifications. The synthetic data will provide a foundation for training the answering model with course-specific knowledge.

Step 1: Spec Parsing & Comprehension
The model parses the project specification document to extract essential details (ie. requirements, constraints). This ensures the dataset generated is a solid reflection of assignment content.

Step 2: Question Generation
Leveraging NLP techniques such as question generation using T5 or other generative transformers, the model generates mock student questions. These questions will be both high-level conceptually-focused (ie. *"What is the purpose of this assignment"*) and low-level technically-oriented (ie. *"How should I format the output?"*) to reflect a range of student question types.
Step 3: Answer Generation
For each question, the model generates an answer based on information in the project spec. The model may employ rule-based methods or fine-tuned prompts to guarantee that the answers remain concise, contextually accurate, and directly relevant to the questions posed.
Step 4: Evaluation and Filtering
To maintain dataset quality, the generated QA pairs undergo an evaluation and filtering process to eliminate redundancy and irrelevance. This may involve human-in-the-loop assessments or automated consistency checks. The final dataset will consist of a curated set of QA pairs, reflective of common student questions and clear, assignment-specific answers.

The synthetic QA dataset produced in this stage serves as the core material for training the answering model in Step 2.

4.2 Language Model for Question Answering

In Step 2, the synthetic QA dataset is utilized to develop a second system specifically for answering student questions. This model is intended to function as a virtual teaching assistant, providing accurate responses to student inquiries.

Step 1: Fine-tune on RAG
The QA model will be fine-tuned on the synthetic dataset produced in Step 1, allowing it to learn the language, context, and expectations specific to the course. This helps the model acquire the ability to accurately respond to questions relevant to course content, filling any gap in historical course data.
Step 2: Adapt to Question Variability
To handle variability in student phrasing, the model will be trained to recognize and respond to paraphrased questions that are conceptually similar to those in the dataset. We will use techniques such as paraphrasing recognition and embedding-based similarity, ensuring the model can interpret and accurately respond to different formulations of the same question.
Step 3: Evaluation & Iterative Refinement
The model's efficacy will be evaluated using real student questions sourced from platforms like Piazza (e.g., EECS280 Piazza data). This will provide a measure of the model's accuracy, with results informing further refinements to enhance robustness.
Step 4: Performance Benchmarking
The model's performance will be benchmarked against a baseline (LM without synthetic data training) to measure improvements in accuracy, relevance, and response depth, validating the QA data's effectiveness. This approach equips the answering model to handle diverse student inquiries accurately, even in courses with limited pre-existing QA data, providing a scalable solution for enhancing AI-driven course support.

5 Project Plan and Work Division

5.1 Phase 1: Implementation (Nov. 4-16)

Goal: Develop LLM1 for QA generation and LLM2 for student question answering.
Tasks: (1) *Design and Architecture*: Define key modules, I/O formats, and model architecture for each LLM. (2) *Integration and Testing*: Integrate modules, conduct initial testing, and finalize implementations.
Sam and Ann will work on LLM1, while *Yuxuan and Boyuan* will handle LLM2.

5.2 Phase 2: Training (Nov. 17-23)

Goal: Train and fine-tune both LLMs with Piazza data.
Tasks: (1) *Data Prep*: Gather and process Piazza data for training. (2) *Model Training*: Train both LLMs, monitor, and fine-tune.

5.3 Phase 3: Testing & Finalization (Nov. 24 - Dec. 1)

Goal: Comprehensive testing and reporting.
Tasks: (1) *Testing*: Evaluate models with real-world scenarios. (2) *Performance Analysis*: Document strengths and weaknesses. (3) *Reporting*: Prepare a final report and poster.

References

Ryan Barron, Ves Grantcharov, Selma Wanna, Maksim Eren, Manish Bhattarai, Nicholas Solovyev, George Tompkins, Charles Nicholas, Kim Rasmussen, Cynthia Matuszek, and Boian Alexandrov. 2024. Domain-specific retrieval-augmented generation using vector stores, knowledge graphs, and tensor factorization.

Rahul Basu, Dolasaha, Chiranjib Dutta, and Ananjan Maiti. 2023. *Automatic Transformation and Generation of Question Papers in Education with NLP Techniques*.

Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Henley, Paul Denny, Michelle Craig, and Tovi Grossman. 2024. Codeaid: Evaluating a classroom deployment of an llm-based programming assistant that balances student and educator needs. pages 1–20.

Rounak Meyur, Hung Phan, Sridevi Wagle, Jan Strube, Mahantesh Halappanavar, Sameera Horawalavithana, Anurag Acharya, and Sai Munikoti. 2024. Permitqa: A benchmark for retrieval augmented generation in wind siting and permitting domain.

Lala Septem Riza, Yahya Firdaus, Rosa Ariani Sukamto, Wahyudin, and Khyrina Airin Fariza Abu Samah. 2023. Automatic generation of short-answer questions in reading comprehension using nlp and knn. *Multimedia Tools Appl.*, 82(27):41913–41940.

Altaj Virani, Rakesh Yadav, Prachi Sonawane, and Smita Jawale. 2023. Automatic question answer generation using t5 and nlp. In *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*, pages 1667–1673.