Can Small Language Models With Retrieval-Augmented Generation Replace Large Language Models When Learning Computer Science?

Suqing Liu*
suqing.liu@mail.utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

Zezhu Yu*
zezhu.yu@mail.utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

Feiran Huang
philipp.huang@mail.utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

Yousef Bulbulia
yousef.bulbulia@mail.utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

Andreas Bergen
andi.bergen@utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

Michael Liut
michael.liut@utoronto.ca
University of Toronto Mississauga
Mississauga, Ontario, Canada

ABSTRACT

Leveraging Large Language Models (LLMs) for personalized learning and support is becoming a promising tool in computing education. AI Assistants can help students with programming, problem-solving, converse with them to clarify course content, explain error messages to help with debugging, and much more. However, using cloud-based LLMs poses risks around data security, privacy, but also control of the overarching system.

To address these concerns, we created a locally-stored Small Language Model (SLM) that leverages different Retrieval-Augmented Generation (RAG) methods to support computing students' learning. We compare one SLM (neural-chat-7b-v3 - fine-tuned version of Mistral-7B-v0.1) against two popular LLMs (gpt-3.5-turbo and gpt-4-32k) to see the viability for computing educators to use in their course(s).

We use conversations from a CS1 course (N = 1, 260), providing students with an AI Assistant (using gpt-3.5-turbo) to help them learn content and support problem-solving while completing their Python programming assignment. In total, we had 269 students use the AI Assistant, with a total of 1, 988 questions asked. Using this real conversational data, we re-ran student questions using our novel SLM (neural-chat-7b-v3 testing nine different RAG methods) and gpt-4-32k, then compared those results against the original gpt-3.5-turbo responses.

Our findings indicate that using an SLM with RAG can perform similarly, if not *better*, than LLMs. This shows that it is possible for computing educators to use SLMs (with RAG) in their course(s) as a tool for scalable learning, supporting content understanding and problem-solving needs, while employing their own policies on data privacy and security.

*Both authors equally contributed for first authorship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ITiCSE 2024, July 8–10, 2024, Milan, Italy

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0600-4/24/07
https://doi.org/10.1145/3649217.3653554

CCS CONCEPTS

• **Information systems** → **Users and interactive retrieval**; • **Computing methodologies** → **Natural language generation**; • **Social and professional topics** → **Computing education**; *CS1.*

KEYWORDS

Small Language Models, Retrieval Augmented Generation, Large Language Models, Intelligence Concentration, Conversational Agent, Personalized AI Agent, Locally Deployable AI, Intelligent Tutoring System, Intelligent Teaching Assistant, CS1, Computing Education

ACM Reference Format:
Suqing Liu, Zezhu Yu, Feiran Huang, Yousef Bulbulia, Andreas Bergen, and Michael Liut. 2024. Can Small Language Models With Retrieval-Augmented Generation Replace Large Language Models When Learning Computer Science?. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649217.3653554

1 INTRODUCTION

Artificial intelligence (AI), especially those based on large language models (LLMs), have been widely used as Intelligent Tutoring Systems and Teaching Assistants in the education field [21, 24]. However, using and deploying LLMs locally requires substantial computing resources, time, and cost [4]. Additionally, cloud-base models can pose risks around data security, privacy, and organizational policies.

This work investigates the practical aspects of deploying language models, emphasizing hardware efficiency, ease of deployment, and data privacy on consumer electronics. Our tests are conducted on consumer-grade devices using Nvidia graphics cards (8GB VRAM min.) and Apple Silicon (16GB RAM min.). Furthermore, we analyze SLMs' capabilities in language understanding and content generation, spotlighting the efficacy of our Information Concentration (IC) Framework within the Retrieval-Augmented Generation (RAG) approach in retrieving highly relevant, user-specific data. Then we discuss how this approach paves the way for building secure, locally-operated personal assistants for computing students, addressing the growing concerns about data privacy in cloud-based systems.

The research questions (RQs) we investigate are as follows:

Downloaded from the ACM Digital Library by James Stevens on April 7, 2025.

RQ1: Can computing educators feasibly use and deploy Generative AI locally to avoid the risks around data security, privacy, and organizational policies?

RQ2: Do SLMs with RAG outperform LLMs while using course-specific data in supporting computer science students' learning and problem-solving?

This research is driven by critical questions surrounding these models' functionality, effectiveness, and practical applications, particularly in computing education.

2 BACKGROUND

2.1 Integrating LLMs into computing education

With the emergence of large language models (LLMs), the field of computing education has been given much to think about. These models possess the ability to generate code and understand natural language instructions [8, 10, 32]. With such powerful capabilities now accessible to students at the click of a button, educators have grown interested in how this new technology will change computing pedagogy [36].

While some of the literature surrounding LLMs in computing education has focused on assessing their capabilities and using them for generating teaching material, part of the literature surrounding LLMs has focused on how these models can be leveraged as tools to support student learning [17, 36]. For example, in order to help students improve their programming abilities, Pankiewicz et al. harnessed OpenAI's gpt-3.5 model to provide students with feedback on programming assignments, finding that AI-generated guidance proved helpful [34]. Leinonen et al. investigated how LLMs can be used to generate quality explanations of code examples, and proposed that these explanations can be used to help scaffold student ability to understand and explain code [26]. Seeking to assist CS1 students in dealing with errors, Leinonen et al. attempted to use LLMs to create explanations and suggest fixes for confusing error messages, but found that the explanations were often unsatisfactory [27]. And to help facilitate assistance on online help platforms, Hellas et al. found that LLMs can be used on posts in course help forums to identify errors in student code, acting as a potentially helpful supplement to teaching staff [16].

One interesting use case of LLMs is that of Intelligent Tutoring Systems (ITSs) and Intelligent Teaching Assistants (ITAs). ITSs are computer programs that use AI techniques to tutor students, replicating the experience of learning with a human tutor [33]. CodeHelp is a tool in this area of research, which provided on-demand personal assistance to students while refraining from directly revealing solutions [30]. Similarly, Al-Hossami et al. made GPT engage in Socratic questioning to help guide students to the answers to their questions instead of giving them direct answers [1]. Kumar et al. developed QuickTA [24] which provides problem-solving support to computer science students in CS1 but also other domains such as database systems [22]. ITAs on the other hand are similar to ISTs in that they can provide tutelage to students, but also help instructors as well [40, 45, 46]. For example, Zhou (2020) used deep learning techniques to notify instructors when students were distracted in class, helping them manage class attentiveness [47].

While tools like these may be helpful, many researchers have been curious as to the abilities and limitations of LLMs in education [36]. In an effort to determine the capabilities of GitHub Copilot, Dakhel et al. compared Copilot and human solutions to algorithmic problems, and found that Copilot can be an asset if used by experts, but possibly a liability if used by novices who fail to identify buggy solutions [13]. Research has also been done into how student performance is affected by Copilot use, finding that it acted as a useful starting point for programmers, but found no change in time to completion or success rate of programming exercises [42]. Bellettini et al. found that GPT-3 has the potential to create false solutions to programming problems, and that this poses a danger to student learning [6].

2.2 Small Language Models (SLMs) vs. Large Language Models (LLMs)

LLMs are able to perform as well as they do in large part due to their size. As the scale of a language model increases, its performance improves in many ways [20]. In addition, as the scale of these models increase, new abilities that were nonexistent at smaller scales suddenly emerge once the model is massive enough [44]. As such, small language models by themselves are unable to achieve as remarkable performance as their larger counterparts due to their limited size. However, they are less computationally expensive as larger models tend to use more computational resources [37].

To attain efficiency, researchers have tried to improve performance without scaling model size. For example, Schick et al. adapted PET [38] for tasks that require predicting multiple tokens and combined it with ALBERT [25] to achieve performance that surpassed GPT-3 [39]. In addition, Chung et al. explored how fine-tuning is relatively compute-efficient, and determined that SLMs with instruction fine-tuning can occasionally beat LLMs without it [12].

Currently, popular SLMs include Mistral 7B [18], Gemini Nano [2] and Llama-2 7B [41], among others.

2.3 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is a technique to help language models complete knowledge-intensive tasks by allowing them to retrieve contextual information relating to a user query from a data store, thus giving them stronger knowledge of the topic at hand [28]. RAG has been used to improve code generation and summarization [28, 35], enhance text-to-image generation [11], and perform more advanced slot filling [15], among other use cases.

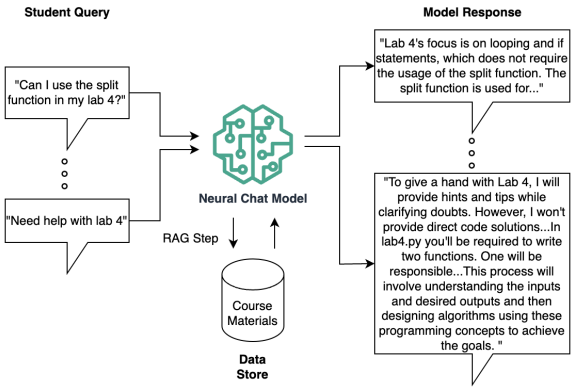

Figure 1: RAG with real Student Queries & Model Responses

Downloaded from the ACM Digital Library by James Stevens on April 7, 2025.

3 METHODS

This section describes the collection method used to obtain CS1 conversational data from a large public North American research-focused institution utilizing gpt-3.5-turbo to support problem-solving. We investigate an alternative implementation, leveraging Mistral and compare its performance, using RAG methods, against gpt-3.5-turbo and gpt-4-32k.

3.1 Dataset Choice - CS1 Conversational Data

In our exploration of the integration of Generative AI in computing education, the series of tests utilized conversation data from the freshman-level Computer Science Introduction Course CS1. In this course, 269 unique students interacted with a gpt-3.5-based AI Assistant, called *QuickTA* [23, 24], cumulatively engaging in a total of 1, 260 conversations, with a total of 1, 988 questions asked. These interactions, which varied in depth with the maximum number of exchanges in a single conversation reaching 31 and the minimum being just one, provided a rich and diverse dataset for our analysis. The dataset columns included prompts, student questions, and responses from gpt-3.5, offering a comprehensive view for evaluating the effectiveness of our SLM-RAG implementation. This variation in the length and complexity of conversations, along with the wide range of unique student engagement patterns, forms a robust basis for assessing the adaptability and performance of the SLM-RAG model in a real-world educational setting.

3.2 Implementation of an SLM (Mistral)

Mistral, a 7.3 billion parameter language model developed by Mistral AI Organization [18], is central to our study. We utilized Intel's conversationally fine-tuned iteration, "neural-chat-7b-v3". This model operates on our laboratory machine with 4-bit quantization [5] and FlashAttention-2 [14], leveraging a single 3090 GPU with 24 GB of VRAM. The decision to employ RAG over subject or student-specific fine-tuning was driven by scalability and cost considerations [31].

3.3 Integration of RAG and LLM comparisons

Retrieval-Augmented Generation (RAG) enhances a language model's ability to generate responses by first retrieving relevant information from a large database or document collection, and then using this retrieved context to inform and guide the generation process 1.

Integrated into Mistral, RAG notably enhances its retrieval capabilities [7, 28, 39]. This addition broadens information accessibility, empowering Mistral to yield more thorough responses. For this purpose, CS1 course data is stored as embedded vectors within the Milvus database collections [43], forming the core of our RAG methodology.

Shifting to our comparative analysis, we utilize essential performance metrics such as accuracy, response time, and hallucination frequency to evaluate the efficacy and dependability of each model in generating pertinent responses [9]. This analysis pits open-source models like Mistral and neural-chat-7b-v3-1 against OpenAI's gpt-3.5-turbo and gpt-4-32k models. Further, we extend our comparison to include neural-chat-7b-v3-1 paired with various RAG methods. Our assessment spans a range of RAG techniques within our IC framework, aimed at pinpointing the most effective strategies. These methods encompass direct retrieval with filter, LangChain strategies (map reduce, stuff, refine, map rerank), and approaches such as LlamaIndex and LMLingua (sentence window, auto merging).

At its core, IC is designed to pre-arrange and categorize course materials based on the anticipated needs of the students, ensuring that the database is ready to address specific queries. This preparation involves breaking down the course content into various levels of detail and abstraction, which are then stored across different IC collections (IC1-3 and KB). This allows for a more efficient retrieval process, enabling the system to provide responses that are not only accurate but also contextually aligned with the students' current learning phase. The dynamic adaptability feature further tailors responses to the evolving interaction patterns, making IC a robust and responsive retrieval system that adapts to the nuanced and changing needs of computing students.

3.3.1 IC Implementation Details: There are two main components of our Intelligence Concentration (IC).

- **i**: First, the *Question Generator* analyzes users' summarized chat history using a SLM and anticipates future inquiries via prompt engineering, generating a list of potential questions.
- **ii**: Second, the *Answer Generator* then allocates these questions across IC collections 1 to 3. Retrieving K relevant documents from our Knowledge Base (KB) for each question; it initially stores responses in IC3. Subsequently, material from IC3 is utilized for responding and stored in IC2, following the same approach for IC1, ensuring a layered and nuanced approach to information retrieval and response generation.

Upon receiving user queries, the system searches across four collections (starting with IC1, then IC2, then IC3 and finally the KB) using specified query strings and K values. The system then evaluates the top K documents from each collection based on their mean relevance. The collection with the highest mean relevance is selected, providing the top K documents in response. The intention is to ensure that a response always has a mapping back to a KB, much like a citation in a research paper.

3.3.2 Layered Retrieval Framework: Our methodology's IC component utilizes a multi-layered retrieval framework, inspired by the FLARE approach [19]. This framework refines the relevance and specificity of the information retrieved by Mistral, with dynamic adaptability to user interactions.

3.3.3 Dynamic Adaptability: IC layers within Milvus are designed for dynamic updates, influenced by user interactions and ongoing dialogue. This adaptability ensures that Mistral's responses are more personalized and context-sensitive, which is important in computing, drawing inspiration from GATE and Self-RAG methodologies [3, 29].

3.4 Scoring Process

To guarantee an equitable comparison, identical prompts are presented to each model. In the prompt, we used a two-shot approach, a method where two examples are provided to the language model to illustrate the desired response format or content. This is to determine if the evaluated model returned responses akin to a Teaching Assistant (TA), providing an example of how a TA should respond: no direct answer, but specific to the retrieved texts, and determining if the generated text is helpful for the user [23]. Subsequently, the model's responses are rated on a 0 to 100 scale, evaluating the

Downloaded from the ACM Digital Library by James Stevens on April 7, 2025.

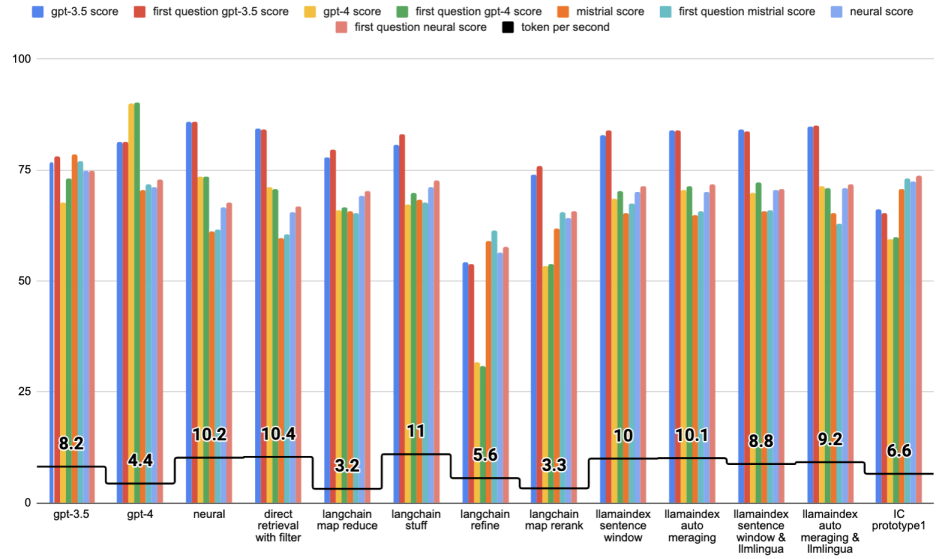

Figure 2: Tokens used per second and performance of models on answering students' questions evaluated by gpt-3.5-turbo, gpt-4-32k, Mistral-7B-Instruct-v0.1, and neural-chat-7b-v3-1. X-axis: models tested, Y-axis: score from 0-100, "first question score" indicates no previous conversation memory while "score" has conversational memory (see Section 3.4 for scoring).

quality and relevance of each model's output based on how a TA should answer their question(s) (see Figure 2).

4 RESULT

Comparing the performance metrics of Mistral and various GPT models combined with different approaches reveals distinct performance characteristics and limitations within each system. This section delves into a quantitative assessment of model efficacy with respect to benchmarks and tasks. To evaluate the different models, we used models from two parties: OpenAI models (gpt-3.5-turbo, gpt-4-32k) and open-source models (Mistral-7B-Instruct-v0.1, neural-chat-7b-v3-1). We asked each model the same prompt to evaluate the responses from gpt-3.5, gpt-4 and neural-chat-7b-v3-1 when presented with real CS1 student questions.

In Figure 2, each bar, divided into eight segments, compares two types of models: OpenAI's (gpt-3.5 and gpt-4) on the left, and open-source (Mistral-7B-Instruct-v0.1 and neural-chat-7b-v3-1) on the right. The black line on each bar marks the rate of token generation per second, which is a indicator to show the average response time for each method (a higher token rate means a faster response).

4.1 Mistral (SLM) vs. GPT (LLM)

In Table 1 showing the comparative analyses, gpt-3.5 and gpt-4 outperform neural-chat-7b-v3-1 (a fine-tuned version of Mistral-7B-v0.1). Their mean scores are 75.6 and 85.8 (OpenAI evaluation) and 76.75 and 70.85 (opensource evaluation) respectively, compared to neural-chat-7b-v3-1's 79.65 (OpenAI) and 63.85 (opensource). This superiority can be attributed to their expansive training dataset, encompassing a diverse range of internet text, and its advanced algorithmic structure, notably featuring ROHF (Regularization of Output Hidden Features) [20]. ROHF aids in stabilizing the learning process and enhancing the model's ability to generalize across various tasks.

Notably, however, the combined approach of neural-chat-7b-v3-1 with an auto-merging RAG method by LlamaIndex with LLMLingua presents a compelling case. Although smaller than GPT models, this model scores 78.1 (OpenAI) and 68.1 (opensource) and shows capabilities close to its larger GPT counterparts The RAG component, which integrates a neural network with a retrieval mechanism, enables the model to access and incorporate external information dynamically during the generation process. This method effectively bridges the gap in raw data size and complexity, illustrating that models with smaller parameter counts, when augmented with intelligent retrieval systems, can achieve performance levels comparable to larger, more complex models like GPTs. Such findings underscore RAG's transformative potential in elevating AI models' efficacy. An additional notable difference lies in the response time, particularly in the token generation rate; gpt-4's token generation rate stands at only 4.4 tokens per second, which is substantially lower than most of the other SLM with RAG methods (the average around 10 tokens per second). This metric highlights not just the efficiency but also the potential for quicker information processing and response generation in SLM with RAG models compared to some larger-scale counterparts.

4.2 IC vs. Other RAG Methods

Since LlamaIndex Automerging and LLMLingua emerged as the most effective RAG methods in our tests, we utilized them to develop our IC framework. During testing, we maintained a summary of user conversations along with new interactions without any clean-up. A new IC build, incorporating previous user conversations, was initiated after every 74 conversations (~323 questions). The results from the IC framework were somewhat mixed. In the open-source model outcomes, IC appeared to be the best-performing RAG method and the second-best model overall, 62.75 (OpenAI) and 71.55 (opensource), ranking higher than gpt-4 but slightly below gpt-3.5. This performance aligns logically, considering that IC is built upon the two most successful RAG methods we evaluated. However, when tested with OpenAI models such as gpt-3.5 and

Downloaded from the ACM Digital Library by James Stevens on April 7, 2025.

gpt-4, these models assessed the IC framework as less effective than other RAG methods.

Table 1: Empirical Comparison

Models / RAGs	OpenAI Average Score	Open Source Average Score
gpt-3.5-turbo (Best Open Source Evaluated Model)	75.6	**76.75**
gpt-4-32k (Best OpenAI Evaluated Model)	**85.8**	70.85
llamaindex auto merging & llmlingua (Best OpenAI Evaluated RAG)	**78.1**	68.1
IC prototype1 (Best Open Source Evaluated RAG)	62.75	**71.55**

5 DISCUSSION

5.1 Can computing educators feasibly use and deploy Generative AI locally?

Our research indicates that localized deployment of Generative AI is not only feasible but also crucial for maintaining data security and privacy within educational settings.

The primary benefits of using Small Language Models (SLMs) locally include independence from cloud services. Local deployment offers significant cost advantages, particularly for long-term or high-volume use. For example, using OpenAI's API for models like gpt-3.5 or gpt-4 incurs a cost based on the number of tokens processed. A rough estimate for 1,000 conversations, depending on their length, can range from US$5.00 for shorter ones to US$10.00 for longer ones with gpt-3.5, and from US$250 to US$500 for gpt-4.

Regarding hardware requirements, a key consideration is the need for a capable GPU. For instance, our implementation utilized a single 3090 GPU with 24 GB of VRAM which supported up to 4 neural chat models simultaneously. However, other NVIDIA GPUs and Apple M-series chips can also run these models. To enhance usability for students, it is best to run the models and vector databases on a server and host a web page that students can access.

The localized approach ensures that educational content remains within a controlled and secure institutional environment, addressing major concerns about data security and privacy in the use of Generative AI for educational purposes. For example, conversation data between the student and the model will not be sent, stored, or reused elsewhere. All the data are localized and controlled by the local system.

5.2 Do SLMs with RAG outperform LLMs in supporting CS students?

The integration of Small Language Models (SLMs) with Retrieval-Augmented Generation (RAG) and course-specific data shows promising potential to outperform Large Language Models (LLMs) in aiding computer science students.

The RAG component, with its dynamic retrieval mechanism, plays a pivotal role in this advancement. It provides contextually relevant, precise, and diverse information, significantly enhancing the learning experience. For instance, a few SLM with RAG combinations (e.g., LlamaIndex Auto Merging & LLMLingua, IC Prototype1) come close to matching the scores of LLMs such as gpt-3.5 and gpt-4. Our prototype version of IC even surpassed the score of gpt-4 (71.55 vs. 70.85) using open-source model evaluation.

This approach addresses common issues such as incorrect or biased responses often encountered with traditional models, offering more tailored and effective educational support. Given their comparable performance, and since SLM with RAG can be deployed locally for enhanced data security and lower management costs, along with the ability to update the vector database with the most relevant course information over time, using SLM with RAG becomes a viable alternative to LLMs for educational purposes.

5.3 We recommend using Neural-7B + IC.

When assisting beginners in CS1 with learning and problem-solving, consider it a choice between two types of educators. Imagine a renowned computer science professor, akin to large language models (LLMs), who is extremely knowledgeable but not specialized in this course. They offer a lot of information, yet it might not always be precisely what students need. In contrast, think of a teaching assistant, similar to Small Language Models with Retrieval-Augmented Generation (SLMs with RAG), especially when boosted by our new feature, Intelligence Concentration (IC). This assistant not only has a deep understanding of the course content but also excels in identifying and conveying the needed information.

For CS1 students, this smaller, smarter system delivers clear, precise answers that are tailored to individual student queries and challenges, akin to a tutor who understands precisely where a student is struggling and provides just the right amount of guidance and explanation. Moreover, this system is more accessible and cost-effective for implementation in schools or online courses. This makes it an excellent choice for educators seeking to equip their students with effective learning tools without incurring high expenses.

6 CONCLUSION AND FUTURE WORK

In conclusion, our research underscores the transformative potential of integrating Generative AI in education through a combination of localized Small Language Models (SLMs), Retrieval-Augmented Generation (RAG) for enhanced educational support, and Intelligence Concentration (IC) for improved efficiency. This framework not only ensures data privacy and control but also significantly enriches the learning experience by providing accurate, relevant, and timely educational content. Despite its current limitations, including the lack of practical deployment and direct student feedback, as well as the ongoing development of the IC framework, the foundational concept of integrating smaller language models with RAG shows immense promise. It exemplifies the potential for Generative AI to revolutionize educational methodologies and outcomes, indicating a future where AI can substantially aid in personalized and effective learning.

Our future work includes implementing and rigorously testing this system in other computing courses (e.g., Information Security, Databases, etc.) so we can closely focus on monitoring student engagement and gather feedback, as well as optimize various IC implementations. Through this practical application, we aim to validate and refine our framework, addressing limitations and expanding its capabilities.

ACKNOWLEDGMENTS

This work was funded by the Learning & Education Advancement Fund from the Office of the Vice-Provost, Innovations in Undergraduate Education, University of Toronto and Microsoft's Accelerating Foundation Model Research program.

Downloaded from the ACM Digital Library by James Stevens on April 7, 2025.

REFERENCES

[1] Erfan Al-Hossami, Razvan Bunescu, Ryan Teehan, Laurel Powell, and Khyati Mahajan et al. 2023. Socratic questioning of novice debuggers: A benchmark dataset and preliminary evaluations. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*. 709–726.

[2] Gemini Team Google: Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. arXiv:2312.11805 [cs.CL]

[3] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv preprint arXiv:2310.11511* (2023).

[4] Maria Teresa Baldassarre, Danilo Caivano, Berenice Fernandez Nieto, Domenico Gigante, and Azzurra Ragone. 2023. The Social Impact of Generative AI: An Analysis on ChatGPT. In *Proceedings of the 2023 ACM Conference on Information Technology for Social Good* (Lisbon, Portugal) *(GoodIT '23)*. ACM, 363–373.

[5] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. 2019. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems* (2019).

[6] Carlo Bellettini, Michael Lodi, Violetta Lonati, Mattia Monga, and Anna Morpurgo. 2023. DaVinci goes to Bebras: a study on the problem solving ability of GPT-3. In *Proceedings of the 15th International Conference on Computer Supported Education. 2: CSEDU*. 59–69.

[7] Norbert Braunschweiler, Rama Doddipatla, Simon Keizer, and Svetlana Stoyanchev. 2023. Evaluating Large Language Models for Document-grounded Response Generation in Information-Seeking Dialogues. *arXiv preprint arXiv:2309.11838* (2023).

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, and Jared Kaplan et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020).

[9] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2023. Benchmarking large language models in retrieval-augmented generation. *arXiv preprint arXiv:2309.01431* (2023).

[10] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, and Henrique Ponde de Oliveira Pinto et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[11] Wenhu Chen, Hexiang Hu, Chitwan Saharia, and William W. Cohen. 2022. Re-imagen: Retrieval-augmented text-to-image generator. *arXiv preprint arXiv:2209.14491* (2022).

[12] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, and Yi Tay et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[13] Arghavan Moradi Dakhel, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, and Michel C. Desmarais et al. 2023. Github copilot ai pair programmer: Asset or liability? *Journal of Systems and Software* (2023).

[14] Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691* (2023).

[15] Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, and Alfio Gliozzo. 2021. Robust retrieval augmented generation for zero-shot slot filling. *arXiv preprint arXiv:2108.13934* (2021).

[16] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, and Lilja Kujanpää et al. 2023. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. *arXiv preprint arXiv:2306.05715* (2023).

[17] Yann Hicke, Anmol Agarwal, Qianou Ma, and Paul Denny. 2023. AI-TA: Towards an Intelligent Question-Answer Teaching Assistant using Open-Source LLMs. arXiv:2311.02775 [cs.LG]

[18] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, and Devendra Singh Chaplot et al. 2023. Mistral 7B. arXiv preprint arXiv:2310.06825.

[19] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, and Qian Liu et al. 2023. Active retrieval augmented generation. arXiv preprint arXiv:2305.06983.

[20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, and Benjamin Chess et al. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[21] Devang Kulshreshtha, Muhammad Shayan, Robert Belfer, Siva Reddy, Iulian Vlad Serban, and Ekaterina Kochmar. 2022. Few-shot Question Generation for Personalized Feedback in Intelligent Tutoring Systems. *arXiv preprint arXiv:2206.04187* (2022).

[22] Harsh Kumar, Ilya Musabirov, Mohi Reza, Jiakai Shi, and Anastasia Kuzminykh et al. 2023. Impact of Guidance and Interaction Strategies for LLM Use on Learner Performance and Perception. *arXiv preprint arXiv:2310.13712* (2023).

[23] Harsh Kumar, Ilya Musabirov, Mohi Reza, Jiakai Shi, Xinyuan Wang, Joseph Jay Williams, Anastasia Kuzminykh, and Michael Liut. 2024. Impact of Guidance and Interaction Strategies for LLM Use on Learner Performance and Perception. arXiv:2310.13712 [cs.HC]

[24] Harsh Kumar, Ilya Musabirov, Joseph Jay Williams, and Michael Liut. 2023. QuickTA: Exploring the Design Space of Using Large Language Models to Provide Support to Students. In *Workshop on Partnerships for Co-Creating Educational Content at the Learning Analytics and Knowledge Conference 2023*. Arlington, TX, USA.

[25] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, and Piyush Sharma et al. 2019. Albert: A lite bert for self-supervised learning of language representations. arXiv:1909.11942 [cs.CL]

[26] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, and Seth Bernstein et al. 2023. Comparing code explanations created by students and large language models. arXiv:2304.03938 [cs.CY]

[27] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, and Paul Denny et al. 2023. Using large language models to enhance programming error messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 563–569.

[28] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, and Vladimir Karpukhin et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[29] Belinda Z. Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. 2023. Eliciting human preferences with language models. *arXiv preprint arXiv:2310.11589* (2023).

[30] Mark Liffiton, Brad Sheese, Jaromir Savelka, and Paul Denny. 2023. Codehelp: Using large language models with guardrails for scalable support in programming classes. arXiv:2308.06921 [cs.CY]

[31] Ziyang Luo, Can Xu, Pu Zhao, Xiubo Geng, and Chongyang Tao et al. 2023. Augmented Large Language Models with Parametric Knowledge Guiding. *arXiv preprint arXiv:2305.04757* (2023).

[32] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, and Huan Wang et al. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474* (2022).

[33] Hyacinth S Nwana. 1990. Intelligent tutoring systems: an overview. *Artificial Intelligence Review* 4, 4 (1990), 251–277.

[34] Maciej Pankiewicz and Ryan S. Baker. 2023. Large Language Models (GPT) for automating feedback on programming assignments. *arXiv preprint arXiv:2307.00150* (2023).

[35] Md Rizwan Parvez, Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2021. Retrieval augmented code generation and summarization. *arXiv preprint arXiv:2108.11601* (2021).

[36] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, and Ibrahim Albluwi et al. 2023. The robots are here: Navigating the generative ai revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*. 108–159.

[37] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, and Jordan Hoffmann et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446* (2021).

[38] Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference. arXiv preprint arXiv:2001.07676.

[39] Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118* (2020).

[40] George Siemens and Ryan S. J. d. Baker. 2012. Learning analytics and educational data mining: towards communication and collaboration. In *Proceedings of the 2nd international conference on learning analytics and knowledge*. 252–254.

[41] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, and Amjad Almahairi et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[42] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*. 1–7.

[43] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, and et al. 2021. Milvus: A Purpose-Built Vector Data Management System. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. 2614–2627.

[44] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, and Barret Zoph et al. 2022. Emergent abilities of large language models. arXiv preprint arXiv:2206.07682. Available online at https://arxiv.org/abs/2206.07682.

[45] Kalina Yacef. 2002. Intelligent teaching assistant systems. In *International Conference on Computers in Education, 2002.* IEEE, 136–140.

[46] Shengquan Yu and Yu Lu. 2021. *An introduction to artificial intelligence in education.* Springer.

[47] Zheyu Zhou. 2020. An intelligent teaching assistant system using deep learning technologies. In *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*. 18–22.

Downloaded from the ACM Digital Library by James Stevens on April 7, 2025.