The Journal of Computing Sciences in Colleges

Papers of the 30th Annual CCSC Central Plains Conference

April 5th-6th, 2024
Graceland University
Lamoni, IA

Bin Peng, Associate Editor
Park University

Joseph Kendall-Morwick, Regional Editor
Washburn University

Volume 39, Number 6 **April 2024**

The Journal of Computing Sciences in Colleges (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors

Following is a listing of the contact information for the members of the Board of Directors and the Officers of the Consortium for Computing Sciences in Colleges (along with the years of expiration of their terms), as well as members serving CCSC:

Scott Sigman, President (2024), ssigman@drury.edu, Mathematics and Computer Science Department, Drury University, Springfield, MO 65802.

Bryan Dixon, Vice President/President-Elect (2024), bcdixon@csuchico.edu, Computer Science Department, California State University Chico, Chico, CA 95929.

Baochuan Lu, Publications Chair (2024), blu@sbuniv.edu, Division of Computing & Mathematics, Southwest Baptist University, Bolivar, MO 65613.

Ed Lindoo, Treasurer (2026), elindoo@regis.edu, Anderson College of Business and Computing, Regis University, Denver, CO 80221.

Cathy Bareiss, Membership Secretary (2025), cathy.bareiss@betheluniversity.edu, Department of Mathematical & Engineering Sciences, Bethel University, Mishawaka, IN 46545.

Judy Mullins, Central Plains Representative (2026), mullinsj@umkc.edu, University of Missouri-Kansas City, Kansas City, MO (retired).

Michael Flinn, Eastern Representative (2026), mflinn@frostburg.edu, Department of Computer Science & Information Technologies, Frostburg State University, Frostburg, MD 21532.

David R. Naugler, Midsouth Representative (2025), dnaugler@semo.edu, Brownsburg, IN 46112.

David Largent, Midwest Representative(2026), dllargent@bsu.edu, Department of Computer Science, Ball State University, Muncie, IN 47306.

Mark Bailey, Northeastern Representative (2025), mbailey@hamilton.edu, Computer Science Department, Hamilton College, Clinton, NY 13323.

Shereen Khoja, Northwestern Representative(2024), shereen@pacificu.edu, Computer Science, Pacific University, Forest Grove, OR 97116.

Mohamed Lotfy, Rocky Mountain Representative (2025), mohamedl@uvu.edu, Information Systems & Technology Department, College of Engineering & Technology, Utah Valley University, Orem, UT 84058.

Tina Johnson, South Central Representative (2024), tina.johnson@mwsu.edu, Department of Computer Science, Midwestern State University, Wichita Falls, TX 76308.

Kevin Treu, Southeastern Representative (2024), kevin.treu@furman.edu, Department of Computer Science, Furman University, Greenville, SC 29613.

Michael Shindler, Southwestern Representative (2026), mikes@uci.edu, Computer Science Department, UC Irvine, Irvine, CA 92697.

Serving the CCSC: These members are serving in positions as indicated:
Bin Peng, Associate Editor, bin.peng@park.edu, Department of Computing and Mathematical Sciences, Park University, Parkville, MO 64152.
Brian Hare, Associate Treasurer & UPE Liaison, hareb@umkc.edu, School of Computing & Engineering, University of Missouri-Kansas City, Kansas City, MO 64110.
George Dimitoglou, Comptroller, dimitoglou@hood.edu, Department of Computer Science, Hood College, Frederick, MD 21701.
Megan Thomas, Membership System Administrator, mthomas@cs.csustan.edu, Department of Computer Science, California State University Stanislaus, Turlock, CA 95382.
Karina Assiter, National Partners Chair, karinaassiter@landmark.edu, Landmark College, Putney, VT 05346.
Deborah Hwang, Webmaster, hwangdjh@acm.org.

CCSC National Partners

The Consortium is very happy to have the following as National Partners. If you have the opportunity please thank them for their support of computing in teaching institutions. As National Partners they are invited to participate in our regional conferences. Visit with their representatives there.

Gold Level Partner

Rephactor
ACM2Y
ACM CCECC

Welcome to the 2024 CCSC Central Plains Conference

It is my pleasure to welcome you to the 30th annual Consortium for Computing Sciences in Colleges Central Plains Region Conference hosted by Graceland University in the community of Lamoni, Iowa. The conference this year appeals to a variety of different levels with a broad range of papers, nifty assignments, and workshops. Topics range from the effect of Generative AI to engaging younger students in coding. The diverse program is sure to have something for everyone. There will be two outstanding speakers to share with us. Dr. Monica McGill has a passion for computing education research. She founded her own company, Institute for Advancing Computer Education(IACE), which focuses directly on equity-focused K-12 CS education. The banquet speaker is Mr. Kevin Juhnke who has been at Principal Financial Group since his graduation from Graceland in 1990. His experience ranges from COBOL developer to Enterprise Architecture. Both speakers will deliver valuable insights for attendees. Additionally, the program includes activities for students: paper and poster presentations, Hack-A-Thon reports, and a programming contest.

The paper acceptance rate for this year was 56%. Every paper was reviewed by at least three reviewers. This ensures that the papers accepted in this program continue to be first-rate. I am certain that the conference program will benefit both computer science educators and students.

I am thankful to have worked with a group of dedicated individuals. The devoted committee members, reviewers, session moderators, and many other volunteers, make this conference go and provide an excellent experience for the conference attendees. I would also like to express my appreciation to the administration, staff, colleagues, ITS staff and Ackerley scholars at Graceland University. They volunteered many hours to prepare and support the conference. Lastly, thank you to the numerous individuals, vendors, and organizations whose support helped make the conference possible.

I am pleased to be hosting this year's conference in Lamoni, which is a gateway city to the state of Iowa. Please take time to enjoy the many Amish shops, antique stores, and heritage of the small rural community with a big heart. Graceland University welcomes all to the Lamoni campus. I hope you find the conference inspirational and educational. We look forward to seeing you in April.

Kevin Brunner, Ph.D.
Graceland University
CCSC-2024 Central Plains Conference Chair

2024 CCSC Central Plains Conference Steering Committee

Conference Chair
Kevin Brunner ..... Graceland University
Conference Publicity
Bill Siever ..... Washington University in St. Louis
Joan Gladbach ..... University of Missouri Kansas City
Kevin Brunner ..... Graceland University
Shane Adams ..... Graceland University
Keynote Speaker
Kevin Brunner ..... Graceland University
Pre-Conference Workshop
Wen-Jung Hsin ..... Park University
Judy Mullins ..... Retired
Joan Gladbach ..... University of Missouri Kansas City
Papers
Charles Riedesel ..... University of Nebraska-Lincoln
Judy Mullins ..... Retired
Ron McCleary ..... Retired
Panels, Tutorials, Workshops
Ron McCleary ..... Retired
Judy Mullins ..... Retired
Mohammad Rawashdeh ..... University of Central Missouri
Mahmoud Yousef ..... University of Central Missouri
Nifty Assignments
Mohammad Rawashdeh ..... University of Central Missouri
Brian Hare ..... University of Missouri Kansas City
Bill Siever ..... Washington University in St. Louis
Ron McCleary ..... Retired
Judy Mullins ..... Retired
Lightning Talks
Joseph Kendall-Morwick ..... Washburn University
Bill Siever ..... Washington University in St. Louis
Wen-Jung Hsin ..... Park University
K-12 Nifty Assignments and Lightning Talks
Bill Siever ..... Washington University in St. Louis
Perla Weaver ..... Johnson County Community College
Belinda Copus ..... University of Central Missouri
Mohammad Rawashdeh ..... University of Central Missouri
Student Paper Session
Scott Sigman ..... Drury University

Wen-Jung Hsin ..... Park University
Mahmoud Yousef ..... University of Central Missouri
Joseph Kendall-Morwick ..... Washburn University

Student Poster Competition

Joseph Kendall-Morwick ..... Washburn University
Ron McCleary ..... Retired

Student Hack-a-thon

Scott Sigman ..... Drury University
Chris Branton ..... Drury University
Mahmoud Yousef ..... University of Central Missouri
Bill Siever ..... Washington University in St. Louis

Student Programming Contest

Charles Riedesel ..... University of Nebraska-Lincoln
Joan Gladbach ..... University of Missouri Kansas City
Brian Hare ..... University of Missouri Kansas City

Two-Year College Outreach

Suzanne Smith ..... Johnson County Community College
Trisch Price ..... Johnson County Community College
Mahmoud Yousef ..... University of Central Missouri

Local Arrangements

Kevin Brunner ..... Graceland University

Regional Board — 2024 CCSC Central Plains Region

Regional Rep & Board Chair
Judy Mullins ..... Retired
Registrar & Membership Chair
Ron McCleary ..... Retired
Current Conference Chair
Kevin Brunner ..... Graceland University
Next Conference Chair
Eric Manley ..... Drake University
Past Conference Chairs
Mahmoud Yousef ..... University of Central Missouri
Perla Weaver ..... Johnson County Community College
Secretary
Diana Linville ..... Northwest Missouri State University
Regional Treasurer
Ajay Bandi ..... Northwest Missouri State University
Regional Editor
Joseph Kendall-Morwick ..... Washburn University
Webmaster
Deepika Jagmohan ..... St. Charles Community College

Reviewers — 2024 CCSC Central Plains Conference

Imad Al Saeed ..... Saint Xavier University, Chicago, IL
Rad Alrifai ..... Northeastern State University, Tahlequah, OK
Beth Arrowsmith ..... University of Missouri - St. Louis, Saint Peters, MO
Ajay Bandi ..... Northwest Missouri State University, Maryville, MO
Chris Branton ..... Drury University, Springfield, MO
Kevin Brunner ..... Graceland University, Lamoni, IA
John Buerck ..... Saint Louis University, St. Louis, MO
David Bunde ..... Knox College, Galesburg, IL
Karla Carter ..... Bellevue University, Bellevue, NE
Aziz Fellah ..... Northwest Missouri State University, Maryville, MO
Ernest Ferguson ..... Northwest Missouri State University, Maryville, MO
David Furcy ..... University of Wisconsin Oshkosh, Oshkosh, WI
Brian Hare ..... University of Missouri-Kansas City, Kansas City, MO
Suvineetha Herath ..... Carl Sandburg College, Galesburg, IL
Charles Hoot ..... Northwest Missouri State University, Maryville, MO
Wen Hsin ..... Park University, Parkville, MO
Joseph Kendall-Morwick ..... Washburn University, Topeka, KS
Brian Kokensparger ..... Creighton University, Omaha, NE
Srinivasarao Krishnaprasad ..... Jacksonville State University, Jacksonville, AL
Diana Linville ..... Northwest Missouri State University, Maryville, MO
Baochuan Lu ..... Southwest Baptist University, Bolivar, MO
Eric Manley ..... Drake University, Des Moines, IA
Thomas Mertz ..... Kansas State Polytechnic, Salina, KS
Jose Metrolho ... Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal
Kian Pokorny ..... McKendree University, Lebanon, IL
Hassan Pournaghshband ..... Kennesaw State University, Kennesaw, GA
Charles Riedesel ..... University of Nebraska - Lincoln, Beatrice, NE
Michael Rogers ..... University of Wisconsin Oshkosh, Oshkosh, WI
Jamil Saquer ..... Missouri State University, Springfield, MO
William Siever ..... Washington University, St. Louis, MO
Cindy Tu ..... Northwest Missouri State University, Maryville, MO
Timothy Urness ..... Drake University, Des Moines, IA
Henry Walker ..... Grinnell College (retired), Napa, CA
Maria Weber ..... Saint Louis University, St. Louis, MO
Mudasser F Wyne ..... National University, San Diego, CA
Cong-Cong Xing ..... Nicholls State University, Thibodaux, LA
Baoqiang Yan ..... Missouri Western State University, Saint Joseph, MO
Mahmoud Yousef ..... University of Central Missouri, Warrensburg, MO

Navigating the Technological Tide: How Leveraging Past Perspectives Can Help You Create Sound Software for Everyone*

Opening Keynote

Monica McGill, Ed.D. (she/her)
CEO and Founder, CSEdResearch.org

Abstract

In the ever-expanding landscape of our digital existence, artificial intelligence (AI) is seamlessly integrating into the fabric of our lives, much like a dye diffusing through water. While technological progress is not a novel concept, insights from the past offer invaluable lessons about how innovation shapes our world. Drawing from these historical lenses, my keynote talk aims to provide a framework within the realm of computer science that encompasses AI and technological advancements. Join me on a journey to delve into its nuanced impact on present and future developers, emphasizing the importance of adopting critical perspectives and providing you with some basic tools on how to do so. Leveraging these insights can imbue your projects with depth, reflection, and profound meaning in the ever-evolving landscape of computing and AI.

Bio

Dr. Monica McGill is passionate about computing education research, which motivated her to found the Institute for Advancing Computing Education (IACE) as a non-profit. She has led the organization in conducting equity-focused K-12 CS education research funded by grants and contracts from various national and international organizations, including the National Science

Foundation, Google, Amazon Future Engineer, CSTA, Code.org, and more. Monica earned her B.S. in Computer Science and Mathematics from University of Illinois-Urbana Champaign, M.S. in Computer Science from George Washington University, and Ed.D. in Curriculum and Instruction from Illinois State University as a non-traditional student. Prior to forming IACE, she worked for several years in industry as a computer scientist and then as a twice-tenured professor of computer science and game design/development for over 15 years.

She has authored/co-authored 85+ articles related to computing education and is currently working as a primary investigator on several National Science Foundation (NSF) grants exploring CS education in the US. She also served as inaugural chair for the ACM-W North America committee, on the Computer Science Teachers Association (CSTA) Board and Sjögren's Foundation board and as an associate editor of the ACM Transactions on Computing Education.

Monica enjoys embarking on travel adventures with her husband Dan, sometimes bringing along their dogs, Coco and Benny, and sometimes road-tripping cross country to visit one of their daughters.

Perspectives on Technology's impact on Financial Services and the Future Workforce*

Banquet Address

Kevin Juhnke
Enterprise Architect

Abstract

A variety of experiences over a 30+ year career in the computing field at Principal Financial provides for a life-long journey with many fascinating insights. Mr. Juhnke will focus his thoughts on:

- Business and market challenges in Financial Services and their impact on an organization's Technology Strategies
- The impact key maturing and emerging technologies have on the Financial Services industry including...
  - Cloud Advancements
  - Generative AI
  - Blockchain
  - "Citizen" Development

He will conclude his thoughts with perspectives on areas where educators can help position students to be more marketable and impactful in financial services tech jobs after graduation.

Bio

Kevin Juhnke is a Director of Enterprise Architecture at Principal Financial Group in Des Moines, IA and a '90 graduate of Graceland College (University) with a BS in Computer Science and Math. Kevin has been at Principal for the entirety of his career, beginning as a COBOL developer, transitioning to leadership roles in Technology R&D, and eventually settling into the realm of Enterprise Architecture. For the last 20+ years, Kevin has led the Enterprise Architecture practice of various divisions within Principal including Corporate Finance, Enterprise CRM, Principal Asset Management, Principal International and is currently the lead Enterprise Architect for Principal Retirement and Income Solutions (i.e. Principal's Retirement Division). In the role of Enterprise Architect, Kevin spends his days identifying and shaping technology strategy, enabling transition of business and technology strategy to execution, and driving modernization of technology portfolios to enable future business strategies; some days with more success than others.

The Effect of ChatGPT: Student Perspective and Performance Achievement*

Wen-Jung Hsin
Computer Science and Information Systems
Park University
Parkville, MO 64152
wen.hsin@park.edu

Abstract

ChatGPT, introduced in November 2022, has rapidly used in various educational systems, prompting the U.S. Department of Education to explore the role of Artificial Intelligence (AI) in teaching and learning. This paper focuses on the impact of AI, particularly ChatGPT, in Computer Science education from the student's perspective and student's performance achievement. Specifically, a study in a Computer Networking course encouraged students to use ChatGPT for learning-related questions, followed by a post-exam survey to evaluate its impact on their learning. Both student feedback and performance achievement indicate that ChatGPT has made a positive impact in their learning in the Computer Networking course.

1 Introduction

In the realm of education, the utilization of virtual, conversational assisting tools such as Chatbots [2, 11], as well as Amazon Alexa and Google Home, has experienced significant growth over the past decade. This increase can be

attributed mainly to its capacity to provide effective learning experience in a one-on-one teacher-student context, effectively addressing challenges found in scenarios such as large class sizes and online education platforms [26]. Since ChatGPT unleashed in November 2022, it has been used in various education systems such as healthcare, K-12, colleges, special, online learning, language learning, and career counseling in an astounding rate [2, 12]. As such, fairly recently, the Office of Educational Technology in the U.S. Department of Education issued a report entitled "Artificial Intelligence (AI) and the Future of Teaching and Learning" [6] delving into the benefits and hurdles associated with incorporating AI into education. The timing of this publication underscores the increasing significance of AI in education. In this paper, we look at the effect of AI tools on Computer Science education from the student's perspective. In particular, we narrow our focus to the student's viewpoint on whether ChatGPT helps student learning. Examining whether ChatGPT helps student learning from the student's perspective is essential for understanding student engagement, motivation, and the effectiveness of their learning experience. Additionally, it aids in identifying challenges and areas for improvement. To achieve this, we conduct a study in a computer networking course. While the students were preparing for a proctored, face-to-face exam, we suggested to the students to ask a list of questions related to computer networking using ChatGPT. After the exam, we had the students take a survey to see if ChatGPT improves their learning in the subject matter. This paper reports the student survey result and performance achievement. This paper is organized as follows. Section 2 gives a literature review. Section 3 describes our research method and student survey result. Section 4 discusses the student survey result. Section 5 analyzes the student performance achievement. Section 6 gives a summary and conclusion.

2 Literature Review

Since the debut of ChatGPT in November 2022, there are a lot of research studies [3, 5, 7, 9, 10, 14, 16, 22, 25, 27] investigating its use in education. For example, papers [2, 23] study the benefits of ChatGPT provided to teaching and learning. Paper [21] tries to understand ChatGPT's implication on assessment. Papers[8, 14, 19] study ChatGPT in its potential in medical or media educations. Paper [28] looks at the user experience of ChatGPT to find the implication in education. Papers [9, 20] advocate implementing ChatGPT's in education in a responsible way. Additionally, there are several review or survey papers such as [1, 4, 5, 13, 17, 18] describing the use of ChatGPT in education from student's perspective without actual student data support. For example, paper [17], a review paper, calls for responsible and ethical use of ChatGPT

for lifelong learning. Papers [5, 13, 18] provide commentary or review on the benefits and challenges of ChatGPT for students and teachers. Paper [24] reviews 100 articles on how ChatGPT disrupts higher education (both enhances and affects the student learning.) Paper [1] conducts 30 tests on ChatGPT and suggests ways for instructors to change assignments or tests to prevent the possibility of future human unlearning. Paper [13], a review paper, presents ChatGPT as opportunities for learning from elementary school students to university students, plus learners with disabilities. Notably, none of these papers present empirical student's viewpoint data to support their assertions. In this paper, our emphasis is on the students' perception on ChatGPT, and the data presented is drawn directly from the students' own experiences and feedback in a computer networking class.

3 Our Study

3.1 Research design

In Spring 2023 semester, we conducted the research study in a computer networking class with 12 students. The students in class are college juniors and seniors. While the students were preparing for an in-person, proctored exam, we recommended that the students utilized ChatGPT and asked the suggested questions as shown in Table 1.

Throughout the semester, students have developed a fundamental grasp of computer networking concepts, and the questions presented in Table 1 show some of the key concepts covered in the course.

3.2 Student Feedback

After the exam, a survey is conducted as shown in Table 2. The students rate each prompt with a 5-point Likert scale (i.e., 5-Strongly agree, 4-Agree, 3-Neutral, 2-Disagree, and 1-Strongly disagree.) Table 2 shows the average and standard deviation of each prompt asked in the survey. These findings indicate that the majority of students either agreed or strongly agreed with the prompts, reflecting a high level of consensus among the respondents.

Written feedback from students also shows that ChatGPT has been extremely beneficial to them. Here are positive comments reflecting their experience with ChatGPT.

- "ChatGPT is a useful resource because it is able to further explain topics I might struggle in."
- "It helps to explain information and I can ask for more info if I don't understand."

Q1	What is the current trend of computer networking? Please provide some real-life examples.
Q2	What is the best way to protect computer networking from security threat? Please provide some real-life situations and solutions.
Q3	Why do we need protocols and models in computer networking?
Q4	What is the key difference between OSI and TCP/IP?
Q5	What are the characteristics of CSMA/CA? Currently, which technologies in real-life are using CSMA/CA?
Q6	Are there cases where people try to break the fiber optic cables on the ocean floor? If so, please briefly describe some of these cases.
Q7	What is the address resolution in computer networking used for?
Q8	In IPv4, how do we know if an address is a network address or a broadcast address?
Q9	In IPv4, how do I know if an address is public or private? Please give an example of each kind.
Q10	Why does a switch need MAC address table?
Q11	What is the purpose of NAT in computer networking? If we did not have NAT, what would have happened?
Q12	What is a default gateway?
Q13	I cannot remember the names of TCP/IP layers. Can you suggest ways to remember the names of the TCP/IP layers?

Table 1: Recommended Questions for ChatGPT

- "I pose the questions that I got wrong on quizzes I took this semester. Using ChatGPT I was able to get a more through out [SIC] way of the actual answer for those questions. I think that using ChatGPT can be a great resource to look at studying as well as correcting wrong answers."
- "I like that it gives further information to what my question answers more like a broad explanation. I like it because I am a slow learner."
- "It is good if you just need a few refresher tips, such as formulas..."
- "It makes finding information easier than to trying to look it up."
- "... if I don't understand it, I can tell that it [ChatGPT] dumbs it down so that I can understand the information in a more simple [SIC] way."
- "... Overall, I think that this [ChatGPT's examples] is one the best advantages of using ChatGPT."

ID	Survey Prompt	Average	STD
S1	ChatGPT provides useful resources for learning and understand computer networking concepts.	4.25	0.83
S2	Compared the learning with and without ChatGPT, learning with ChatGPT helps me understand the concepts of computer networking better than without ChatGPT.	3.92	0.86
S3	ChatGPT offers practical applications of how computer networking concepts are applied in real-world scenarios.	4.17	0.80
S4	Through ChatGPT's real-world examples and applications, I have gained a deeper appreciation of the computer networking.	3.83	0.80
S5	Through ChatGPT, I have gained a deeper understanding of computer networking concepts.	3.75	1.09
S6	ChatGPT has broadened my knowledge in computer networking.	3.83	1.07

Table 2: Survey and Result

- "I will say that using ChatGPT, I understand concepts in computer networking a lot better. It helps me visualize but also teach me concepts in ways that I understand. Using this [ChatGPT] has given me a better appreciation for this course."
- "I am amazed on how ChatGPT works by itself, and it also increased my motivation to learn, and it is like to have a 'mentor' 24/7."
- ".. For some [students], it is also like a tutor. Most people aren't as comfortable with speaking up in class, so to have that [ChatGPT] as an option is very helpful to me."
- "The only thing I could say that it did show me better was different examples, so that is a bright spot in my opinion."
- "I feel like I learned a lot with what I have used ChatGPT. Its answers are always clear and precise."

Here are some negative comments from using ChatGPT:

- ".. I like to work on things hands-on... while it does give me the answers to the problems, it does not give the steps to get there."
- " ... I can see this tool being abused by students."

3.3 How do the students use ChatGPT?

In the survey, we also ask the students how they have been using ChatGTP so far by selecting all those roles applied to them [15]. Figure 1 illustrates the results, with the Y bar representing the number of students selecting each role.

Other roles such as translating languages or troubleshooting aids receive very few votes. As can be seen from Figure 1, the majority of the students have used ChatGPT as a tutor and have used it to create content. Some students provided more specifics on how they have used ChatGPT, including

- "Using ChatGPT to help prepare me for interviews."
- "It mostly only applies to text summaries."
- "I have asked it to write short stories."
- "It helps with decoding code and errors which is great and also gives me an explanation on where I went wrong."
- "Trying to use it to generate code, it sort of works, but often I found it to be fragmented pieces of code glued together when it came to anything complex."
- "To check my grammar and writing skills."
- "I mainly use it for studying or getting further clarification on topics for classes."

4 Discussion: Interpretation of Student Survey Results

Question: Did ChatGPT enhance student learning in computer science courses from the student's perspective? The feedback provided by the students suggests that ChatGPT has played a highly valuable role in their learning experiences. The fact that the majority of students either agreed or strongly agreed with the survey prompts indicates a positive reception of ChatGPT's assistance in understanding computer networking concepts. The low standard deviations in the responses highlight the consistency in student opinions, further underscoring the overall positive impact of ChatGPT. These findings suggest that AI-powered tools, such as ChatGPT, could be valuable in education, offering students accessible support for understanding challenging subjects such as computer networking. Figure 1 and the extensive feedback from students show the varied applications in which ChatGPT has been utilized as a versatile supporting tool. In particular, a majority of students have employed ChatGPT

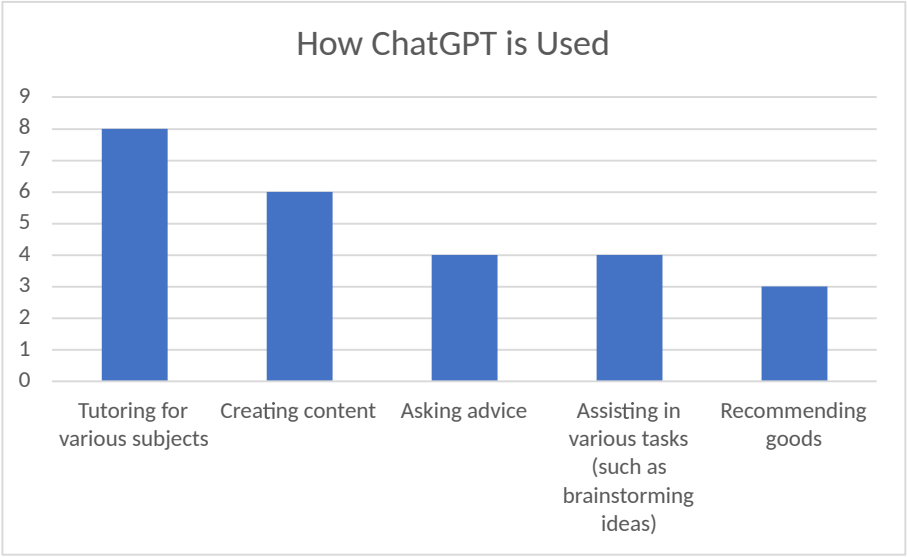


Figure 1: How ChatGPT is used by the Students

as a personal tutor, leveraging it for tasks such as content generation, interview preparation, text summarization, creative writing, debugging assistance, advice solicitation, product recommendations, and even code generation. Despite some students observing constraints in handling intricate coding tasks, the general impression indicates that ChatGPT can serve as a valuable and versatile tool for students across different environments, providing them with support, guidance, and enrichment in their learning journeys.

5 Student Performance Achievement: Comparative Result

We are interested in knowing whether ChatGPT has any effect in student performance. As such, we compared the student exam performance result between 2023 Spring semester (after ChatGPT became publicly available) and 2021 Fall semester (before ChatGPT was publicly available.) The exams in both semesters are comprehensive, proctored, final exams, and have the same level of difficulties. The class size in 2023 Spring semester is 12 students, and the class size in 2021 Fall semester is 10 students. Both classes were taught by the same instructor.

Table 3 reveals that the exam average for the 2023 Spring semester surpasses

	Average	Standard Deviation
Student Performance in 2023 Spring with ChatGPT	20.93	6.62
Student Performance in 2021 Fall without ChatGPT	18.43	11.94

Table 3: Student Performance Comparison With and Without ChatGPT

that of the 2021 Fall semester, and the standard deviation is lower in the 2023 Spring semester. This indicates that students performed better in the 2023 Spring semester. Our initial investigation suggests that using ChatGPT as a learning aid enhances students' comprehension of computer networking concepts. However, it's important to note that this study is based on 2 small classes, and further research with a larger dataset is necessary.

6 Summary and Conclusion

The outcome of the student survey serves as informal evidence of the value that ChatGPT contributes to the classroom in computer networking. Students express contentment with the assistance it offers, noting its capacity to provide personalized guidance, answer questions, and offer explanations, thereby enhancing their understanding and overall learning experience. Moreover, the survey result highlights the various roles ChatGPT can fulfill. Finally, the integration of ChatGPT has led to enhanced student performance in the study, comparing courses before and after its public accessibility, though it's important to acknowledge the study's limited dataset.

References

[1] Mohammad Awad AlAfnan et al. "Chatgpt as an educational tool: Opportunities, challenges, and recommendations for communication, business writing, and composition courses". In: *Journal of Artificial Intelligence and Technology* 3.2 (2023), pp. 60–68.

[2] David Baidoo-Anu and Leticia Owusu Ansah. "Education in the era of generative artificial intelligence (AI): Understanding the potential benefits of ChatGPT in promoting teaching and learning". In: *Journal of AI* 7.1 (2023), pp. 52–62.

[3] Yejin Bang et al. "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity". In: *arXiv preprint arXiv:2302.04023* (2023).

[4] Emmanuel Mensah Bonsu and Daniel Baffour-Koduah. “From the consumers’ side: Determining students’ perception and intention to use ChatGPT in Ghanaian higher education”. In: *Journal of Education, Society & Multiculturalism* 4.1 (2023), pp. 1–29.

[5] Aras Bozkurt et al. “Speculative futures on ChatGPT and generative artificial intelligence (AI): A collective reflection from the educational landscape”. In: *Asian Journal of Distance Education* 18.1 (2023).

[6] M. Cardona, R. Rodriguez, and K. Ishmael. *Artificial Intelligence and the Future of Teaching and Learning*. `https://www2.ed.gov/documents/ai-report/ai-report.pdf`. Accessed 2024-01-13.

[7] Yogesh K Dwivedi et al. ““So what if ChatGPT wrote it?” Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy”. In: *International Journal of Information Management* 71 (2023), p. 102642.

[8] Aidan Gilson et al. “How does ChatGPT perform on the United States medical licensing examination? The implications of large language models for medical education and knowledge assessment”. In: *JMIR Medical Education* 9.1 (2023), e45312.

[9] Mohanad Halaweh. “ChatGPT in education: Strategies for responsible implementation”. In: *Contemporary Educational Technology* 15.2 (2023).

[10] Abid Haleem, Mohd Javaid, and Ravi Pratap Singh. “An era of ChatGPT as a significant futuristic support tool: A study on features, abilities, and challenges”. In: *BenchCouncil transactions on benchmarks, standards and evaluations* 2.4 (2022), p. 100089.

[11] Jin-Xia Huang et al. “A chatbot for a dialogue-based second language learning system”. In: *CALL in a climate of change: adapting to turbulent global conditions–short papers from EUROCALL* (2017), pp. 151–156.

[12] Mohd Javaid et al. “Unlocking the opportunities through ChatGPT Tool towards ameliorating the education system”. In: *BenchCouncil Transactions on Benchmarks, Standards and Evaluations* 3.2 (2023), p. 100115.

[13] Enkelejda Kasneci et al. “ChatGPT for good? On opportunities and challenges of large language models for education”. In: *Learning and individual differences* 103 (2023), p. 102274.

[14] Tiffany H Kung et al. “Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models”. In: *PLoS digital health* 2.2 (2023), e0000198.

[15] *List of ChatGPT Roles*. `https://medium.com/@autonomizedinnerpulse/list-of-chatgpt-roles-82892bfc2d88`. Accessed 2024-01-13.

[16] Brady D Lund and Ting Wang. "Chatting about ChatGPT: how may AI and GPT impact academia and libraries?" In: *Library Hi Tech News* 40.3 (2023), pp. 26–29.

[17] David Mhlanga. "Open AI in education, the responsible and ethical use of ChatGPT towards lifelong learning". In: *SSRN 4354422* (2023).

[18] Emmanuel Opara, Adalikwu Mfon-Ette Theresa, and Tolorunleke Caroline Aduke. "ChatGPT for teaching, learning and research: Prospects and challenges". In: *Glob Acad J Humanit Soc Sci* 5 (2 2023), pp. 33–40.

[19] John V Pavlik. "Collaborating with ChatGPT: Considering the implications of generative artificial intelligence for journalism and media education". In: *Journalism & Mass Communication Educator* 78.1 (2023), pp. 84–93.

[20] Mike Perkins. "Academic Integrity considerations of AI Large Language Models in the post-pandemic era: ChatGPT and beyond". In: *Journal of University Teaching & Learning Practice* 20.2 (2023), p. 07.

[21] Jürgen Rudolph, Samson Tan, and Shannon Tan. "ChatGPT: Bullshit spewer or the end of traditional assessments in higher education?" In: *Journal of Applied Learning and Teaching* 6.1 (2023).

[22] Sakib Shahriar and Kadhim Hayawi. "Let's Have a Chat! A Conversation with ChatGPT: Technology, Applications, and Limitations". In: *Artificial Intelligence and Applications* 2.1 (June 2023), pp. 11–20.

[23] Sarin Sok and Kimkong Heng. "ChatGPT for education and research: A review of benefits and risks". In: *SSRN 4378735* (2023).

[24] Miriam Sullivan, Andrew Kelly, and Paul McLaughlan. "ChatGPT in higher education: Considerations for academic integrity and student learning". In: *Journal of Applied Learning & Teaching* 6.1 (2023).

[25] Teo Susnjak. "ChatGPT: The end of online exam integrity?" In: *arXiv preprint arXiv:2212.09292* (2022).

[26] Rainer Winkler and Matthias Söllner. "Unleashing the potential of chatbots in education: A state-of-the-art analysis". In: *Academy of Management Proceedings*. Vol. 2018. 1. Academy of Management Briarcliff Manor, NY 10510. 2018, p. 15903.

[27] Xiaoming Zhai. "Chatgpt for next generation science learning". In: *XRDS: Crossroads, The ACM Magazine for Students* 29.3 (2023), pp. 42–46.

[28] Xiaoming Zhai. "ChatGPT user experience: Implications for education". In: *SSRN 4312418* (2022).

Key Performance Indicator Selection for Online Teaching Performance Prediction*

Gary Yu Zhao and Cindy Zhiling Tu
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
zhao@nwmissouri.edu, cindytu@nwmissouri.edu

Abstract

This paper attempts to explore a simple and effective set of key indicators for online instructor's performance prediction based on a large amount of data generated from the learning management system (LMS). Data was collected from a Midwest university LMS platform - Canvas. Univariate and bivariate analyses were conducted, and then the filter-based technique and a wrapper-based method were combined to select the key influence variables. We compared the performance of the logistic regression (LR) machine learning model on different selected key indicator sets. Ten selected variables were suggested as the best key indicators for online instructors' performance evaluation.

1 Introduction

As the prevalence of online programs and enrollment of remote students has surged significantly in higher education, it is important for educational institutions to evaluate online instructors' performance to provide high-quality online education. Traditional instructor performance evaluation is based on student evaluation surveys. However, the response rate for online classes is far lower than for campus classes. These surveys are usually completed at the end of the

semester. Compared to the traditional teaching performance evaluation, the data of online teaching is limited to the learning management system (LMS) platform only. Online students do not have a chance to meet with instructors in person. Therefore, for the timely interventions given to online instructors, it is crucial to predict their performance evaluation using LMS data as early as possible. Nowadays, one of the major challenges of higher educational institutions is the large volume of data from ERP and LMS platforms and how it can be used to improve the quality of academic programs, services, and managerial decisions [1]. Researchers have done a lot of research on building the prediction model based on different modeling methods to predict students' academic performance in the online education context. However, there is a significant lack of studies on constructing the instructor's performance prediction models [14, 10, 2]. There are several limitations in the existing literature. First, there is a dearth of research using LMS behavioral and activity data for constructing instructors' performance prediction models. Instead, the survey data is predominately used to train and test the models. Second, in existing work, there is a lack of research on employing variable selection methods for determining the key indicators that affect the prediction of instructors' performance. Third, in the previous studies, the prediction of the instructor's performance is primarily a binary classification problem, e.g., the performance is defined as "acceptable" or "no-acceptable"; there is a lack of related work focusing on multi-class classification problems. To fill the abovementioned gaps, this study aims to solve the problem: What indicators can be selected from LMS to predict online teachers' performance effectively? This study explores a few variable selection methods with the same data scenario for seeking the improvement of performance and interpretability of the prediction solution. Moreover, we use the Logistic Regression (LR) model to compare and evaluate these sets of selected key indicators.

2 Related Work

We reviewed and synthesized the existing research from three aspects: data, variable selection, and evaluation.

2.1 Data

The data used for online instructor performance prediction can be from an individual or a combination of these three data sets: demographic and socioeconomic information such as age and gender, subjective perception data such as the survey of students' perceived teacher's teaching quality, behavioral and activity data such as language characteristics, teaching and learning process, and summative data. In extant literature on learning performance prediction,

explicit and implicit behavioral data or behavioral data combined with demographic data has been widely used to analyze and build models. For example, Pan et al. (2016) applied the quantile regression analysis to investigate how the time and frequency of login curriculum, browsing teaching material, and curriculum discussion affect the final-term teachers' performance assessment [12]. With the popularization of the LMS platform, scholars can use abundant records about instructors' and learners' activities to facilitate the model construction [15, 16, 11, 7]. Researchers construct the prediction model using the LMS data that includes users' information, course selection information, course content viewing data, participation in discussions, and academic assessment data [16, 4, 5, 9].

2.2 Variable Selection

Compared to the demographic information and perception data, the learning process and summative behavioral data extracted from LMS can be used as better variable selection factors [11]. Machine learning methods are widely employed to improve the accuracy of the prediction model. Reported literature shows that various machine learning approaches can provide satisfactory results and similar performance. Nonetheless, it is a big challenge to analyze the prediction performance of machine learning algorithms due to different evaluation indicators and problem specifications that directly affect the effectiveness and efficiency of the algorithms [6]. Only a few previous studies consider the impact of independent variables on the performance of machine learning methods and the interpretability of the prediction models. For example, Huichao Mi et al. (2022) use multiple logistic regression, sequential forward selection (SFS), and sequential backward selection (SBS) to select the optimal feature combination that affects the performance prediction [11]; Ahmed et al. (2016) conduct attribute selection using the OneR algorithm in Weka [5]; Agaoglu (2016) applies stepwise discriminant analysis method to determine the significant variables [3]. Some other methods have been used in the existing work, such as grouping and ranking analysis [13], ablation feature analysis [8], weight analysis by random forest algorithm [16], and traditional linear regression analysis [7].

2.3 Evaluation

In existing studies, accuracy, precision, recall, and F1-score have been primarily used to evaluate and compare the performance of the prediction models [2, 4, 5, 9, 3]. Accuracy measures the rate of total correct predictions to all predictions. Precision measures the correctness rate of the class predictions done as positive by the classifier, whereas recall measures the rate of positives correctly predicted as positive by the classifier. In this study, we use these

metrics to analyze and compare the effectiveness of various key indicator sets based on the Logistic Regression model.

3 Methodology

This section presents the methodology of our work, which includes data collection, data preprocessing, variable selection, and explanation for analysis and prediction.

3.1 Data Collection

Data were collected from a Midwest public university, which contains 37 online programs and 1864 online classes taught from 2019 to 2022. The data was extracted from Canvas - an LMS platform and EvaluationKit – integrated survey management software to Canvas for instructors' teaching performance evaluation by students. Students evaluate instructors' performance from four aspects using 12 Likert (values are 1-4) questions: instructional design, instructional delivery, instructional assessment, and course management. We retrieved all students' evaluation raw data and aggregated the mean value for each course taught by a particular instructor. This performance evaluation result is used as the dependent variable in our study. The key performance indicators are independent variables that consist of behavioral and activity features, including teaching and learning process data and summative data in LMS. These variables include:

- Faculty ID – The identity number of the online instructor.
- Course_sis_section_id – Identity string of online class.
- Instructor total activity time – Accumulated action time in a specific class.
- Instructor page views – Viewed pages on the course site.
- Publish course site on time – The instructor publishes the course site on the term start date or later.
- Assignment grading days – Days of all assignment grading in a course.
- Student dropout rate – Number of dropped students out of total students enrolled in a course.
- Evaluation of total responses – Total of students who should evaluate the course instructor.

- Total responded – Total of students who have completed instructor evaluation.
- Total student enrolled – Number of students who enrolled in specific class.
- Student total activity time – Accumulated student action time in the course site.
- Student page view – Student-viewed pages on the course site.
- Student participation rate – Student participation of online discussions/quizzes.
- Assignment submission rate – Assignment submissions out of total assignments.
- On-time assignment submission rate – On-time submissions out of total assignments.
- Late assignment submission rate – Late submissions out of total assignments.
- Missing assignment submission rate – Missing submissions out of total assignments.
- Student course score – Student current overall scores (grades).

3.2 Data Preprocessing

The target variable represents the performance of the instructor. The raw data are numeric values between 1 and 4. We encoded these values to four classes based on the historical experience and university policies, i.e., "3" (Excellent, score 3.85-4.0), "2" (Good, score 3.5-3.85), "1" (Acceptable, score 3.25-3.5), "0" (Improve, score 1.0-3.25). All student individual-level features were aggregated to the course level. In addition, we encoded faculty ID and course-section-sis-id using the hash encode method to hide identification data. Finally, we merged instructor activity, student, and instructor evaluation data based on the encoded ID. We did the quality check and preprocessing for all 16 variables using Python Pandas, NumPy, Matplotlib, and Seaborn. First, we conducted univariate analysis to identify and remove the records with outliers, missing values, and error data. Univariate analysis also provides summary statistics, such as mean, median, mode, variance, and standard deviation, which describe the central tendency and variability of the variable. Then, we used bivariate analysis to identify relationships between two variables. Identifying which variables are strongly related to the target variable (in the case of regression or

classification) can guide model development. In machine learning and statistics, bivariate analysis helps identify potential predictor variables for building predictive models. The dataset is split into two subsets: training data (80

3.3 Key Indicator Selection

A valid variable selection process can reduce overfitting, improve accuracy, and decrease training time. We performed the variable selection with a combined method before modeling using the Python Scikit-learn library. Firstly, we used different methods independently to select variables, e.g., univariate and bivariate analysis, filter-based method on Pearson correlation (Figure 1), and wrapper-based method on sequential forward selection (SFS). Then, we compared and analyzed various sets of selected variables on the Logistic Regression model.

3.4 Evaluation

We used four popular measurements, accuracy, precision, recall, and F1-score, to assess the model performance.

4 Results

All exploration jobs, such as variable selection, model fitting, testing, evaluation, etc., are completed in the Jupyter notebook with Python 3 environment. Most popularly used machine learning related Python libraries were imported for our work, e.g., Pandas, NumPy, matplotlib, seaborn, sklearn, and mlxtend.

4.1 Indicator Selection

As shown in Figure 2, we originally extracted 18 independent variables from the LMS platform based on the educational literature and institutional experience. Then, we used three variable selection methods independently. Fourteen variables were selected based on univariate and bivariate analysis. Twelve variables were selected based on the Pearson correlation filter-based method. Eight variables were selected based on the SFS wrapper-based method. Finally, ten independent variables were determined by human experts based on the above three methods. They are (1) Instructor total activity time, (2) Mean of grading days, (3) Total student enrolled, (4) Dropout rate, (5) Evaluation total responded, (6) Student average activity time, (7) Student average page views, (8) Average on-time submission rate, (9) Average late submission rate, and (10) Student average score.

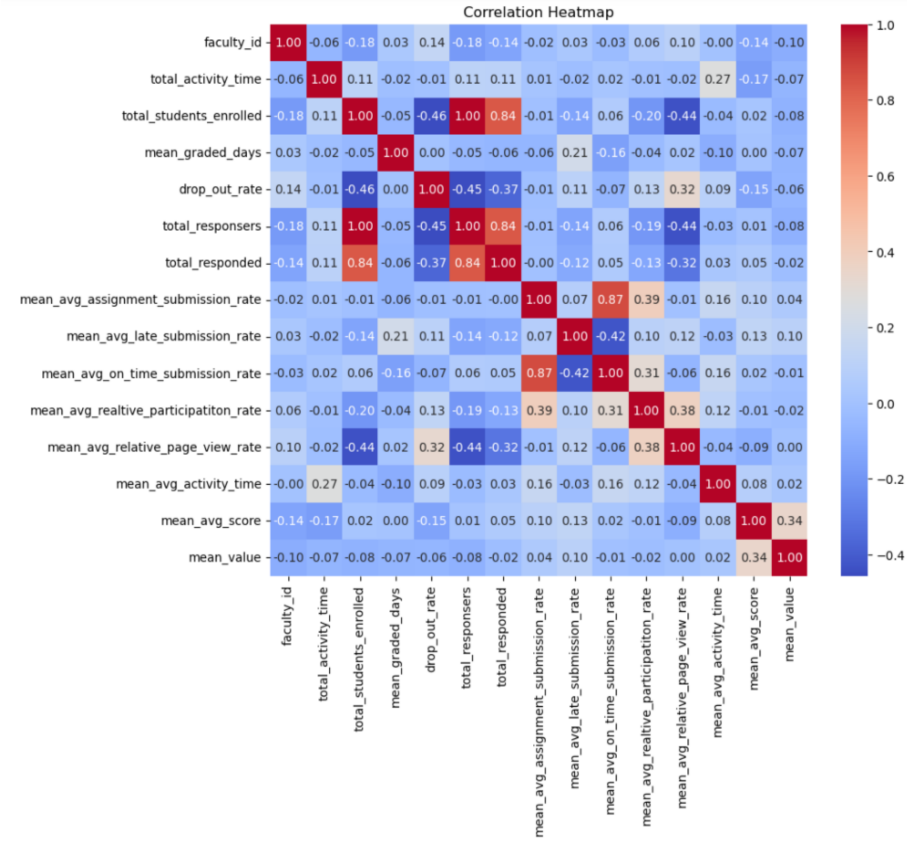


Figure 1: The correlation heat map between variables

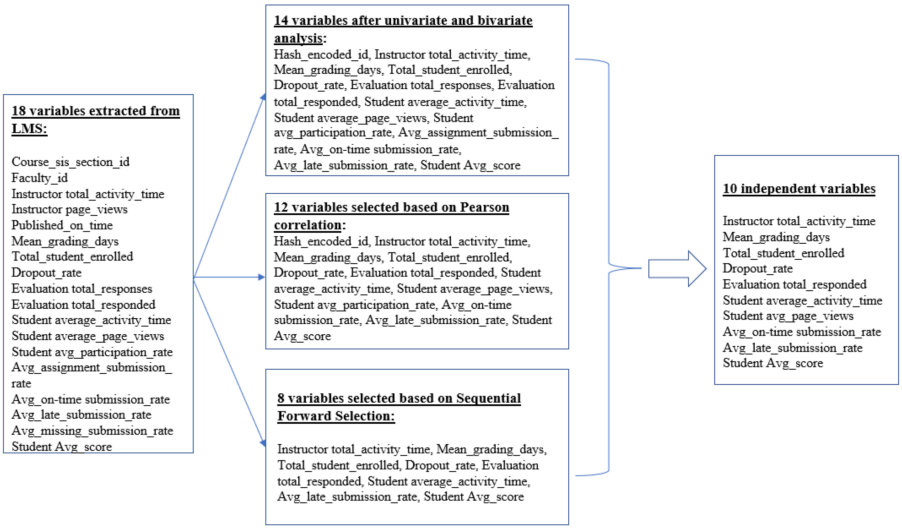


Figure 2: The process of indicator selection

4.2 Comparison of Key Indicator Sets

We compared four sets of key performance indicators on the LR machine learning model. We also compared four sets of variables' performance with the cross-validation method on the LR model. As shown in Table 1, when we fit the dataset to the Logistic Regression (LR) model, the set of 10 indicators optimized by human experts synthesized from the other three variable selection methods achieved the highest performance (60.05% Accuracy), followed by 12 indicators (58.71%), 14 indicators (55.50%), and eight indicators (53.89%). Moreover, we can see that computing time decreases along with reducing the indicators.

Set of Indicators	Accuracy	Precision	Recall	F1-Score	Computing Time (sec.)
14 indicators	55.50%	55.50%	55.50%	55.50%	0.1159
12 indicators	58.71%	58.71%	58.71%	58.71%	0.0903
Ten indicators	60.05%	60.05%	60.05%	60.05%	0.0810
Eight indicators	53.89%	53.89%	53.89%	53.89%	0.0085

Table 2 shows the performance of four sets of indicators using 5-fold and 10-fold cross-validation train processes on the LR model. First, the performance

of the LR model increases slightly by 2% from 5-fold to 10-fold cross-validation for each set of indicators. Second, the performance of LR in both 5-fold and 10-fold cross-validation for each set of indicators does not change. The computing time increases slightly from 5-fold to 10-fold for all sets of indicators. The computing time keeps decreasing in both 5-fold and 10-fold, along with the reduction of the indicators.

cross-validation	5-fold			10-fold		
Set of Indicators	Mean Cross-Validation Score - Accuracy	Standard Deviation of Cross-Validation Scores	Computing Time (sec.)	Mean Cross-Validation Score - Accuracy	Standard Deviation of Cross-Validation Scores	Computing Time (sec.)
14 indicators	53%	3%	0.1247	55%	6%	0.1826
12 indicators	53%	3%	0.0863	55%	6%	0.1190
Ten indicators	53%	3%	0.0706	55%	6%	0.0948
Eight indicators	53%	3%	0.0638	55%	6%	0.0606

5 Discussion

To answer the research question, what indicators can be selected from LMS to predict online teachers' performance effectively, we conclude that ten variables selected by human experts synthesized from the other three variable selection methods show the best performance on the LR model. Conducting variable selection iteratively can balance the trade-off between the number of features selected and model performance. In summary, our best variable set on the LR model is ten indicators. This set achieves 60.05% accuracy, which is lower than what we expected. The small dataset may cause this – we have collected only 1,864 with 16 features. According to the literature, we may need more than 5,000 data items. For example, Abunasser et al. (2022) achieved over 90% accuracy using a dataset including 5820 with 33 features [2]. Moreover, the low performance may be due to the imbalanced dataset, inappropriate models, low informative variable selections, etc. Additionally, these issues lead to the ineffectiveness of the cross-validation training process. These issues will be turned into our future work for this study.

6 Conclusion

Compared to the existing studies on predicting online instructors' performance, our study is outstanding in three aspects. First, this study focuses on the per-

formance prediction for online instructors based on their behavior and activities on the LMS platform instead of using subjective student survey data. All data, including critical indicators and the target variable are extracted from online platforms. Second, we present how variable selection methods impact performance prediction. These feature selection methods include a univariate and bivariate analysis-based method, a filter-based method, a wrapper-based method, and a human experts-based comprehensive method. Third, we train, test, and compare four sets of indicators, then suggest the best solution for satisfying the research objectives – a set of 10 indicators. Theoretically, this work enriches the research of applying variable selection methods in a learning analytics context, especially in instructors' performance prediction with LMS data. Practically, the artifact of prediction modeling with variable selection methods and a proposed optimal set of indicators can be used for online instructors' performance prediction effectively and efficiently. It helps the early intervention during the learning and teaching process and improves overall educational performance.

References

[1] Anwar Muhammad Abaidullah, Naseer Ahmed, and Edriss Ali. "Identifying Hidden Patterns in Students" Feedback through Cluster Analysis". In: *International Journal of Computer Theory and Engineering* 7.1 (2015), p. 16.

[2] Basem S Abunasser et al. "Prediction of instructor performance using machine and deep learning techniques". In: *International Journal of Advanced Computer Science and Applications* 13.7 (2022).

[3] Mustafa Agaoglu. "Predicting instructor performance using data mining techniques in higher education". In: *IEEE Access* 4 (2016), pp. 2379–2387.

[4] Fateh Ahmadi and ME Shiri Ahmad. "Data mining in teacher evaluation system using WEKA". In: *International Journal of Computer Applications* 63.10 (2013).

[5] Ahmed Mohamed Ahmed, Ahmet Rizaner, and Ali Hakan Ulusoy. "Using data mining to predict instructor performance". In: *Procedia Computer Science* 102 (2016), pp. 137–142.

[6] Fisnik Dalipi, Ali Shariq Imran, and Zenun Kastrati. "MOOC dropout prediction using machine learning techniques: Review and research challenges". In: *2018 IEEE global engineering education conference (EDUCON)*. IEEE. 2018, pp. 1007–1014.

[7] Pengcheng Jiao et al. “Artificial intelligence-enabled prediction model of student academic performance in online engineering education”. In: *Artificial Intelligence Review* 55.8 (2022), pp. 6321–6344.

[8] Hamid Karimi et al. “Online Academic Course Performance Prediction Using Relational Graph Convolutional Neural Network.” In: *International Educational Data Mining Society* (2020).

[9] P Kathiroli et al. “Predicting the performance of instructors using Machine learning algorithms”. In: *High. Technol. Lett* 26 (2020), pp. 694–705.

[10] Moyan Li and Yawen Su. “Evaluation of online teaching quality of basic education based on artificial intelligence”. In: *International Journal of Emerging Technologies in Learning (iJET)* 15.16 (2020), pp. 147–161.

[11] Huichao Mi et al. “Research on Constructing Online Learning Performance Prediction Model Combining Feature Selection and Neural Network.” In: *International Journal of Emerging Technologies in Learning* 17.7 (2022).

[12] Wen-Tsao Pan, Chiung-En Huang, and Chiung-Lin Chiu. “Study on the performance evaluation of online teaching using the quantile regression analysis and artificial neural network”. In: *The Journal of Supercomputing* 72 (2016), pp. 789–803.

[13] Cynthia Schubert-Irastorza and Dee L Fabry. “Improving Student Satisfaction with Online Faculty Performance.” In: *Journal of Research in Innovative Teaching* 4.1 (2011).

[14] Wen Wang et al. “What makes a star teacher? A hierarchical BERT model for evaluating teacher’s performance in online education”. In: *arXiv preprint arXiv:2012.01633* (2020).

[15] Yongxian Yang. “The evaluation of online education course performance using decision tree mining algorithm”. In: *Complexity* 2021 (2021), pp. 1–13.

[16] Jing Yu. “Academic Performance Prediction Method of Online Education using Random Forest Algorithm and Artificial Intelligence Methods.” In: *International Journal of Emerging Technologies in Learning* 15.5 (2021).

Examining Student Use of AI in CS1 and CS2*

Eric D. Manley, Timothy Urness,
Andrei Migunov, and Md. Alimoor Reza
Department of Mathematics and Computer Science
Drake University
Des Moines, IA 50311
{eric.manley,timothy.urness,andrei.migunov,md.reza}@drake.edu

Abstract

The launch of ChatGPT in November 2022 marked a seismic disruption to many disciplines and industries, including higher education. For the first time, students everywhere have widely available access to a Large Language Model (LLM) capable of generating content - including solutions to programming assignments in CS1 and CS2 - that can pass as the work of a high-achieving student while making traditional plagiarism-detection obsolete. This has spurred various responses in higher education, including a shift to more in-class and unplugged assessments. At the same time, LLMs are transforming the way that many people work, including professional software developers, and students similarly might be able to use them to enhance their learning. In this paper, we report on our experiences with a permissive policy towards the use of ChatGPT and other artificial intelligence (AI) tools for assisting students with their programming assignments in CS1 and CS2 courses in the Spring 2023 semester. Students were allowed to use these tools however they wished as long as they submitted a form which included a transcript of their chat and a reflection on what they learned, if anything, through the interaction. We found that students largely approached the AI in positive ways and that they seemed to genuinely learn from the

experience. We also document some things that did not go well and that remain challenges to using AI in programming courses, along with our recommendations on how these might be dealt with in the future.

1 Introduction

Shortly after the introduction of OpenAI's ChatGPT in November 2022 [8], students and instructors everywhere quickly realized AI's potential to solve exam problems, generate high-quality essays, and write code on par with high-achieving students. The *New York Times* was among many popular media sources to profile the AI plagiarism threat and how higher education was reacting to thwart it [3], noting many examples of schools that are phasing out asynchronous assignments in favor of in-class written and oral assessments. The article also highlights tools that are being developed to detect AI-generated work along with the schools that are eager to use them.

Computer Science educators had perhaps been more aware of the coming danger than the general population. In their paper *The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming*, Finnie-Ansley et al. [1] tested OpenAI Codex (a GPT-3 model fine-tuned for code generation) on 23 typical introductory programming tasks found in CS education literature. The vast majority of tasks were successfully solved by the AI, and the others were all correct with the exception of a trivial formatting error. The authors conclude that the technology "could be considered an emergent existential threat to the teaching and learning of introductory programming" [1]. At the same time, professional software developers have been using it as a resource for learning and increasing productivity [11], either as a standalone tool or integrated into developer tools like GitHub Copilot [2] which had been shown to increase developer productivity [9].

This raises an important question for CS educators: *Do LLMs represent an existential threat that we should fight against or should we embrace it as just another evolution in the way students and professionals work?* Before reacting, we first wanted to know more about how students approach the use of these tools and whether they learn anything as a result. To this end, we implemented a permissive policy towards the use of AI in our CS 1 and CS 2 courses in the Spring 2023 semester. Students were allowed to use them however they wished, including possibly to generate complete solutions to programming assignments, as long as they completed a learning reflection to help us understand their experience. The rest of this paper describes what we learned. We will first cover some additional related work and then discuss our implementation details, results, conclusions and recommendations going forward.

2 Background and Related Work

Because LLMs have only recently been made widely available to the public, we are not aware of any studies into how programming students themselves approach these tools for assistance on their own. However, researchers have investigated other questions intersecting the use of LLMs and education. As discussed above, Finnie-Ansley et al. demonstrated Codex's remarkable ability to solve problems typically encountered in introductory programming courses [1]. MacNeil et al. consider how GPT-3 and Codex can be integrated into CS pedagogy - for generating code explanations, generating programming assignments, and generating code for larger software projects [6]. Kazemitabaar et al. performed a controlled experiment that showed students in introductory programming courses had increased performance when using Codex while note degrading performance on later learning assessments [4].

Looking beyond CS, Rudolph et al. [10] explore the challenges and opportunities for ChatGPT in education and note that the concern that ChatGPT threatens traditional written assignments lends for the opportunity for more innovative, effective assessment with the potential to transform education. Rudolph et al. [10], McMurtrie [7], and Sharples [12] also recommend that the new technology be embraced and incorporated into future pedagogy by exploring how to shape and harness the new tools as opposed to stopping students from using them. Suggestions include using flipped learning to emphasize the critical pieces of work that are completed during class and avoiding formulaic assignments.

3 Implementation Details

Students enrolled in one CS1 and one CS2 section at our institution (a private midwestern university), during the Spring 2023 semester were presented with a permissive course policy on using ChatGPT and other AI to assist their learning (as long as they filled out an AI learning reflection form) as well as surveys to discover their views about LLMs. With IRB approval for this study, students were asked for consent to publish data obtained from the surveys, student work, and learning reflections. We removed data for students who did not consent and removed identifying information from those who did. We also removed examples of work where the students did not provide enough information to analyze (for example, some students filled out the requested reflection form but did not provide transcripts of their interaction with the AI). We discuss the policy and surveys below, followed by information on the coding scheme we used to summarize student work.

3.1 AI Learning Reflection

The learning reflection form asked students to include the entire transcript of their interaction with the AI tool, even parts they didn't use. It also included questions asking students to explain whether/how they used the AI content as part of their submission and how they checked its accuracy. Finally, they were invited to reflect on their learning with the prompt *"Give some evidence that shows what you learned from using the AI tool for this assignment. For example, this could be a written description showing you can explain the content in question, some new code that applies what you learned to a different problem, a new version of the code that was changed in sufficient ways to better solve the problem, etc."*

3.2 Surveys

Near the beginning of the semester, students were given a survey in order to judge their prior familiarity and views about ChatGPT and similar AI. The purpose of this was to test whether educator fears about plagiarism were validated by student views and whether students were likely to approach these tools for positive use cases (like a kind of AI-Teaching-Assistant - for debugging, help with understanding) or negative use cases (like plagiarism).

1. How do AI Assisted chatbots, like ChatGPT or IBM Watson, make you feel? (Nervous or scared; Excited; Interested; Indifferent)

2. What do you think a college's policy on using AI in classes should be? (Totally allow; Allow in most cases; Ban in most cases; Totally ban)

3. How often do you think students in courses like this will turn in AI-generated content as if it were their own work? (Never; Rarely; Sometimes; Often; Always)

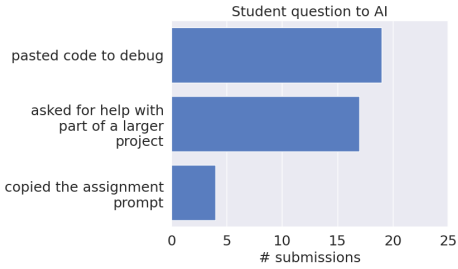
4: How often do you think students in courses like this one will use ChatGPT to help debug code? (Never; Rarely; Sometimes; Often; Always)

5: How often do you think students in courses like this will use ChatGPT to help them understand ideas that they struggle with? (Never; Rarely; Sometimes; Often; Always)

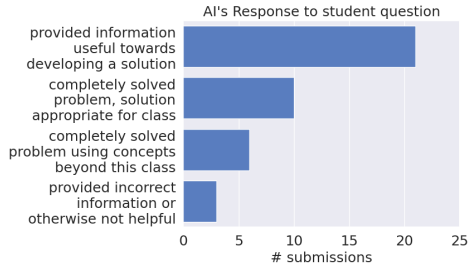
Note that we asked these questions in terms of how often they thought students in courses like this would do these things as a proxy for their own behavior in order to elicit more honest answers - a student might not admit to dishonest intent on the survey, but it might give insight into student perspectives on actions their social circles might find acceptable.

At the end of the semester, students filled out a similar survey which included an additional question on whether they complied with the course policy:

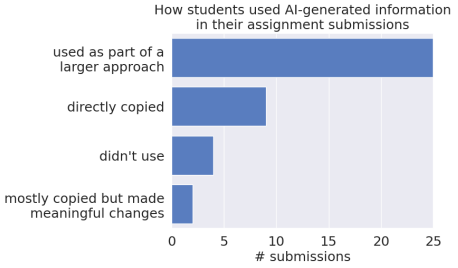
6. If you used ChatGPT or a similar AI tool to help you with an assignment in this class, how often did you fill out the AI-Assisted Learning Reflection



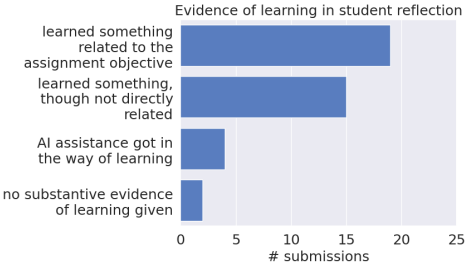
(a) Student Question



(b) AI Response



(c) Student Assignment Submission



(d) Evidence of Learning

Figure 1: Summary of student interaction with AI for assignments

document? (Not applicable - I didn't use ChatGPT or a similar AI tool for any assignments; Always - I filled out the form every time I used an AI for an assignment; Sometimes - there were times I filled out the form and other times I didn't; Never - I used AI but never filled out the form)

For each of the learning reflections in the study, we reviewed the AI chat transcripts, student learning reflections, and assignment submissions. We categorized them according to four criteria: (1) the kind of question that the student posed to the AI, (2) the kind of response given by the AI, (3) how the student used the information in their submission, and (4) the evidence of learning provided in the student reflection. Fig. 1 summarizes the codes within each criterion along with the number of submissions that exhibited each code.

4 Summary of Student Work and Reflections

The study contained 40 submissions, with some students submitting the learning reflection for multiple assignments and others submitting none. As shown in Figure 1a, there were a small number of cases in which the students were

seeking a direct solution to the assigned problem. In most cases, students either pasted their own code for debugging or asked for help with only a part of a larger problem. As shown in Figure 1b, the AI usually provided code which either completely solved the problem or was otherwise useful in helping the students develop a correct solution. Interestingly, in less than a quarter of cases, the students directly copied the code as their solution (see Figure 1c). This is despite the fact that students often found that the AI gave them more help than they were asking for. For example, one student said that they asked the AI

> *"just to get me started/see if there were any basic ideas I was missing. Basically, it did its job "too well" and gave me code that would just finish the assignment."*

Another student mentioned

> *"I wanted to attempt to gradually develop the code with the assistance of chat GPT but the first prompt I gave it, it reuturmed the full code." (sic)*

Students also seemed to genuinely learn important concepts through their interaction with the AI. As shown in Figure 1d, students provided satisfactory evidence of learning in most cases - often centering on the content central to the assignment, but nearly equally often on other important concepts. For example, in an assignment intended to increase proficiency in the use of lists and dictionaries, a student asked ChatGPT about an error caused by trying to access a variable outside of the function it was defined in. The AI gave an explanation which included the following:

> *"The error message you're encounter is because you're trying to call the `most_popular_in_genre` function with the `movies` variable, but `movies` is not defined in your code."*

In fact, the variable was defined *inside* of `most_popular_in_genre` and the ChatGPT suggestions for fixing it were incorrect, but the student was able to fix their code and came away with a better understanding of local vs. global variables and the relationship between arguments and parameters - something that was supposed to have been previously learned. While not the primary intent of the assignment, this was a positive learning experience for the student. In a handful of cases, the AI provided code that got in the way of student learning. For example, in a case where students were supposed to use a custom stack class to solve a problem, the AI solved the problem using a standard list as a stack. The code solved the problem, and the student turned in the code, but it missed one of the main points of the assignment. In another case, the AI

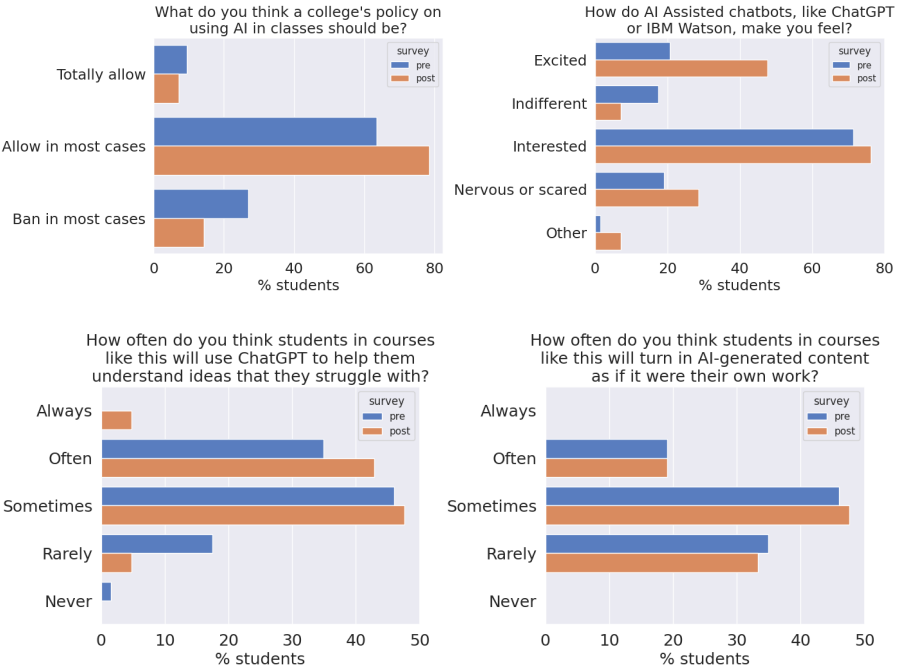


Figure 2: Summary of Student Responses to Survey

solution used a Pandas DataFrame (when the intention was to scan through and filter a list of dictionaries) and included the the operation `filtered_df = df[df["Province"] == selected_state]`, which involves complicated operations that act on the entire Series - something too advanced for students of this level. And yet, the student remarked

> *"I learned about pandas. Pandas has a built-in function called Dataframe that allows you to scan a set of data and makes it much easier to access in the future."*

In this case, we judged that the AI help got in the way of student learning. However, it should be stressed that these were a very small minority of cases.

5 Summary of Student Views

Comparing students' responses to survey questions, summarized in Fig. 2, at the beginning and the end of the semester indicates how their views have

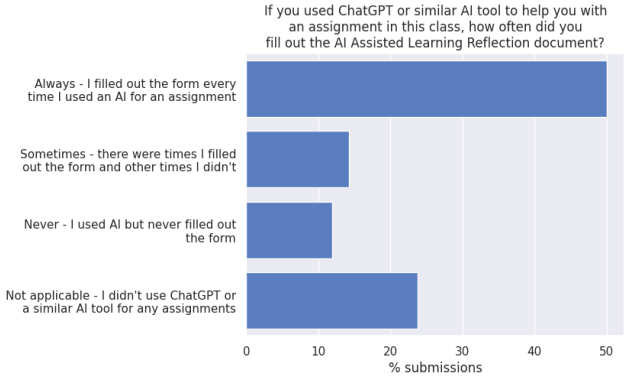


Figure 3: How students reported following the course policy

changed with their experiences with LLMs. One notable difference is their thoughts on the college's policy. At the beginning of the semester, 27% of students selected "Ban in most cases", while at the end of the semester, students seemed to embrace the technology in an academic setting more as only 14% favored banning in most cases.

Another noteworthy difference in the pre and post-semester surveys is the response to the question, "How do AI Assisted chatbots, like ChatGPT or IBM Watson, make you feel?" At the beginning of the semester, only 21% noted "Excited", while at the end of the semester the percentage of "excited" students grew to 48%. Interestingly, there was also an increase in students indicating "Nervous or scared" - perhaps exposure to ChatGPT has made students aware of the potential disruptive nature that these tools introduce to the workforce. Students also seemed to realize positive use cases over the course of the semester, with increases in expected use for understanding and debugging (Q4 data not shown, but saw the "Often" response jump from 30% to 53%).

Lastly, we think it is also remarkable that two thirds of the students thought students would use AI-generated content as if it were their own work at least sometimes, and this perception didn't change much over the course of the semester. When it came to their actual behavior, about a quarter of the students admitted to at least once using AI but not submitting the required form (see Fig. 3). This seems to be in line with the perceptions of some educators who are afraid that the tool will primarily be used in dishonest ways.

6 Conclusions and Future Work

Overall, we found that most of the ways that our students used LLMs in our courses were positive and compatible with learning in CS1 and CS2. Most of them prompted the AI to seek help with debugging or with an isolated problem within a larger project that they needed help with. We also discovered some negative outcomes - that in a small number of cases, students used it in a lazy manner, that it is difficult to avoid the AI giving too much help, and that the AI occasionally gets in the way of learning by presenting solutions above the student's level. Furthermore, some students did not comply with the required course policy to acknowledge and reflect on the assistance they received from the AI - a challenge for teaching students to learn to use AI properly. And, students perceive that their peers are likely to use the technology dishonestly. However, it may be that this is simply a new manifestation of an old problem: convincing students it is in their interest to seek *assistance* rather than *solutions* and to be transparent about sources.

One category of approaches to dealing with these negatives is to *prevent* students from using AI through things like in-class and unplugged assessments. However, for those interested in continuing to explore how AI might enhance student learning, we suggest investigating the following ideas for *mitigating* the negatives:

Establish an easy way for students to cite AI assistance in their code. For example, have students link to the transcript from a code comment (which some web-based services now support). By making it easy, we take away barriers to honesty.

Provide incentives for reflective learning. For example, give some assignment credit for written reflection on how the activities have led to student learning (whether it involved AI or not) rather than exclusively on the student code, testing, etc.

Coach students on good ways to use AI. And then, have them verify it with their citations and reflections - these could be made hard requirements if necessary.

- Have students include their own code/attempts in the AI prompt rather than the assignment text they were given.
- Have students ask follow-up questions on the parts that they do not understand.
- Limit the scope of what they ask of the AI - instead of requesting a solution to the whole problem, they must ask for help with small parts and then integrate solutions into their overall code. This might mean limiting the number of lines of code that can be asked about, but we do

not recommend prompting the AI to limit the number of lines of code - in our experience this makes it more likely for it to generate more advanced, though concise, code.

- Explore prompts that tend to lead to more fruitful conversations. The CS education community should explore and share design guidelines for *prompt engineering*, similar to those being investigated in other disciplines [5].

References

[1] James Finnie-Ansley et al. "The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming". In: *Proceedings of the 24th Australasian Computing Education Conference*. ACE '22. Virtual Event, Australia: Association for Computing Machinery, 2022, pp. 10–19. ISBN: 9781450396431.

[2] Nat Friedman. "Introducing GitHub Copilot: your AI pair programmer". In: *GitHub Blog* (June 2021).

[3] Kalley Huang. "Alarmed by A.I. Chatbots, Universities Start Revamping How They Teach". In: *New York Times* (Jan. 2023).

[4] Majeed Kazemitabaar et al. "Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming". In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–23.

[5] Vivian Liu and Lydia B Chilton. "Design guidelines for prompt engineering text-to-image generative models". In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 2022, pp. 1–23.

[6] Stephen MacNeil et al. "Generating Diverse Code Explanations Using the GPT-3 Large Language Model". In: *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2*. ICER '22. Lugano and Virtual Event, Switzerland: Association for Computing Machinery, 2022, pp. 37–39.

[7] B McMurtrie. "AI and the future of undergraduate writing". In: *The Chronicle of Higher Education* (2022).

[8] OpenAI. *ChatGPT: Optimizing language models for dialogue*. `https://openai.com/blog/chatgpt/`. 2022.

[9] Sida Peng et al. "The impact of AI on developer productivity: Evidence from github copilot". In: *arXiv preprint arXiv:2302.06590* (2023).

[10] Jürgen Rudolph, Samson Tan, and Shannon Tan. “ChatGPT: Bullshit spewer or the end of traditional assessments in higher education?” In: *Journal of Applied Learning and Teaching* 6.1 (2023).

[11] Cliff Saran. “How ChatGPT will become a programmer’s best friend”. In: *ComputerWeekly.com* (Mar. 2023).

[12] Mike Sharples. “New AI tools that can write student essays require educators to rethink teaching and assessment”. In: *Impact of Social Sciences Blog* (2022).

Introducing Controlled Variability in Programming Assignments*

Charles Hoot, Nathan W. Eloe, and Diana Linville
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
{hoot,nathane,dianar}@nwmissouri.edu

Abstract

Evaluating students in Computer Science related fields has always been challenging. Often times programming and problem solving skills are evaluated either with a fully creative (or client driven/capstone) project or a program that generates "the right answers". As class size increases, evaluating the work product of students is more challenging. We illustrate a middle ground approach that allows students in a larger class setting to benefit from some creative choices while still ensuring a minimum level of difficulty and efficient grading processes.

1 Introduction

In appropriate situations, creative assignments allow students to explore a problem space and engage with material in a way that assignments with objectively correct solutions do not; students have fewer restrictions and can create something unique and interesting to them while still demonstrating learning outcomes and acquired skills. Fully creative assignments work well in smaller classes since the extra overhead they create is manageable. Assessing student work on these assignments requires determining whether or not they have met

a criteria instead of producing a single correct answer; this often requires human intervention and additional person hours to determine whether students understand the required material. As classes get larger this becomes infeasible; in a class of 50 students a grader will have to grade 50 unique submissions for every assignment. Common ways to curb this growth have disadvantages; specifically group projects allow students to hide behind the work of others and make it more difficult to assess an individual's mastery of the material. This is less than ideal, especially in introductory or foundational classes where an individual's mastery directly impacts their success in future courses.

As personal computational devices have become ubiquitous instructors have moved more course components online as well [2, 1]. The result is a push to automate as many elements of the assessment process, such as exams [4]. When building the exam, a personalized version can be created via random selection from a question bank to using question templates to generate a problem instance. [3] Administration of the exam can be done on-line, and grading can be automated as well. In computer science, automation can also be introduced in the evaluation of assignments through the use of test cases that submitted code is run against. These tests report for correctness [7]. Unfortunately automation breaks down with fully creative submissions.

This trend to online delivery accelerated with the advent of COVID where over a one year span, most universities moved class materials, assignments, and exams to on-line. While there has been a shift back towards in person courses due to a general disappointment in the learning achievements of students in the on-line format, the interest remains [5, 6].

This paper will look at how variation can be introduced in assignments in a context where submissions are on-line and the students are not in a controlled environment over the duration of the assignment. We present a middle ground where there is variation in assignments that allow students some creative freedom while reducing the overhead of having all unique submissions in large classes. Additionally, we report how these approaches have been used scaled down to smaller class sizes to make sure students are achieving learning outcomes within boundaries. These approaches are not intended to address cheating on quizzes and exams, and indeed the fact that they introduce some small amount of friction to academic integrity issues on assignments is a bonus.

2 Goals

There are many reasons one may want to introduce controlled variability into assignments, and not all reasons may apply to all classes or assignments.

One primary goal of these assignments is to control the base level of difficulty while still allowing a creative element. Fully creative projects can keep

students engaged, but often times requires a proposal process to ensure that students are creating a project with enough rigor given the information they have about the class. Additionally, students often over– or under–estimate the difficulty of the project they decide to undertake. Anecdotally, students need to pivot away to a different project based on unforeseen problems relatively frequently.

If the assignment involves exploring a problem space (see examples below), adding variation in starting points allows the class as a whole to get a bigger picture of reality than if everyone started at the same place. This gives students a chance to share what they've discovered and see how it differs from what other people have found, essentially crowdsourcing the exploration.

Introducing variability into assignments can also add a barrier to academic dishonestly and plagiarism. In cases where students are directed to have different content it can be quickly detected, and requiring students to change what their projects contains from another person's code may make cheating more "expensive" than just doing the assignment on their own. While not the primary driver for choosing to introduce variability into assignments, it has been a benefit.

One of the key factors allowing the use of controlled variations is to have an efficient evaluation process. In the following examples, this is primarily met by having a visual component. This can be in the form of screen shots or through a user interface. This is a quality of life improvement for people grading assignments in large classes, though it does nothing directly for students other than keeping the feedback loop relatively short in large classes.

3 Types of Variability

Over the development of courses that these types of variable assignments are appropriate for, three main types seem to be predominant: *Required Components*, *Fixed Structure Variable Content*, and *A La Carte*. In this section we define these types of assignments and how to evaluate them; examples of each will be given in Section 4.

3.1 Required Components

In this kind of assignment, students are given a list of required skills to demonstrate. This works particularly well when you want to assess syntax and usage of a language or its components. These can often be evaluated with a quick glance through the student's submission of code or the resulting interface to ensure all requirements are met.

The minimum difficulty is controlled by setting the required set of components. Students can feel free to increase the complexity of the assignments by

building out from the core requirements.

3.2 Fixed Structure Variable Content

Assignments in this category give students a set of requirements in the structure of the assignment and give them flexibility in the specific content. Limitations or various requirements can be placed on the content to ensure that there is variability between submissions. This works better than a Required Components assignment when you want students to demonstrate the interaction between certain elements of the assignment (e.g. the nesting of elements in a reactive HTML webpage). As these assignments often involve a GUI of some sort, evaluation can once again be done with a quick look at the code and interface of a given submission.

The overall difficulty is more fixed than in a Required Components assignment, as students aren't given as much leeway to expand on the assignment.

3.3 A La Carte

In these assignments students are given a a menu of options each worth a certain number of points as well as a point total X. They must complete (at least) X points worth of options for the assignment. This can be offered with some bonus credit opportunities so students aren't required to solve the Subset Sum Problem to determine how to earn exactly X points. These work well as expansions to in class tutorials or previous assignments where students have practiced the coding elements and the goal of the assignment is more practice with a tool, not with specific elements of the tool. Depending on what the tool is, it may not be immediately obvious what choices a student made when "ordering" so part of the grading process might involve requiring students to describe the choices they made with the submission.

The minimum difficulty of the assignment is controlled by setting point values for menu items and the total point value. There is nothing keeping students from going above and beyond this minimum difficulty level as long as the choices they made are still apparent in the final submission.

4 Examples

In this section we give example assignments and courses for each of these assignment types. This list is not exhaustive, but intended to give an idea of the types of classes where these could be used.

With respect to class sizes, recently both Web Apps and Operating Systems tend to be larger classes (at least 40 per section), while other classes listed tend to be smaller depending on the semester.

4.1 Required Components

These work well in intermediate or advanced programming classes where an element of the assignment is meant to be review of a programming language or tool.

4.1.1 Example: Personal Project (Web Development II)

Within the Web Development II course, the students work throughout the semester learning HTML and CSS. They are then individually assigned a project where they are asked to apply these concepts and implement a website. They are allowed to choose the purpose(topic) of the website and must include the following components:

- At least 3 pages
- A template file
- One e-mail hyperlink
- One external hyperlink - pseudo-class must be applied
- Consistent banner logo area
- Consistent navigation
- An external style sheet (.css file)
- One Image
- One two-column or three-column layout
- At least 1 media query design applied (tablet or desktop)

Prior to the project beginning students must submit their topic for approval and create the wire–frame for the website. The students use GitHub to publish their websites.The goal of this assignment is to assess that the students can take the concepts learned in the first modules and implement them within a website correctly.

4.1.2 Example: C Review (Operating Systems)

Students use C in this Operating Systems class and were first exposed to the language in the previous course in the sequence. The first assignment (after a quick review of how the language operates and is different from higher level languages they've used) requires the students to write a compiling program that contains the following programming constructs:

- Variable
- `for` loop
- `while` or `do while` loop
- At least one function that returns a value (non-void return type, `main` does not count)
- At least one function that takes at least one parameter (non-void return type, `main` does not count)
- Printing to the screen with `printf`
- Single branch conditional (`if` statement)
- Dual branch conditional (`if else`)

The students are given a set of problems to choose from (`https://projecteuler.net/archives`), but there is no check that their submissions produce the correct answer. The goal of this assignment is to check that they can write a program that compiles and runs; the syntax itself here is what is being reviewed and evaluated, not the logic itself.

4.2 Fixed Structure Variable Content

These assignments are appropriate for classes where the languages or libraries have a certain limitations or interactions between the elements that need to be demonstrated by students.

4.2.1 Example: Mining GitHub (Web Mining)

As an in class activity, the GitHub API is used to determine the most popular and widely used language based on a subgraph generated by looking at stargazers and owned public repositories starting from a seed repository. Students are then asked to pick a unique seed repository and show through graph visualizations what the four most popular (used by most users) and four most widely used (used for most repositories) languages show up in their generated subgraph of GitHub.

This assignment is run as a discussion so the class can build a more complete look of what the current landscape of programming languages is, and how it changes based on what part of GitHub you are investigating.

4.2.2 Example: HTML/CSS Assignment (Web Apps)

In this assignment students are required to create and style a web page where the content is a small chunk of a play. The goal is to introduce students to the application of CSS styles to elements. Each character and stage instruction gets their own style. This could also work with other media forms like opera.

The variation is introduced by creating ten groups of sources (either plays or acts). Students select from the group that corresponds to the last digit of their student ID. The student ID is used as their email address so it is publicly available and should be more or less uniformly distributed.

The major part of grading this assignment is determining if appropriate styles have been applied. This is speed up by viewing the HTML on a browser or through a screen shot.

4.2.3 Example: HTML/JS Assignment (Web Apps)

In this assignment students are asked to create three pieces of content. Each piece of content has an image and an `onClick` Javascript event.

The variation is introduced by having a general category from which students will select specific items for their content. Each of the three items will start with one of the first three letters in their last name. While this does not guarantee a uniform distribution, it creates more pools of content to select from.

Browsing the HTML provides a relatively quick way of checking the content as well as the responses to the clicks. (Note: This was part of a larger assignment that also used variation based on student ID affording more combinations of content choice.)

4.3 A La Carte

These work well in courses that are working with tools that work along side programming languages to produce a result.

4.3.1 Example: Tutorial "Improvement" (Game Dev)

After completing the official tutorial in class for the game engine (currently Godot: `https://godotengine.org/`), students are given an assignment to "improve" the tutorial (with **heavy** air quotes around improve). Any new artwork or music they find must be permissively licensed, or at least legally obtained. They are told to choose 10 required points (with 3 possible bonus points) of the following improvements:

1. Change the animated player sprite and modify the collision box to be appropriate: 1 pt

2. Add or replace at least one enemy type, including animations: 1 pt
3. Change the background to either:
    - an image: 2 pts
    - an animation: 3 pts
4. Keep track of the highest score the player's gotten since the game has opened (no persistence) and display at some point: 3 pts
5. Allow the player to get hit more than once before dying, and indicate the number of lives remaining with either
    - displaying a counter on screen: 2 pts
    - Using icons/visual representation: 4 pts
6. If option 5 is done, make an enemy disappear when it is hit: 1 pt
7. If option 5 is done, add a sound when the player hits an enemy: 1 pt
8. Enable WASD navigation (requires investigating the input map): 2 pts

It's important to note that students must update the README in their game to indicate their choices, as some might not be immediately noticeable to the user. Failure on the student's part to do so will result in a 15% deduction in their grade.

5 Conclusions

In this paper, three basic strategies have been discussed to add controlled variation into assignments and some examples of classes where these approaches are currently implemented have been provided. The goal is to allow student exploration of a problem space without introducing excessive grading overhead for those assessing student work. It has the added benefit of discouraging academic dishonesty by introducing friction in direct copying, though is not intended to be a comprehensive solution to plagiarism. Example assignments have been presented of each type along with discussion of how the variation was introduced. While these assignments cannot really be automatically assessed through unit tests or output testing, the visual element present in many of them affords an efficiency in grading. Techniques for quickly evaluating these kinds of assignments have been addressed in many cases through the examples. While not appropriate for every class or every assignment, this controlled variability can be beneficial to educators and students where the desired outcomes allow for these kinds of projects and assignments.

References

[1] Peter Brusilovsky and Sergey Sosnovsky. "Individualized Exercises for Self-Assessment of Programming Knowledge: An Evaluation of QuizPACK". In: *J. Educ. Resour. Comput.* 5.3 (Sept. 2005), 6–es. ISSN: 1531-4278. DOI: 10.1145/1163405.1163411. URL: https://doi.org/10.1145/1163405.1163411.

[2] Anita M. Krsak. "Curbing Academic Dishonesty in Online Courses". In: 2007.

[3] Amruth Kumar. "Rule-based adaptive problem generation in programming tutors and its evaluation". In: *Proc. of Int. Conf. on Artificial Intelligence in Education*. Citeseer. 2005, pp. 36–44.

[4] Ghader Kurdi et al. "A Systematic Review of Automatic Question Generation for Educational Purposes". In: *International Journal of Artificial Intelligence in Education* 30 (Nov. 2019). DOI: 10.1007/s40593-019-00186-y.

[5] Ansie Minnaar. "A framework for controlling dishonesty in open and distance learning (ODL) in higher education". In: 2012.

[6] Gili Rusak and Lisa Yan. "Unique Exams: Designing Assessments for Integrity and Fairness". In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (2020).

[7] Mohamed Tarek et al. "Review of Programming Assignments Automated Assessment Systems". In: *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*. 2022, pp. 230–237. DOI: 10.1109/MIUCC55081.2022.9781736.

Coding Integrity Unveiled: Exploring the Pros and Cons of Detecting Plagiarism in Programming Assignments Using Copyleaks*

Chandra Mouli Madhav Kotteti, Ratan Lal, Prasad Chetti
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
{chandra, rlal, pchetti}@nwmissouri.edu

Abstract

Before the advent of generative Artificial Intelligence (AI) tools, for example, ChatGPT, students traditionally approached assignment development authentically by employing libraries and by referring to textbooks. However, with the widespread reliance on powerful AI tools for assignment completion, the process has become more convenient. Unfortunately, this ease of use has led to a potential detriment in students' genuine understanding of subjects, as well as a decline in their problem-solving and innovative thinking skills. Moreover, AI tools like ChatGPT will evolve as technology advances such that the need to detect AI-generated content is even more crucial in educational setting to reinforce the value of original work [5]. This paper aims to address this issue by focusing on the detection of plagiarism in student assignments through the utilization of the Copyleaks[1] tool, specifically designed to identify AI-generated code. The accuracy of the tool is systematically evaluated by submitting various pairs of codes, each with similar functionality, wherein one is generated by AI and the other by humans.

[1]https://copyleaks.com/ai-content-detector

1 Introduction

In the ever-evolving landscape of education and technology, the rise of AI has significantly transformed how students engage with their coursework. Nowadays, students increasingly turn to third-party AI tools, such as ChatGPT, to deal with the range of assignments and course objectives. This dependence on AI is not only an indication of the growing integration of technology into educational practices but also poses challenges in ensuring the reliability of student's work [10]. In this situation, the need for robust AI content detection tools becomes essential to maintain academic integrity.

Various AI content detection tools have emerged to address the related challenges posed by using AI in student submissions. Codio[2], for instance, makes code plagiarism detection easier to use, making it possible for educators to spot instances of code copying and cheating within the class using a sophisticated algorithm. Another notable tool is MOSS, short for Measure Of Software Similarity, developed by Standford University, which precisely measures similarity between pairs of code files, employing a document fingerprinting algorithm named winnowing [8]. It goes beyond traditional plagiarism checks by utilizing AI to analyze code structures comprehensively.

JPlag excels in finding similarities among source code files, specifically designed to discourage unauthorized copying of student exercise programs in programming education [7]. Codequiry[3] introduces an AI-driven approach, utilizing machine learning to identify patterns and runs against both internal and external sources to detect all forms of unoriginal code. Additionally, Plagiarism Checker[4] is a cloud-based solution that scans code against over 100 million sources from major repositories and 2 billion instances on the web. These tools collectively contribute to maintaining the integrity of code submissions and promoting fair and honest academic practices.

This study explores the field of artificial intelligence content detection, with a primary focus on Copyleaks. As technology advances, the need to adapt our detection methods to account for AI-generated content becomes evident. Copyleaks stands out as a tool designed to specifically address this challenge, providing insights into potential AI-based plagiarism. The experiment outlined in this paper aims to explore the effectiveness of Copyleaks in detecting AI-generated content and its contribution to maintaining academic integrity. By concentrating on this essential tool, we seek to provide valuable insights into the evolving field of AI content detection, offering educators and institutions an informed perspective on safeguarding the authenticity of student work in an increasingly AI-driven educational environment.

[2] https://www.codio.com/
[3] https://codequiry.com/
[4] https://www.grammarly.com/plagiarism-checker

2 Problem statement

- Comparison of AI-generated code and manual code follows a few approaches i.e., contrasting the code & comments that will be generated by AI. To compare efficiency, measure code metrics like lines of code, memory usage, and execution time. Examine the results and performances considering the established standards. Lexical and syntactic features of code can be useful for distinguishing AI-generated and human-written code [2].
- In some cases, comparing AI-generated code directly to a student's-written code would not match because AI-generated code may not be readable or may adhere to different coding standards. On the contrary, even if it matches technically, it should not immediately be considered plagiarism because any tool is designed in such a way that it checks with chunks or paraphrases [4]. In such cases, a plagiarism checker may produce false positives or false negatives i.e., mark the non-AI generated code as human-written and vice-versa, which is not valid [12]. Nonetheless, it is trivial for any advanced generative AI tool to generate code in several ways incorporating all the techniques, strategies, and methods ever known to computer scientists.

3 Copyleaks

As a plagiarism detection tool, Copyleaks enjoys a compelling reputation. They have also expanded to include AI detection due to the growing need for AI content detection tools. It consists of versatile applications, namely, AI Content Detector, Plagiarism Detector, Codeleaks, Grammar Checker API, Gen AI Governance, and AI Grader. Copyleaks functions by scanning and comparing text across various online sources to identify potential instances of plagiarism. Mainly, it can help educators, especially teachers teaching programming courses, in simplifying the process to check whether a student's code is created by an AI tool or not [1]. Moreover, it can be integrated with most of the LMS systems like Canvas, Moodle, Blackboard, Brightspace, Schoology, and Sakai. Many researchers across the globe published outstanding results declaring that Copyleaks is the most accurate for detecting LLM-generated text [9, 3].

4 Experimental setup

Our experiment focuses on evaluating the AI Content Detector feature of Copyleaks that can detect AI-generated source code, for instance, GitHub Copilot and ChatGPT (including GPT-4) with 99.1% detection accuracy and 0.2%

false positive rate. We considered two dummy student profiles, namely, StudentA and StudentB. Both are given a programming task, where they need to provide solutions for checking a sequence of characters form a palindrome or not, finding n^{th} Fibonacci number, and finding n factorial using Java programming language. Between them, StudentA has provided human-written code and ChatGPT 3.5 is used to generate the solutions of StudentB. The Copyleaks' AI Content Detector is used to examine both students' solutions.

5 Results & Analysis

We used Copyleaks to perform AI Content Detection. We asked it to omit citations and code comments, check both the online resources and its internal database, and set the scan sensitivity to low for a more comprehensive scan.

Table 1: Copyleaks report for StudentA's files.

Source file	AI-Coverage
Fibonacci.java	0
Palindrome.java	0
Factorial.java	0

Table 2: Copyleaks report for StudentB's files.

Source file	AI-Coverage
GPTFibonacciSeries.java	0
GPTPalindromeChecker.java	0
GPTFactorialCalculator.java	100

Tables 1 and 2 show the scan results of StudentA and StudentB solutions, again, StudentA's solutions are human-created and StudentB's are ChatGPT generated. The good outcome is that AI-coverage for the StudentA's solutions i.e., for the 3 Java files, namely, *Fibonacci.java*, *Palindrome.java*, and *Factorial.java* is 0%. However, the tool is not greatly performed when it comes to detecting the AI-coverage for the StudentB's solutions, which shows that only *GPTFactorialCalculator.java* has 100% AI-Coverage and other files have 0%, which is not true.

Since there is a limitation on scanning the same file multiple times on Copyleaks. We renamed the Java files of both the students and performed one more round of scan without any content change. The results are showed

Table 3: Copyleaks report for StudentA's renamed files.

Source file	AI-Coverage
PalindromeRenamed.java	0
FactorialRenamed.java	0
FibonacciRenamed.java	0

Table 4: Copyleaks report for StudentB's renamed files.

Source file	AI-Coverage
GPTFibonacciSeriesRenamed.java	0
GPTFactorialCalculatorRenamed.java	0
GPTPalindromeCheckerRenamed.java	0

in Tables 3 and 4 for StudentA and StudentB, respectively. Again, the AI-coverage for StudentA's renamed files are 0%. However, this time, the AI-coverage for the StudentB's renamed files are all 0%. This proves that there is no guarantee that the scan report results are not accurate if we run the scan on the same document multiple times.

If this tool is used for AI content detection in students' submissions, there is a significant concern on the accuracy of the scan results. For example, in the first round, the *GPTFibonacciSeries.java* and *GPTPalindromeChecker.java* files of the StudentB are marked with 0% AI-coverage, which means that even if a student uses a generative AI tool to complete the assignment, there could be instances, where the student can get away without being caught for AI content detection.

Listing 1: GPTFactorialCalculator

```
package studentB;

import java.util.Scanner;

public class GPTFactorialCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a non-negative integer to
            calculate its factorial: ");
        int n = scanner.nextInt();

        if (n < 0) {
            System.out.println("Please enter a non-negative
```

```
                integer.");
        } else {
            long factorial = calculateFactorial(n);
            System.out.println("Factorial of " + n + " is: "
                + factorial);
        }

        scanner.close();
    }

    private static long calculateFactorial(int n) {
        if (n == 0 || n == 1) {
            return 1;
        } else {
            return n * calculateFactorial(n - 1);
        }
    }
}
```

Listing 2: GPTFactorialCalculatorRenamed

```
package studentB;

import java.util.Scanner;

public class GPTFactorialCalculatorRenamed {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a non-negative integer to
            calculate its factorial: ");
        int n = scanner.nextInt();

        if (n < 0) {
            System.out.println("Please enter a non-negative
                integer.");
        } else {
            long factorial = calculateFactorial(n);
            System.out.println("Factorial of " + n + " is: "
                + factorial);
        }

        scanner.close();
    }

    private static long calculateFactorial(int n) {
```

```
        if (n == 0 || n == 1) {
            return 1;
        } else {
            return n * calculateFactorial(n - 1);
        }
    }
}
```

Moreover, in a scenario, if one student shares source code file with other student, and the other student just renamed the source code file and submitted the assignment, using Copyleaks it is difficult to determine that the other student used AI-generated code because the software may not even detect any AI-coverage in some situations, for instance, even though the content of *GPTFactorialCalculator.java* and *GPTFactorialCalculatorRenamed.java* files are same as shown in code listings 1 and 2, the *GPTFactorialCalculatorRenamed.java* submission has got 0% AI-coverage.

6 Challenges

In our study on AI content detection, several challenges arise that complicate the identification of plagiarism. Firstly, the ability of AI to generate code that mimics human writing style makes manual identification exceptionally difficult as AI tools can produce content with high level of originality [6]. The code produced by advanced AI models closely resembles that of a human, eliminating traditional visual clues that might indicate automated generation. This human-like quality of AI-generated content demands sophisticated detection mechanisms that go beyond manual scrutiny.

Secondly, even when suspicions arise about the code being AI-generated, proving it might be difficult. Unlike traditional plagiarism, where direct evidence or similarities can be highlighted, AI-generated content lacks distinctive markers that unequivocally prove its automated origin. It becomes a challenging task to definitively link a piece of code to AI-generated one using any AI content dectector's report without the student's acknowledgment since AI content detectors as susceptible for false positives and false negatives [11].

Adding more complexity, a tricky situation emerges when students admit to usage of AI tools for coursework but claim innocence regarding plagiarism if at all get caught. For instance, if multiple students input the same statement into a chat-based AI model like ChatGPT and receive identical code outputs that they use for the coursework submissions and they are not regenerating the initial code/response, it becomes challenging to decide that they all have cheated on the coursework item since there solutions may be identical but generative AI tools may generate same response multiple times if the input

statement is the same.

7 Conclusion

In our research endeavor to precisely identify plagiarism in student assignment submissions, Copyleaks assumed a pivotal role. However, its efficacy in discerning the authenticity of AI-generated code has been inconsistent. While there were instances where Copyleaks accurately identified code for certain problems as AI-generated, there were also occurrences where code developed by us for the same problem was wrongly categorized as AI-generated by the tool. These encountered challenges underscore the limitations of relying solely on such tools for unambiguous determination of AI-generated code. In conclusion, despite instances where assignments are flagged as AI-generated, definitive confirmation remains conditional upon the student's acknowledgment of such categorization.

8 Acknowledgements

The author wishes to thank Venkata Rayudu Adapa, Gopi Lokindi, Sri Charan Vattikonda, Sai Dinesh Kopparthi, and Vasavi Devineni who are graduate students in the School of Computer Science and Information Systems at the Northwest Missouri State University, MO, for their assistance in this project.

References

[1] Ruchi Bansal and Sunil Kumar. "A REVIEW ON: PLAGIARISM". In: *World Journal of Pharmaceutical Research* 11.9 (2022).

[2] Sufiyan Bukhari, Benjamin Tan, and Lorenzo De Carli. "Distinguishing AI-and Human-Generated Code: a Case Study". In: (2023).

[3] Chaka Chaka. "Detecting AI content in responses generated by ChatGPT, YouChat, and Chatsonic: The case of five AI content detection tools". In: *Journal of Applied Learning and Teaching* 6.2 (2023).

[4] Ahmed M Elkhatat, Khaled Elsaid, and Saeed Almeer. "Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text". In: *International Journal for Educational Integrity* 19.1 (2023), p. 17.

[5] Simone Grassini. "Shaping the future of education: exploring the potential and consequences of AI and ChatGPT in educational settings". In: *Education Sciences* 13.7 (2023), p. 692.

[6] Mohammad Khalil and Erkan Er. “Will ChatGPT get you caught? Rethinking of plagiarism detection”. In: *arXiv preprint arXiv:2302.04335* (2023).

[7] Lutz Prechelt, Guido Malpohl, Michael Philippsen, et al. “Finding plagiarisms among a set of programs with JPlag.” In: *J. Univers. Comput. Sci.* 8.11 (2002), p. 1016.

[8] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. “Winnowing: local algorithms for document fingerprinting”. In: *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. 2003, pp. 76–85.

[9] Michael Sheinman Orenstrakh et al. “Detecting LLM-Generated Text in Computing Education: A Comparative Study for ChatGPT Cases”. In: *arXiv e-prints* (2023), arXiv–2307.

[10] Levent Uzun. “ChatGPT and academic integrity concerns: Detecting artificial intelligence generated content”. In: *Language Education and Technology* 3.1 (2023).

[11] William H Walters. “The effectiveness of software designed to detect AI-generated writing: A comparison of 16 AI text detectors”. In: *Open Information Science* 7.1 (2023), p. 20220158.

[12] Debora Weber-Wulff et al. “Testing of detection tools for AI-generated text”. In: *arXiv preprint arXiv:2306.15666* (2023).

Engaging Middle Schoolers in Game Programming: A Scratch-Based Workshop Experience*

Abbas Attarwala
Computer Science Department
California State University, Chico
Chico, CA 95973
aattarwala@csuchico.edu

Abstract

This paper presents an experience report on a week-long, remote game programming workshop for middle school students, conducted using Scratch over Zoom. The workshop, designed and led by me (then a faculty member at Boston University in the department of computer science) with the assistance of high school students from Boston University Academy, aimed to introduce middle schoolers to basic programming concepts using game programming. The workshop employed a combination of direct instruction and breakout sessions, where students worked on projects and received personalized guidance.

A unique aspect of the workshop was the participation of industry speakers from prominent organizations like National Instruments, Class Central, Dropbox, and ToothPike Games, providing students with insights into real-world applications of coding. Feedback from participants indicated a highly positive reception, with students enjoying the fun and interactive learning environment, appreciating the hands-on approach, and valuing the supportive nature of the instructor and tutors.

The paper highlights the effectiveness of interactive, project-based learning in engaging young learners in computer science and offers insights for educators looking to implement similar programs.

1 Introduction

The rising demand for STEM (Science, Technology, Engineering, and Mathematics) professionals in the United States contrasts sharply with the stagnating number of graduates in these fields [3, 12]. This discrepancy could stem from a dwindling interest in STEM courses or possibly a lack of early educational opportunities in these areas.

Addressing this gap, some researchers [8] have underscored the significance of computer science (CS) outreach programs, such as summer workshops. These initiatives, which fall under the category of non-formal education, have been notably effective in boosting female participation in CS.

[8] also mentions that activities focused on developing programming skills in short-term interventions have shown promising results in enhancing self-efficacy and interest in pursuing computing careers among young women. Moreover, studies [5, 9] indicate that introducing computing concepts before high school can significantly increase the likelihood of maintaining interest in the subject into higher education.

Middle schoolers stand at a crucial juncture in their educational journey. Often, they are just beginning to explore their interests and abilities, many without prior exposure to programming concepts. Yet, their enthusiasm for interactive and engaging activities, like video games, provides a unique gateway to introduce them to the world of CS and mathematics. Seizing this opportunity, I and high school students from Boston University Academy (BUA) led a CS outreach program in Boston, aiming to ignite a spark of interest in these young minds. The medium of choice was Scratch - a platform that simplifies the complexities of coding into a more accessible and enjoyable format. My goal was not just to impart knowledge, but to kindle excitement and curiosity. I envisioned these students proudly sharing their creations with their families, thereby extending the impact of the workshop beyond the classroom [2, 13].

Another integral component of these workshops was connecting these young learners with real-world professionals. It's one thing to learn about programming in a classroom, but another to understand its application and potential from those actively shaping the industry. Leveraging my network and the resources of BUA, I invited colleagues from ToothPike Games, Class Central, Dropbox, and National Instruments. These guest speakers shared their journeys, struggles, and triumphs in the CS field. The objectives of the guest speakers were to inspire, to motivate, and to provide tangible, relatable role models for these students. I wanted the children to not only learn about CS but to see its potential as a path they might someday choose.

This paper narrates the story of this unique teaching experience, its challenges, successes, and the lessons learned along the way. Section 2 of this paper presents the literature review. In Section 3, I detail my experience on

how I conducted these workshops and share my experience of implementing a specific game with my students. This game was designed to demonstrate the placement of objects on a grid and to simulate the effect of gravity on falling objects. Section 4 presents some of the feedback that I have received from the middle schoolers and my reflections. In Section 5, I discuss potential future work and approaches for conducting a similar workshop in the future. Finally, in Section 6, I present my conclusions.

2 Literature Review

[10] have detailed their journey in establishing a new community outreach program focused on computing. Inspired by their efforts, my ambition at California State University, Chico is to initiate a similar endeavor. This project will actively engage both my undergraduate and graduate students. Moreover, I am considering enhancing future versions of the game programming workshops, previously conducted at BU, by incorporating Lego MindStorms. This idea is encouraged by reports of successful programming education initiatives with young students, including elementary school children [4], and efforts in K-12 outreach promoting interest in Mechanical Engineering [7].

With formal CS curricula in schools currently undergoing significant changes, it's crucial to introduce K-12 students to CS concepts early, aiding them in making informed career choices [1]. Outreach programs, functioning independently or alongside formal education, present an effective solution. [1] compared five such successful programs reveals a common theme: they do not consider programming experience as a prerequisite for engaging with CS concepts. This approach was mirrored in our outreach efforts in Boston's middle schools, where we similarly did not require any prior programming experience for participating in the gaming workshops.

Research by [14] explored the impact of block-based programming tools, such as Hopscotch, on elementary students' attitudes towards programming. Conducted with 4th and 5th graders in the U.S. over a seven-week period, this study assessed changes in perceptions before and after a programming curriculum, finding that block-based programming positively influenced students' views on programming, highlighting its suitability for early education. Similarly, [18] emphasized the effectiveness of programming teaching and scaffolded programming approaches in enhancing computational thinking for K-12 students.

In my gaming workshop for middle schoolers, I selected Scratch for its straightforward, user-friendly approach to programming. This decision was based on the expectation that it would enable effective participation, even for students without prior programming experience. My workshops followed

a similar structure, starting with 75-90 minute guided Scratch sessions. Students were then encouraged to extend their learning in breakout rooms. For instance, after demonstrating how to program a witch flying on a broomstick, the students creatively added new features, such as missile-firing capabilities to combat dragons.

[6] examined the impact of programming camps on middle school students, focusing on their effectiveness in enhancing programming knowledge, computational thinking skills, and attitudes towards computing. Their literature review found substantial use of block programming, highlighting its benefits as a less intimidating introduction to programming, which aligns with the positive feedback I received from students. The study also noted the successful use of project-based approaches in teaching, countering the perception of computer science as isolating. Inspired by these findings, in future I plan to extend my workshops to two weeks instead of one week, allowing students to collaborate with each other and to work on group projects.

3 Gaming Workshops

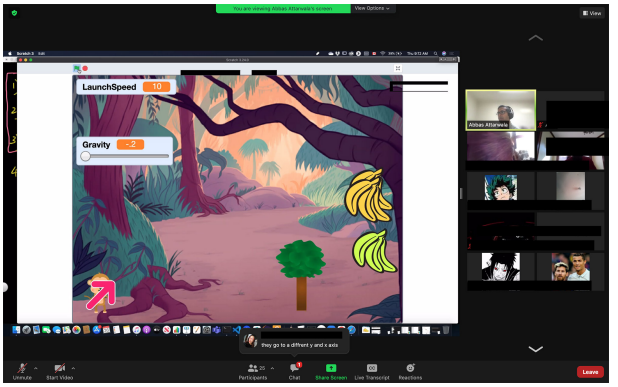
The gaming workshop designed for middle school students spanned five days, with each session lasting approximately 3 to 4 hours on Zoom. I had the support of students from BUA who assisted me in facilitating the workshop. We welcomed around 25 middle school students who registered for the workshop, which was offered at no cost. Each day I demonstrated programming constructs involving variables, branching, loops, cloning of sprites, video motion, using text to speech and sound effects by building a new game. At the conclusion of each day, I arranged for a guest speaker to join and share their personal journey in CS with the students. They discussed what initially sparked their interest in the field, their passions during their own middle school years, and what inspired them to pursue a career in CS. I used the resources [15, 16, 17, 11] when designing and making games.

Every day at the beginning of the workshop, I dedicated approximately 90 minutes to creating a new game on Scratch. Each game was designed to progressively introduce programming concepts like loops and conditional statements to the students. During this time, I engaged in live coding on Scratch, ensuring the session was interactive and responsive to student questions. Following this, students were divided into breakout rooms where they were encouraged to enhance the game by adding new features, working individually. I shared my Scratch project with them as a base for their enhancements. In these breakout rooms, my students from BUA provided assistance and guidance to the middle schoolers with any questions or issues they encountered.

In our workshop, we developed a game titled 'Hungry Monkey' where the

goal was to navigate a monkey to collect bananas while avoiding jungle obstacles like trees. This game, demonstrated in Figure 1 on the left, provided an engaging way to teach the basics of coordinates. The intuitive interface of Scratch allowed students to easily understand that decreasing the Y-coordinate made the monkey descend on the screen.

During our workshop, I engaged the students in a discussion on how to realistically simulate gravity's effect on a falling monkey in the game. The students understood that lowering the monkey's Y-coordinate would mimic the effect of gravity, requiring a negative value for a downward motion. When I adjusted the monkey's Y-coordinate with the *FallSpeed* variable, which was influenced by the *Gravity* setting, the students initially thought this change would occur only once. They quickly realized that to continuously simulate falling, this action needed to be repeated. This was their introduction to the concept of loops in programming, highlighting its importance in creating repeated actions. These discussions and explorations culminated in the development of the block code depicted in Figure 1, effectively producing the desired falling motion in our Scratch game. The exploration continued with a student's insightful question about the implications of a positive gravity value. We experimented by setting a positive gravity, which humorously sent the monkey soaring off the screen, resulting in laughter and an enjoyable learning moment for everyone.



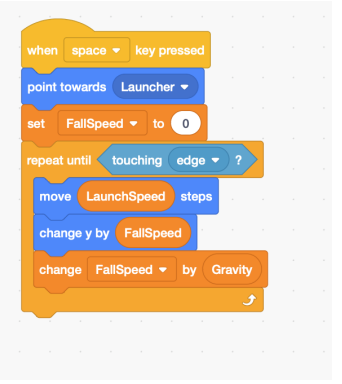


Figure 1: In the left is a Zoom session, where I am demonstrating the 'Hungry Monkey' game, where students can interact with a slider to adjust the gravity parameter and observe its effects. On the right, the displayed block code illustrates how the monkey character falls when the gravity setting is adjusted to a negative value.

4 Student Feedback and Reflections

The workshop culminated with feedback from the participants, providing valuable insights into the effectiveness of the approach. The students, ranging from 6th to 7th grade, expressed a high level of satisfaction, citing the appropriateness of the workshop's difficulty level and the engaging format that combined instruction with hands-on practice. Here are some snippets of feedback that we received via Google form from middle schoolers: About 8 students completed the Google form. Under the question 'What did you like about the workshop', the responses are: (1) *its a fun environment for questions and showing your work that you have worked hard on.* (2) *I like how it's very hands-on and if I or anyone else has a question our question can be answered along with other ones that might have formed while getting the answer and I like creating my unicorn game.* (3) *Everything! I liked how nice the tutors and the professor are. They helped us when we needed help and taught us a lot. It was really fun making games today!* (4) *I liked that we got to make a different game each day.* (5) *The most thing I liked about the program is making games and meeting new people* (6) *How we were able to share our coding skills and games to everyone.*

Under the question, 'Was this program/workshop the right level for me', 7 students responded that it was the right level and 1 responded that it was too easy. Under the question, 'Would you register for this workshop again?', 7 students responded Yes, and 1 responded as 'Not Sure'.

Under the question 'Usually Professor Attarwala would teach during the first half of each day, and then you'd work with tutors during the second half. Did you like this format?', All the 7 students responded Yes.

Under the question 'Is there something you would change about the program?' 5 students responded as 'No' change. Others provided the following feedback: (1) *the one thing I would change about the program is get more breaks* (2) *I wish there was more time to people to work on their own coding projects, not always someone else.*

Under the question 'The camp was one week long, and you met for about three hours every day. What do you think about how long the program was?', 5 students responded that it was perfect and 3 students responded that they wished it was longer.

A consistent theme was the enjoyment of the creative freedom afforded by Scratch programming, with several students enthusiastically sharing their custom enhancements to the projects, such as adding missile-firing capabilities to a broomstick in a game. The integration of guest speakers from the industry, including professionals from ToothPike Games, National Instruments, DropBox and Class Central was met with appreciation, as students valued the real-world connections and stories shared. This element of the program not only enriched the learning experience but also provided students with diverse perspectives

on the field of CS.

In reflection, the feedback suggests that using game programming as an introductory tool for programming concepts can be an engaging and effective strategy for students with little to no prior experience. The learning in breakout rooms with BUA students was also highlighted as a strong point, with students enjoying the ability to showcase their work and receive personalized assistance.

Moving forward, this positive reception encourages the continuation and expansion of such initiatives, with considerations for including technologies like Lego Mindstorms to further enrich the programming curriculum. Additionally, the feedback points to potential improvements, such as extending the program duration, conducting the gaming workshops in person and providing more opportunities for individual project work, which will be taken into account for future workshops.

5 Future Work

In the future, my strategy will encompass the collection of both qualitative and quantitative data. I plan to employ a two-tailed paired t-test to accurately assess the skill development in my middle school students. This will involve conducting an assessment to determine their average (mean) scores at two different points: before starting the workshop and after its conclusion. The objective is to use the paired t-test to compare these average scores from the same group of students, thereby evaluating whether the workshop has led to a statistically significant improvement in their skills.

6 Conclusion

The feedback from our game programming workshop for middle school students highlights the workshop's success in sparking interest in CS. The use of Scratch programming language provided an accessible platform for students to engage creatively with coding concepts, culminating in a positive shift in attitudes towards computing. The introduction of industry professionals as guest speakers further enriched the experience, offering students diverse perspectives and real-world relevance. While the program was well-received, insights for enhancements, such as incorporating technologies like Lego Mindstorms and adjusting program length, have been identified. These findings advocate for the continued integration of innovative, hands-on educational methods in CS education, affirming their potential to inspire the next generation of learners.

Acknowledgement

I gratefully acknowledge the use of OpenAI's ChatGPT for proofreading, grammatical checks, and other text editing tasks. I also like to thank all the wonderful administrative staff, teachers and students at Boston University Academy that assisted me during the gaming workshop.

References

[1] Tim Bell et al. "Overcoming obstacles to CS education by using non-programming outreach programmes". In: *Informatics in Schools. Contributing to 21st Century Education: 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives, ISSEP 2011, Bratislava, Slovakia, October 26-29, 2011. Proceedings 5*. Springer. 2011, pp. 71–81.

[2] Caelin Bryant et al. "A middle-school camp emphasizing data science and computing for social good". In: *Proceedings of the 50th ACM technical symposium on computer science education*. 2019, pp. 358–364.

[3] Bureau of Labor Statistics. *Computer and Information Technology Occupations*. `https://www.bls.gov/ooh/computer-and-information-technology/home.htm`. Accessed on December 10th, 2023. 2023.

[4] Vidushi Chaudhary, Vishnu Agrawal, and Ashish Sureka. "An experimental study on the learning outcome of teaching elementary level children using Lego Mindstorms EV3 robotics education kit". In: *arXiv preprint arXiv:1610.09610* (2016).

[5] Rhonda Christensen et al. "Longitudinal analysis of cognitive constructs fostered by STEM activities for middle school students". In: *Knowledge Management & E-Learning* 6.2 (2014), p. 103.

[6] Carla De Lira, Rachel Wong, and Olusola Adesope. "A systematic review on the effectiveness of programming camps on middle school students' programming knowledge and attitudes of computing". In: *Journal of Computing Sciences in Colleges* 38.1 (2022), pp. 89–98.

[7] Charlotte De Vries et al. "Using LEGO MINDSTORMS in a Control Systems Lab to Impact Next-generation Engineers (Work in Progress)". In: *2015 ASEE Annual Conference & Exposition*. 2015, pp. 26–1670.

[8] Katherine Hiley, Hannah Cebolla, and Mai Elshehaly. "The Impact of Non-Formal Computer Science Outreach on Computational Thinking in Young Women". In: *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 2*. 2023, pp. 642–642.

[9] Linda S Hirsch, Suzanne Berliner-Heyman, and Jacqueline L Cusack. "Introducing middle school students to engineering principles and the engineering design process through an academic summer program". In: *International Journal of Engineering Education* 33.1 (2017), pp. 398–407.

[10] Brian Krupp et al. "CS+ creating a community outreach group in computing from the ground up". In: *Journal of Computing Sciences in Colleges* 35.5 (2019), pp. 51–60.

[11] Craig Steele. *DK Workbooks: Computer Coding with Scratch Workbook*. 1st. Accessed on: December 10th, 2023. DK Children, 2019. ISBN: 1465479287. URL: https://www.amazon.ca/DK-Workbooks-Computer-Scratch-Workbook/dp/1465479287.

[12] Stevens Institute of Technology. *Why Don't Students Stick with STEM Degrees?* https://www.stevens.edu/news/why-dont-students-stick-with-stem-degrees. Accessed on December 10th, 2023. 2023.

[13] Chaoyi Wang and Michael Frye. "Minigems 2018 summer camp evaluation: Empowering middle school girls in steam". In: *2019 IEEE Integrated STEM Education Conference (ISEC)*. IEEE. 2019, pp. 149–155.

[14] Jiahui Wang. "Use hopscotch to develop positive attitudes toward programming for elementary school students". In: *International Journal of Computer Science Education in Schools* 5.1 (2021), pp. 48–58.

[15] Jon Woodcock. *Coding Games in Scratch: A Step-by-Step Visual Guide to Building Your Own Computer Games*. Penguin, 2015.

[16] Jon Woodcock and Steve Setford. *DK Workbooks: Coding in Scratch: Games Workbook: Create Your Own Fun and Easy Computer Games*. 1st ed. Accessed on: December 10th, 2023. DK, 2016. ISBN: 1465444823. URL: https://www.amazon.ca/DK-Workbooks-Coding-Scratch-Workbook/dp/1465444823.

[17] Jon Woodcock and Steve Setford. *DK Workbooks: Coding in Scratch: Projects Workbook: Make Cool Art, Interactive Images, and Zany Music*. Illustrated. Accessed on: December 10th, 2023. DK, 2016. ISBN: 1465444025. URL: https://www.amazon.ca/DK-Workbooks-Scratch-Projects-Workbook/dp/1465444025.

[18] Enwei Xu, Wei Wang, and Qingxia Wang. "A meta-analysis of the effectiveness of programming teaching in promoting K-12 students' computational thinking". In: *Education and Information Technologies* 28.6 (2023), pp. 6619–6644.

Auto-Graded Review Questions: A Modern Take on a Classic Technique*

Jason E. James and Mahmoud Yousef
Department of Computer Science and Cybersecurity
University of Central Missouri
Warrensburg, MO 64093
jejames@ucmo.edu, yousef@ucmo.edu

Abstract

A classic technique of teaching is for the instructor to ask a question to the class, often in an attempt to review some concept and to jog the students' memories. This paper presents a modernized approach to this technique that makes use of technology. In this new version of the technique, each student in the class individually answers the review questions through a computer. The review questions are automatically graded by the system and statistical results are immediately available to the instructor. While the impact of this technique on final grades in courses appears negligible, survey responses overwhelmingly indicate that the students found this approach to be beneficial.

1 Introduction

Introductory computer programming courses are infamous for having a high failure and withdraw rate [2]. A plethora of approaches and experiments have been done regarding this issue. One experiment compared teaching multiple programming languages as opposed to only teaching a single programming language in CS1, which was shown to be effective[6]. Another approach incorporated active learning, which attempts to synthesize putting the student in a

real world scenario and thereby induces greater learning, into the course with positive results [9]. Others have used peer-assisted learning, which strives to bolster learning through the interactions of students helping each other [11]. Yet, others have tried a mixed approach that incorporates collaborative aspects and an emphasis on allowing students to work on individual projects that are meaningful to themselves [5].

Our approach bears some similarities and parallels to clickers. Over the years, a number of works have been published regarding the effectiveness of clickers on learning. Sternberger describes the use of clickers in a constructivist teaching environment and found that, although students perceived the clickers helped their learning, the test scores of the students could have been better [10]. Lantz and Stawiski studied the effectiveness of clickers in a controlled environment [7]. Caldwell looks into the use of clickers in large classrooms and provides tips for the usage of clickers as well as for crafting clicker questions [3]. Premkumar and Coupal also lay out tips for the effective use of clickers [8]. Anderson, Healy, Kole, and Bourne explore efficiency aspects of using clickers [1]. Dong, Hwang, Shadiev, and Chen explore the use of clickers to allow students to pause a lecture [4].

This paper presents an instructional technique that engages students by allowing them to see their own understanding and retention of material as well as giving the instructor a snapshot of the class' understanding and retention.

A classic technique of teaching is for the instructor to ask questions to the class, often in an attempt to review some concept and to jog the students' memories. However, often only a handful of students are actually willing to answer the questions. This result presents a couple shortcomings. First, since only a handful of students are answering the questions, the instructor does not receive a very good sense of how well the class as a whole understands the material. Second, the portion of the class that is not responding to the questions is not being actively engaged and may not even be paying attention.

By having students answer a set of review questions on the computer, each student is given the opportunity to be engaged in the questions and is forced to confront their own recollection and understanding of the material. Since the questions are graded and scores aggregated instantaneously, the instructor, and even the entire class, can easily see the percentages for how well the class responded to a question, allowing the instructor to make clarifications and offer further explanations as necessary.

2 Methodology

The auto-graded review question technique has been used in both a CS1 course teaching structured programming using the Java programming language as well

as a junior level course teaching C programming in the UNIX environment. The fall 2018 semester marks the third semester (fall 2016, fall 2017, and fall 2018) the technique has been used in the CS1 course and fourth semester in the C course (spring 2017, fall 2017, spring 2018, and fall 2018). The review questions are created as a test in the Blackboard course management system for which each student at the institution already has a user account. In the classroom, each student has a computer available to them.

The review questions are prepared prior to class time by the instructor and set to appear on Blackboard at the starting time of the class. Generally, the review questions will cover the lecture or topic from the previous class period. At the start of the class, the students are instructed to work through the review questions while the instructor takes the attendance for the day. Depending on the number of review questions, it usually takes around 5 to 15 minutes for the majority of the students in the class to complete the questions. Once most of the students have submitted their responses to the review questions, the instructor brings up the aggregated statistical results and goes through the questions with the class. Based on the percent of correct or incorrect responses, the instructor can elaborate and clarify on concepts as necessary.

Furthermore, review questions can be configured in such a way so that students can work through the questions multiple times as well as access the questions at a later time, such as when reviewing for an exam. Since the review questions are done as a test in Blackboard, the set of questions can be imported into future sections of the course and reused.

Generally, the review questions are not done for points in the course. By not doing the review questions for points, the need for the instructor to proctor the students is eliminated. This frees the instructor to do tasks such as taking attendance. Additionally, by not making the review questions worth points, students can feel more relaxed. Alternatively, if the review questions were for points, they would essentially become a quiz at the beginning of each class period, which could make the course more stressful.

A survey about the auto-graded review question technique was given to students from three sections of both a CS1 course and one section of a junior level C programming course. Aside from questions about the review question technique, the survey also included some demographic questions and some personality questions. The four sections contained a total of 112 students, of which 78 students consented and responded to the survey.

3 Results

Of the 78 survey respondents, 53 indicated they are male and 25 indicated they are female. The majority of the respondents are in a computing discipline with

47 indicating computer science, 12 indicating cybersecurity, and nine indicating software engineering. Note that it is possible for the same student to be counted multiple times in the major counts (i.e., a student could be both a computer science major and a cybersecurity major). The academic year of the respondents is quite mixed with 27 freshmen, 16 sophomores, 22 juniors, 12 seniors, and one graduate student. Unsurprisingly, all of the freshmen respondents are from the CS1 sections. However, the CS1 sections also comprised a number of sophomore and junior respondents.

Question 3 lists a number of words (i.e., "quiet", "talkative", "reserved", etc.) and asks respondents to mark which words they think describe their personality. A little over half of the respondents marked "quiet" and "reserved."

Question 6 is a multiple choice question that asks students how comfortable they are answering a question asked by the instructor. The responses were an almost perfect bell curve with nine responding Very Comfortable, 16 responding Comfortable, 32 responding Neutral, 15 responding Uncomfortable, and six responding Very Uncomfortable. The large number of Neutral responses could be interpreted that these students, while not uncomfortable with answering questions from the instructor, may not particularly be thrilled to do so.

Question 7 is also a multiple choice question that asks students how often they actually answer questions asked by the instructor. Interestingly, about half the students (42, specifically) responded either Rarely or Very Rarely, 19 responded Neutral, and 17 responded either Often or Very Often. These responses demonstrate that a lot of students are not answering questions posed by the instructor.

A comparison of final grades between sections of the courses that did use the auto-graded review technique and sections that did not use the technique did not show a significant difference. Factors such as variance in students as well as variance in other instructional techniques and grading could partially account for this outcome.

However, student responses in the survey were overwhelmingly favorable towards the auto-graded review question technique. Question 9's text is, "How helpful do you feel the lecture review questions done through Blackboard are?" The text of Question 10 is, "After a set of review questions on Blackboard has been answered, the instructor immediately reviews and explains the answers to the questions to the class. How helpful do you find these reviews and explanations?" Both questions are multiple choice with the following possible responses: Very Helpful, Helpful, Neutral, Unhelpful, and Very Unhelpful.

In the three CS1 sections, the vast majority of students responded to Question 9 with either Helpful or Very Helpful. Specifically, only four responded Neutral. Results were similar for Question 10, nearly all responded either Helpful or Very Helpful, with only five responding Neutral. There were not

responses of Unhelpful or Very Unhelpful to either question amongst the CS1 respondents. These results are illustrated in Figure 1.

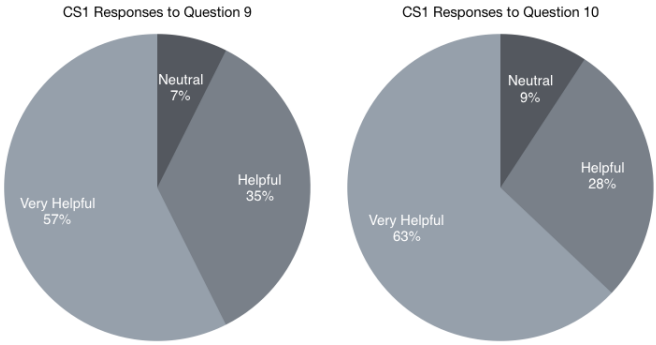


Figure 1: CS1 students responses to Question 9 and Question 10 of the survey.

For the C programming course, results were similar to those of CS1, again with the vast majority of students responding to Question 9 with either Helpful or Very Helpful. Only one participant in this course responded to Question 9 with Unhelpful and likewise only one responded with Neutral. Results were similar for Question 10 in this course as well, with nearly all responding either Helpful or Very Helpful, with only one responding Neutral. These results are illustrated in Figure 2.

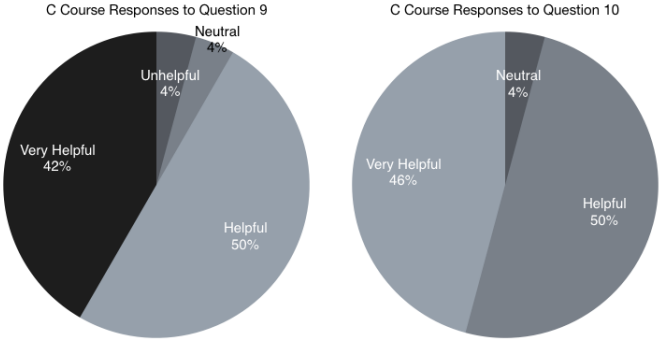


Figure 2: C programming course students responses to Question 9 and Question 10 of the survey.

Question 13 gave the participants a chance to write remarks in their own

words. Specifically, Question 13's text reads, "Please leave any comments that you might have." While not all participants responded to Question 13, the majority of the responses to Question 13 were positive. Here is a sampling of some of the responses to this question:

- "I truly like doing the lecture review questions. I find them very helpful for studying for the concept part of tests. It also helps refresh your mind before the new lecture."
- "I like the review questions and the immediate feedback because it helps me know how I'm doing and not get behind because I immediately know what I'm doing wrong and how to fix it. It also looks similar to the test, so it helps prepare me to do well in that way."
- "Review questions have helped me gauge how I am retaining information and what I need to work on."
- "The review questions help me gauge my own progress in the class, and my understanding of the material."
- "I like that it gives me the opportunity to see if I understood the last lecture or if I need to go back and study it more."
- "I like having review questions at the beginning of class because it really helps me understand what confused me from the previous lecture if I was too shy to ask a question."
- "Review questions tend to help because it refreshes me on content I forgot from the previous class, but it also makes me use that information."

4 Conclusion

As the survey results demonstrate, the students like the auto-graded review question technique. This technique both allows the students to gauge their understanding of the material as well as engages the students. Additionally, the instructor gains a clearer picture of the students' understanding. While the technique has been successfully used in a CS1 course and C programming course, the technique should be generalizable to many other courses as well.

References

[1] Lindsay S Anderson et al. "The clicker technique: Cultivating efficient teaching and successful learning". In: *Applied Cognitive Psychology* 27.2 (2013), pp. 222–234.

[2] Jens Bennedsen and Michael E. Caspersen. "Failure Rates in Introductory Programming". In: *SIGCSE Bull.* 39.2 (June 2007), pp. 32–36. ISSN: 0097-8418. DOI: 10.1145/1272848.1272879. URL: http://doi.acm.org/10.1145/1272848.1272879.

[3] Jane E Caldwell. "Clickers in the large classroom: Current research and best-practice tips". In: *CBE—Life Sciences Education* 6.1 (2007), pp. 9–20.

[4] Jian-Jie Dong et al. "Pausing the classroom lecture: The use of clickers to facilitate student engagement". In: *Active Learning in Higher Education* 18.2 (2017), pp. 157–172.

[5] Cecily Heiner. "Coding, Collaborating, and Creating: A Trio of Pedagogical Practices to Improve Instruction and Retention in CS1". In: *J. Comput. Sci. Coll.* 33.2 (Dec. 2017), pp. 93–99. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=3144645.3144659.

[6] Gongbing Hong et al. "A Multilingual and Comparative Approach to Teaching Introductory Computer Programming". In: *J. Comput. Sci. Coll.* 33.4 (Apr. 2018), pp. 4–12. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=3199572.3199573.

[7] Michael E Lantz and Angela Stawiski. "Effectiveness of clickers: Effect of feedback and the timing of questions on learning". In: *Computers in Human Behavior* 31 (2014), pp. 280–286.

[8] Kalyani Premkumar and Cyril Coupal. "Rules of engagement–12 tips for successful use of "clickers" in the classroom". In: *Medical teacher* 30.2 (2008), pp. 146–149.

[9] Bilal Shebaro. "Using Active Learning Strategies in Teaching Introductory Database Courses". In: *J. Comput. Sci. Coll.* 33.4 (Apr. 2018), pp. 28–36. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=3199572.3199576.

[10] Carol S Sternberger. "Interactive learning environment: Engaging students using clickers". In: *Nursing education perspectives* 33.2 (2012), pp. 121–124.

[11] Matthew F. Tennyson, John Casteele, and Andres Ricardo Pena Morena. "A Study of Peer-assisted Learning in Introductory Programming Courses". In: *J. Comput. Sci. Coll.* 33.5 (May 2018), pp. 55–62. ISSN: 1937-4771. URL: http://dl.acm.org/citation.cfm?id=3204979.3204991.

Teaching functional programming in F# to Grade 9 and Grade 10 students*

Abbas Attarwala
Computer Science Department
California State University, Chico
Chico, CA 95973
aattarwala@csuchico.edu

Abstract

In Fall 2021, I led a series of functional programming seminars in F# for Grade 9 and Grade 10 students at Boston University Academy. These seminars were designed to introduce fundamental concepts of functional programming in an accessible manner, focusing on immutability, pure functions, recursion and higher-order functions. The feedback received from the students highlighted both the successes and challenges of this approach, providing valuable insights for future educational offering of this kind.

1 Introduction

This experience paper details computer science seminars that I conducted in Fall 2021 at Boston University Academy (BUA), where I facilitated one-hour weekly seminars on functional programming for Grade 9 and Grade 10 students using F# programming language. The decision to use F# was influenced by my previous teaching experiences. At the University of Toronto, I taught a third-year programming language course using Haskell. However, Haskell's complexity, particularly with I/O operations and concepts like Monads, can be daunting for students [3]. At Boston University, while teaching OCaml for an equivalent course, students faced challenges with installation and setup in

VSCode. I addressed this by providing a Docker container with OCaml on my GitHub [1], accessible via VSCode's remote container feature.

For the BUA seminars, my objective was accessibility; I anticipated that students might attend using iPads or other mobile devices, where installation could be a barrier. The online F# compiler [4] offered a better experience with features like code completion and auto-complete, essential for beginners, and not found in OCaml's online platforms [8]. I concur with [7] that a programming environment has a huge influence on the perception of the programming language by the users.

I emphasized the pure functional aspects of F#, steering the seminars towards understanding the 'what'—using immutable data and pure functions—and away from the 'how'—the control flow and mutable state found in imperative programming. Most students were novices in programming, so the seminar's goal was to introduce them to CS using functional programming in a stress-free environment, without homework or mandatory assignments.

Half of our time was spent on recursion and higher-order functions, and the rest on functions, lists, and pattern matching. I would introduce practical problems, such as implementing quick sort algorithm in F#, to illustrate these concepts (see Section 3 for more details).

The rest of the paper is outlined as follows. In Section 2, I present a short literature review. Section 3, describes my methodology for teaching quick sort algorithm and higher-order functions. In Section 4, I present the feedback that I received from my students and finally in Section 5, I present my conclusions.

2 Literature Review

When the opportunity arose to conduct a seminar series on CS for 9th and 10th-grade students, I was instantly drawn to the idea of centering it around functional programming. Some of my students had prior experience in programming, having learned Java or Python through personal projects or self-study. However, none had exposure to functional programming. To the best of my knowledge, at that time, BUA did not offer formal computer science education to 9th or 10th-grade students. My initial aim in using the functional programming paradigm was to provide a foundational introduction to programming for my students. Functional programming stands out due to its avoidance of assignment statements, which ensures that a variable's value remains unchanged once assigned. This paradigm is inherently free from side effects, with the primary function of a call being to compute and return a result. The consistency of expression values, safeguarded against side effects, allows for evaluation at any point without influencing the final outcome, offering a stable and predictable framework for learning programming [5].

[2] discuss the benefits of introducing functional programming to high school students, highlighting the use of higher-order functions, recursion, and function composition. While I agree with their points, the paper's prescriptive nature presents challenges. Effectively teaching functional programming requires more than just methodologies; it demands a well-motivated choice of language that resonates with students. Additionally, it's important to cultivate intrinsic motivation among students to engage with and complete assignments. My experience showed that without any incentive for independent study, a significant portion of each seminar needed to be spent reviewing the material from the previous week.

The authors [6] provided two different treatment to computing science students using functional programming and imperative style programming. The authors found that students using functional programming produce better programs with a better structure than students programming in imperative style. Functional programming students were also found to use higher level of abstraction and number of functions in designing their programs. Also the fact that we met for only one hour per week, I aimed for them to understand the representation of computations rather than focusing extensively on processing these computations. However as seen in Section 4, I find mixed results on the uptake of functional programming among my grade 9 and grade 10 students.

3 Teaching of Quick Sort and Higher-Order Functions

In teaching the quick sort algorithm, my emphasis was on understanding the essence of the computation rather than the specifics of its execution. I began with a visual demonstration to explain how quick sort operates when the first element of the list is chosen as the pivot. This helped students grasp the core concept: after the initial partitioning, all elements to the left of the pivot are smaller, and those to the right are larger.

However, students were initially unsure how to separate elements smaller or larger than the pivot. A review of the `filter` function from the previous week's session helped bridge this gap. Once the pivot was positioned correctly, students recognized that the original problem of sorting the list had now been broken down into two smaller problems: sorting the elements to the left and right of the pivot.

This realization sparked an insightful moment for one student, who excitedly suggested that we employ recursion to solve these sub-problems. What remained was to combine the sorted sublist on the left with the pivot and then with the sorted sublist on the right. Thus, we arrived at our complete quick sort algorithm (as seen in code listing 1), which elegantly demonstrated the use of recursion and partitioning using higher-order function `filter`.

Listing 1: Quick sort in F# for sorting distinct elements

```
let rec quickSort = function
| [] -> []
| pivot :: tail ->
    let smaller = List.filter (fun x -> x < pivot) tail
    let larger = List.filter (fun x -> x > pivot) tail
quickSort smaller @ [pivot] @ quickSort larger
```

I then presented my students with a function in F# named `medianOfSortedList`, which calculates the median of a sorted list of integers. I posed a challenge to them: given a list containing multiple lists of integers, how would they compute the median of each individual list? The task was to formulate their approach as an algorithm, first using plain English.

The majority of the students suggested a straightforward process: extract each list, sort it using the quick sort algorithm, and then apply the `medianOfSortedList` function to the sorted list to find the median.

Our previous lessons on list traversal using the `map` function came in handy here. Most students understood that the `map` function was essential, and proposed that it should be used to feed each list into `quickSort`, and subsequently pass the sorted output to `medianOfSortedList`. Despite this, they hit an impasse on integrating these steps into a cohesive solution.

At this juncture, I noticed some tension among my students, stemming from the challenge of translating the 'what' into the 'how'. To alleviate this, I reminded them that `map`, which we could use to traverse each of the list of integers and apply some action on it so that it can achieve the desired outcome. Assuming the presence of a predefined variable `listOfListOfInts`, our approach was as follows (see code listing 2):

Listing 2: Traversing a list of lists of integers and applying an action to each sublist.

```
List.map someActionOnListOfInt listOflistOfInts
```

In an effort to concentrate on the process of manipulating each list of integers, I introduced a function named `someActionOnListOfInt`. This abstraction directed our attention to the method by which we could process each list. We delved into defining `someActionOnListOfInt`, which the students understood should first invoke `quickSort` on the list and then pass its sorted output to the median-finding function. This conceptual understanding led us to the following implementation (see code listing 3):

Listing 3: Utilizing the function composition operator to sort a list of integers with quicksort and then find its median.

```
let someActionOnListOfInt =
medianOfSortedList << quickSort
```

<< is a function composition operator in F# that takes the output of `quickSort` and feeds it as input into `medianOfSortedList`. A few of my students found this example helpful, as it aligned with their pre-calculus studies on function composition, offering a practical application of the concept.

4 Student Feedback and Reflections

At the seminar's conclusion, the administrative staff at BUA collected student feedback. This section presents a subset of qualitative responses and summarizes the quantitative data in Table 1. The insights gained from the students are invaluable for planning potential modifications to the seminars for future Grade 9 and Grade 10 cohorts.

A key takeaway is the value of teaching functional programming. It encourages students, especially those new to programming, to develop a structured and systematic approach to problem-solving, enhancing their overall programming skills. Focusing on recursion and immutability has proven effective in helping students reason about their code without the added complexity of side effects. However, it became apparent that I need to better articulate the reasons for choosing F# as the medium for teaching functional programming. Feedback revealed a preference for Python among many students, likely due to its accessibility and the possibility that they had either learned it independently or worked on personal projects using the language. This is an important consideration for future courses, where integrating functional programming concepts with Python might provide a more relatable and reinforcing learning experience.

Additionally, I observed that hosting a one-hour seminar weekly without supplementary assignments does not sufficiently motivate independent study. Extending the seminars to 90 minutes could afford more time for thorough reviews, which are essential for connecting new concepts with previously covered material. With only 60 minutes, substantial seminar time was dedicated to reviews, leaving limited opportunity to introduce and delve into new topics.

Here is a sample of the qualitative feedback received from the students:

1. *"I enjoyed learning about how math related to CS."*
2. *"To be quite honest, just learning about F# as a whole was very intriguing since I had no prior knowledge in coding."*
3. *"I think I learned to think outside of the box and apply math to CS."*

4. *"Not really a skill but was exposed to more advanced coding for the first time."*

5. *"I learned a lot about functions and how they work in F#. As a whole, I learned that there are multiple ways to go about solving a particular problem."*

6. *"I've learned a lot of mathematical skills and new ways of thinking."*

7. *"Because I don't plan on studying CS in the near future, but I am still somewhat interested in it, I gave the course a 3."*

8. *"I would change the language to Java or Python or something more universally known and applicable."*

9. *"Do something other than F#."*

10. *"The programming language is not very useful in my opinion although learning CS is."*

11. *"While the course did go over basic coding skills, it was done in F#, a coding language that isn't seen much and isn't very useful. If this course was done in Python or in JavaScript, this course would have been much more useful and relevant."*

12. *"While I think that the professor and the class was great, the language that we were learning is not commonly used (the professor just said this was his favorite language)!"*

and here is Table 1 summarizing the quantitative feedback across 26 participants.

Statement	**Average**	**Standard Deviation**
The topics covered in the course were relevant and useful. (1= strongly disagree; 5=strongly agree)	2.8846	0.7656
I found the material in this course interesting. (1= strongly disagree; 5=strongly agree)	3.0385	0.7736
The teaching style and classroom activities were engaging. (1= strongly disagree; 5=strongly agree)	3.6154	0.9414

Table 1: Survey Results based on feedback from 26 participants.

5 Conclusion

In conclusion, the F# seminars at BUA offered a first introduction to functional programming for Grade 9 and 10 students. While the approach was well-received by some, the mixed feedback highlights the need for a more familiar language like Python in future offerings of this seminar. This experience underlines the importance of aligning programming tools with student familiarity and the potential benefits of extending session times for deeper engagement and understanding.

Acknowledgement

I gratefully acknowledge the use of OpenAI's ChatGPT for proofreading, grammatical checks, and other text editing tasks. I also like to thank all the wonderful administrative staff, teachers and students at Boston University Academy.

References

[1] Abbas Attarwala. Live coding in the classroom: Evaluating its impact on student performance through ANOVA and ANCOVA. In *2023 IEEE International Conference on Information, Electronic and Intelligent Research (IEIR)*, pages startPage–endPage. IEEE, 11 2023.

[2] B Bantchev. Functional programming for the high-school curriculum. *Math. and Education in Math*, 32:305–311, 2003.

[3] Manuel MT Chakravarty and Gabriele Keller. The risks and benefits of teaching purely functional programming in first year. *Journal of Functional Programming*, 14(1):113–123, 2004.

[4] F# Software Foundation. Try F# — online F# compiler. `https://try.fsharp.org`, 2023. Accessed: December 12, 2023.

[5] John Hughes. Why functional programming matters. *The computer journal*, 32(2):98–107, 1989.

[6] Stef Joosten, Klaas Van Den Berg, and Gerrit Van Der Hoeven. Teaching functional programming to first-year students. *Journal of Functional Programming*, 3(1):49–65, 1993.

[7] Phil Molyneux. Functional programming for business students. *Journal of Functional Programming*, 3(1):35–48, 1993.

[8] OCamlPro. Try OCaml — online OCaml compiler. `https://try.ocamlpro.com`, 2023. Accessed: December 12, 2023.

Inclusive Practices and Universal Design in the Computer Science Classroom*

Meredith Moore and Timothy Urness
Computer Science
Drake University
Des Moines, IA 50310
{meredith.moore, timothy.urness}@drake.edu

Abstract

Universal Design is the principle for designing products to be usable by all people. In this paper, we discuss how universal design can be incorporated into the traditional computer science classroom by intentionally considering how classroom exercises, assignments, discussions, and designing of code can make for an inclusive learning environment. We describe Universal Design for Learning and give examples of effective practices including course design, classroom environment, equitable participation, lecture structure, assignments, and in-class exercises.

1 Introduction

Universal Design for Learning (UDL) is a framework for considering pedagogical strategies to reduce barriers and support all learners [5]. The goal of UDL is to make learning inclusive – eliminating possible barriers so that students of all abilities can be successful without any additional accommodations. Building courses with universal design in mind leads to an education that is accessible by design, rather than modification.

The discipline of computer science is in high demand and is poised to help solve many different kinds of problems. It is important to incorporate UDL,

as the future of computing will be shaped by those people who feel welcomed into the discipline[1]. Instructors in computer science courses have the potential to create environments that will foster a diverse and inclusive learning environment that maximizes the potential of individual students. As educators, it is our charge to increase the effectiveness of education and the diversity of students that contribute to developing technology [7].

1.1 Contributions

The paper is structured as follows: We first discuss the background and significance of UDL in education, highlighting the challenges that students often face. Next, we discuss the core principles of UDL and demonstrate how UDL can be adapted to the computer science context. Lastly, we list several effective practices that align with universal design in computer science, including discussions around the design of the overall course, classroom, lectures, in-class exercises, and assessments.

This paper will provide some simple, straightforward ideas that are easy to implement and will assist professors in better-adapting course curriculum and in-class practices to incorporate UDL in the classroom.

1.2 Background and Significance

In any college classroom, the needs of potential students are wildly diverse. Disability is just one of many characteristics that a student might possess. For example, students may be of a certain gender, race, age, reading ability, hearing ability, or any other number of factors that make them vulnerable to imposter syndrome or needing accommodations to participate or achieve in the classroom. Universal Design (UD) requires consideration of all characteristics of potential users, including abilities and disabilities, when developing a course or service. UD can be applied to any product or environment [1].

When it comes to utilizing pedagogical practices in a higher-education classroom, educators often face several challenges. The advent of new technologies and the diversity of the technologies require educators to put a significant amount of work into maintaining a consistent learning environment. Classes that adopt a student BYOD (bring your own device) policy must choose tools that are platform independent, and these options may be limited. Furthermore, learning environments likely vary from classroom to classroom; in which seating arrangements, size of classroom, availability of technology, size of projector screens and a number of other factors may not be consistent. It is therefore common for teachers to establish their own practices individually. As a result, the frequency and consistency of technology usage depends solely on the

[1] https://udl4cs.education.ufl.edu/

given teacher's interest, which leads to sporadic and inconsistent integration, particularly with respect to practices designed to accommodate UDL [6].

Despite challenges, a dedicated and strategic approach to UDL principles ensures that students with a wide range of abilities and needs can succeed. The framework of UDL consists of instructional approaches that provide students with choices and alternatives in the materials, content, tools, context, and supports they use [4].

2 Effective UDL Practices for Computer Science

Universal design makes it so that the course is designed specifically to be accessible to all students. We make several recommendations for considerations to lead to a more accessible classroom environment. The applicability of these suggestions may vary based on a particular learning management system or technologies that a university has adopted, a classroom's layout, and a professor's lecture delivery style. However, the following suggestions can be helpful in adopting a more inclusive approach to classroom instruction centered in UDL. We describe the suggestions within each section in order of lowest effort and highest reward; they are as follows:

2.1 Inclusive Course Design Practices

Before a student steps into a classroom, several considerations can be implemented that will help make every class session more inclusive.

2.1.1 Record Lectures for Flexible Learning

Creating a recording of the lecture (e.g. the audio recording of the presenter alongside the video of the projected content) can be easily done with freely available software (e.g. Zoom). By providing students with these materials after each class session, a student who is hard of hearing or missed a part of the class for any reason would have the opportunity to learn the material as presented. It also provides students who may not be native speakers with a way to rewatch the lectures, possibly at a slower pace. Furthermore, this provides means for students who are traveling due to school-sponsored functions (e.g. sporting activities) or ill to keep up with the course content. While not a replacement for attending class, recording the lectures (and making them available via request) is a means for making the material more accessible and less dependent on delivering content that must be digested by all students in the same way.

Recording lectures can be as simple as selecting the record option on a virtual software session when the class begins. Rather than sharing the "live"

link with the class, the professor can be the only one in the virtual room.

This recording process has the added benefit of allowing students to join remotely should they choose–something that supports another UDL principle, flexibility in attendance. In our experience, withholding the link to the hybrid session unless individual students ask for it is a reasonable practice to keep students from joining remotely if they are able to participate in the class in person.

2.1.2 Closed Captioning

Even if a person is not hard of hearing, there are many benefits from having closed-captions streaming via a presentation. Current presentation software can do an excellent job of approximating captioning without requiring a professional stenographer. For example, Google Slides and PowerPoint Live both have options for live captioning.

Providing closed-captioning helps more than deaf or hard of hearing students, it also helps anyone who may have difficulty processing auditory information, have a language barrier, or have difficulty attending to material that is presented in only one modality.

2.1.3 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is the process that converts an image of text into a machine-readable text format. This is particularly important for learners who are visually impaired as they will be unable to utilize a screen reader for content that is not OCR-compatible. If necessary, most campus libraries can make suggestions on finding resources to make your non-OCR'd texts into more accessible OCR'd versions.

2.1.4 Make Reading More Accessible

Students with disorders that affect their ability to read can greatly benefit from the use of tools like Beeline Reader[2], a tool that utilized color gradients to guide the eyes of the reader. Another great resource to know about is the free font OpenDyslexic[3]. OpenDyslexic is designed with dyslexic readers in mind, making the bottom part of the text slightly wider than the top part of the text. This difference makes reading easier for people with dyslexia.

These are both tools that don't require much from the professor's end rather than just showing students that these resources exist, and maybe including them in a folder of accessibility tools in a learning management system.

[2]https://www.beelinereader.com/
[3]https://opendyslexic.org/

2.2 Inclusive Classroom Design Practices

While each classroom may be physically different, the following are recommendations that can be done regardless of the setup that can help students during each class session. These suggestions include ideas to help guide transitions from different learning modalities as well as techniques for classroom participation.

2.2.1 Addressing Sensory Preferences

It is common for students to have sensory preferences that they may or may not feel comfortable sharing [9]. Often, students who are neurodiverse have strong preferences for the lighting and sounds in their environment [2]. Some students struggle with abrupt transitions in lighting or sound. Experiencing these abrupt transitions could cause some students to become dysregulated, which often results in difficulties staying focused, generally inhibiting learning.

It is important to consider these sensory preferences, especially when making changes to your classroom environment. We have found that simply giving students a warning when you are about to change the lighting conditions, or if you are about to play an audio clip that may be loud goes a long way to help students feel safe, seen, and heard in their classroom environment.

2.2.2 Leveraging Music for Contextual Shifts and Time Management

Another way that sound has played a role in making our classrooms more accessible is by using music in class to help signal context shifts. One method we have used is playing quiet, relaxed 'coffee shop' music in the background when students are supposed to be working together, and then turning the music off when it is time for them to pay attention to the professor. This method has helped modulate the attention of the class and makes shifting contexts between collaborating and listening much less abrasive.

Time blindness, also known as time perception deficit, is a common trait among neurodiverse students [8]. Students who experience time blindness are unable to sense when time has passed and estimate the time needed to get something done. An added advantage of having a song playing in the background is that helps students with time blindness keep track of the amount of time passing during in-class activities. Using a song's length as a measure of how much time students have to complete the exercise can be a useful tool. For example, "I'll give you two songs to see if you can figure out how to ..."

2.3 Inclusive Lecture Design Practices

Each class period provides opportunities to inform and inspire students. The following recommendations provide ideas to help ensure students are engaged throughout the entire class.

2.3.1 Break Lectures into "Mini-Lectures"

Striving to break up a class session into a several approximately fifteen-minute "mini-lectures" can be an effective way to keep students engaged. Utilizing an engaging activity between each of these mini-lectures provides time for students to practice what was just taught; additionally, the professor can collect feedback on how the content presented in the mini-lecture was understood.

Examples of engaging activities could be a polling question or a brief coding exercise. If the students can complete the exercise with ease, then it's a good sign that they're ready for the next mini-lecture; however, if they are struggling to complete the exercise, then it might mean the professor should revisit the topic before continuing with presenting new material.

2.3.2 Reduce Ambiguity of In-Class Exercises

Ambiguity leads to avoidance. When the instructions for a given in-class exercise are not clear, students will avoid digging into the exercise often for fear of completing it incorrectly. It is important to include clear, unambiguous instructions for each in-class exercise. These instructions should include precise explanations of the context around the question, as well as what and how the student is expected to produce.

Another subtle, yet effective tool in helping students transition between lecture-mode and exercise-mode is the use of a different colored slide for any in-class exercise. These slides can also have a header that defines what type of in-class exercise students should expect: discussion, short exercise, challenge exercise, or poll question.

2.3.3 Provide Extra Challenge Exercises

One of the difficulties encountered in teaching most computer science classes is the broad spectrum of experience and abilities. The learning rate is quite different between students, which leads to some students getting bored while other students get frustrated by feeling rushed or inadequate. One solution that we have explored is the practice of providing extra challenge questions at the end of most presentations.

These challenge questions are tougher questions that we wouldn't necessarily expect all of the students in the class to get to. The goal of including the

challenge questions must be explained clearly so as not to make students who aren't quite ready for the challenge questions feel like they also belong in the classroom. These challenge questions help keep students who have a strong understanding of the material engaged, while also being able to keep students who may need a little extra support from feeling frustrated or rushed.

2.4 Inclusive Assessment Design Practices

It is important that we have accessible ways to assess the understanding and mastery of the concepts that we teach in class. The following lay out some ideas as to how to make computer science assessments accessible to all students.

2.4.1 Clarify Assignment Goals: Provide Objectives and Prerequisites

Including the assignment's objective and pre-requisites at the top of each assignment can help students clearly understand the goal of an assignment and what they need to know before attempting the assignment.

This approach works particularly well with the practice of making all of the assignments for a course visible on the first day of the course. Being clear about the objective of the assignment can help students understand the purpose, and connect the purpose back to the material they are learning in the class. Including the prerequisite knowledge for a given assignment also helps students understand if they are ready to begin the assignment or if they need to go back and do some learning before attempting the assignment.

2.4.2 Implement Incremental Assignment Deadlines

A simple technique that can help students learn to manage their time more effectively is to not require entire assignments due at a long-range deadline. Instead, providing students a short deadline for which a first attempt must be submitted can help students navigate time management issues.

With the increase in usage of auto-grading systems for coding assignments, it is easier for students to get immediate feedback and incorporate this feedback into their submissions. The main goal is to help students understand that using the autograder to test their code is something that is better done early on in the development process rather than at the last minute. It also may help students alleviate anxiety through good time management and avoid procrastination.

2.4.3 Balancing Creativity and Accessibility in Assignments

Some students are driven by being given creative tasks. Asking them to come up with something original, or apply a computer science concept to another

topic that they are interested in, can lead to some student's best work. Creativity often motivates students to push beyond the boundaries of what they've learned in class. However, other students find these kinds of creative assignments to be overwhelming.

Building assignments to allow students the option of thinking creatively can lead to students who are motivated by creativity, while also not pressuring students who may find generating new ideas to be overwhelming or more time-consuming than they would like. For example, an open-ended assignment to build a dice-rolling game (focused on using concepts learned in class) could be an excellent assignment for students who are motivated by creativity. However, this assignment could be made more accessible by providing the rules to a default game for students to implement if they don't want to come up with an original idea. This default option will likely help some students to feel less overwhelmed by such an assignment.

2.5 Inclusive Exam Design Practices

2.5.1 Remove Timed Elements from Exams

In assessing students, it is important to utilize an assessment that is consistent with course goals. If the goal is to ensure students are competent programmers, a time-pressured paper-and-pencil exam with vocabulary terminology assesses a very different set of skills when opposed to a project-based assessment with a longer-range timeline. When selecting how students will be evaluated, it is important to keep in mind the diversity of test-taking abilities and how these map to the intended outcomes of a course.

2.5.2 Reduce Testing Anxiety

Test anxiety has a negative effect on students, and is believed to affect female students more than male students. In one survey, the researchers estimated that 38.5% of students (30% of male students, and 46.3% of female students) experienced testing anxiety [3, 10].

While there are many different variables when it comes to giving tests in a computer science classroom, as educators there are ways to construct assessments that can reduce testing anxiety. One such strategy for in-person classes is to hand out the tests at the beginning of class, ask students to put any writing utensils away, and then allow the students five minutes to look through the test together and brainstorm. In five minutes, there isn't much time to share entire solutions, but rather the overall idea of what each question is asking. This also might bring up some questions as to how the questions should be interpreted, which is most helpful to discover sooner rather than later.

Another common strategy to help reduce test anxiety is to allow the students to bring in one page of notes. This practice not only helps the students have a clear goal while studying, but it also helps reduce the anxiety that students need to memorize every detail of the material that was presented to them to do well on the exam.

Another option is to make tests fully asynchronous – giving students online take-home, open-note tests rather than timed, closed-note, in-person exams. While this is the most universally designed and inclusive pedagogy for exams, the propensity for academic integrity violations increases with these kinds of exams as there is no proctor. Especially with the rise of generative artificial intelligence tools to help students, asynchronous exams are difficult to ensure that the students' work is their own.

3 Conclusion

The discipline of computer science is in high demand and is poised to help solve many different kinds of problems. It is important to incorporate UDL in today's classroom, as the future of computing will be shaped by those who feel like they belong in the discipline.

As educators, we aim to instruct, challenge, and inspire students to learn elements of computing that will enable them to contribute accomplishments beyond the classroom. By following effective practices for Universal Design, we can maximize our impact by ensuring that courses, lessons, in-class activities, and assessments can reach and inspire as many students as possible.

In this paper, we supplied several promising practices and ideas to inspire educators to adopt techniques that will make their activities adhere to UDL practices and ultimately increase their impact on current and future students. While we acknowledge that implementing all of these suggestions is likely not attainable, any additions that adhere to UDL practices will lead to a more inclusive learning experience.

References

[1] Sheryl Burgstahler. "Universal Design: Implications for Computing Education". In: *ACM Trans. Comput. Educ.* 11.3 (Oct. 2011). DOI: 10.1145/2037276.2037283.

[2] Maria Clince, Laura Connolly, and Clodagh Nolan. "Comparing and Exploring the Sensory Processing Patterns of Higher Education Students With Attention Deficit Hyperactivity Disorder and Autism Spectrum Disorder". In: *The American Journal of Occupational Therapy* 70.2 (Jan. 2016), 7002250010p1–7002250010p9. ISSN: 0272-9490. DOI: 10.5014/ajot.2016.016816.

[3] Travis G Gerwing et al. "Perceptions and Incidence of Test Anxiety". In: *Canadian Journal for the Scholarship of Teaching and Learning* 6.3 (2015), p. 3.

[4] Margo Izzo, Alexa Murray, and Jeanne A. Novak. "The Faculty Perspective on Universal Design for Learning". In: *The Journal of Postsecondary Education and Disability* 21 (2008), pp. 60–72.

[5] Hayley Leonard. "Universal design for learning in computing". In: *Hello World Magazine* 15 (2021).

[6] Cristina Mercader and Joaquın Gairın. "University teachers' perception of barriers to the use of digital technologies: the importance of the academic discipline". In: *International Journal of Educational Technology in Higher Education* 17.1 (2020), p. 4.

[7] Meredith Moore and Timothy Urness. "Strategies for Equitable Participation in an Introductory Computer Science Course". In: *Journal of Computing Sciences in Colleges* 38.4 (2022), pp. 48–57.

[8] Vahid Nejati and Samira Yazdani. "Time perception in children with attention deficit–hyperactivity disorder (ADHD): Does task matter? A meta-analysis study". In: *Child Neuropsychology* 26.7 (2020), pp. 900–916.

[9] Katherine Nelson et al. "Impact of Sensory Processing Preferences on First-Year College Students' Success". In: *The American Journal of Occupational Therapy* 76.Supplement_1 (2022), 7610505047p1–7610505047p1.

[10] Marıa Isabel Núñez-Peña, Macarena Suárez-Pellicioni, and Roser Bono. "Gender differences in test anxiety and their impact on higher education students' academic achievement". In: *Procedia-Social and Behavioral Sciences* 228 (2016), pp. 154–160.

Using Generative AI to Design Programming Assignments in Introduction to Computer Science *

Nifty Assignment

Rad Alrifai
Department of Mathematics and Computer Sciences
Northeastern State University
Tahlequah, OK 74464
alrifai@nsuok.edu

Abstract

Programming stands as an essential requisite in computer science education. Recognizing the challenges students face in learning programming effectively, the proposed assignment aims to integrate generative artificial intelligence (AI) tools to teach students introductory programming constructs. Generative AI has gained an increasing popularity in recent years. Several available Generative AI implementations can now help students learn programming essentials and debugging skills.

1 Introduction

The objective of this assignment is to familiarize students with the introductory programming constructs in C++, including if-else statements, loops, and functions. Students will use generative AI to solve a simple problem and develop programming skills. According to Alvarez et. al., "in introductory computer science (CS1) courses in higher education, approximately one in every three students fails. A common reason is that students are overwhelmed by an accelerated and inflexible pace of learning that jeopardizes success" [1]. The difficulty to learn programming is also cited by others. According to Thune and Eckerdal, "Previous research shows that many students find it difficult to

learn computer programming" [2]. Thus, the proposed assignment uses generative AI to encounter some of the common challenges that students experience in introductory programming classes.

2 Assignment Description

In this assignment, students will write a program to calculate a student's total grade based on the grade evaluation weights in table 1. The program will prompt the user to enter the scores for three courses: Computer Programming 1, Discrete Mathematics, and Calculus 1. It then validates, calculates, and displays the total grade. The programming assignment must be completed in C++ by using if else statements, loops, and functions. Students are required to use generative AI to complete the assignment.

Deliverables	**Weight(%)**
Homework Assignments (16 totals, drop the lowest)	45%
Quizzes (3 total)	15%
Midterm	15%
Final	20%
Attendance and Active Participation	5%

Table 1: Grading Evaluation

3 Student Responsibility

Students are responsible for completing the following tasks independently:

- Before working on the assignment, the student needs to comprehend the functional requirements, input, and output specifications.
- Collect data of the required types for input to the program.
- Provide generative AI with specific instructions to code individual components of the assignment.
- Examine the generated code to identify opportunities for improvement, and make necessary changes.
- Integrate individual programming components, generated by open AI.

- Use an integrated development environment (IDE) to identify and fix bugs in the generated code.
- Use an IDE to run the code.
- Examine the output for completion and accuracy.
- Verify that the program meets all the specified requirements.
- Develop test cases using equivalence classes and boundary value analysis, a common technique in software testing. Test the code with the test cases to verify its accuracy and correct any errors.
- Provide clear and concise documentation of the code by adding comments to explain the purpose of different variables and sections.
- Reflect on this programming experience by identifying programming skills impacted by the use of AI, and the limitations of relying on AI for programming.

Students can use generative AI to successfully complete the following tasks:

- Generate some of the initial code.
- Explore alternative ways to code the assignment.
- Generate examples for coding different programming constructs.
- Identify possible errors in the code and suggest improvements.

4 Conclusion

The proposed assignment introduces the use of Generative AI to learn fundamental programming skills and address common programming challenges. While generative AI can automate many coding tasks, developing complete software projects often requires unique skills and creativity that require human involvement. In introductory programming classes, students can use generative AI for several reasons: enhanced productivity and the ability to gain immediate feedback. Also, it's critical for computer science graduates to acquire skills in using the most recent tools essential for success after graduating. Programming will continue to be an important skill for students to have in the foreseeable future, and generative AI can provide a valuable complement to existing resources available to students to learn programming in introductory computing courses.

References

[1] Claudio Alvarez, Maira Marques Samary, and Alyssa Friend Wise. Modularization for mastery learning in cs1: a 4-year action research study. *Journal of Computing in Higher Education*, pages 1–44, 2023.

[2] Michael Thuné and Anna Eckerdal. Analysis of students' learning of computer programming in a computer laboratory context. *European Journal of Engineering Education*, 44(5):769–786, 2019.

The Interplay of 2D Arrays and Nested Loops*

Nifty Assignment

Cong-Cong Xing[1], *Jun Huang*[2]
[1]*Department of Mathematics/Computer Science*
Nicholls State University
Thibodaux, LA 70310
cong-cong.xing@nicholls.edu
[2]*Dept. of Electrical Engineering and Computer Science*
South Dakota State University
Brookings, SD 57007
huangj@ieee.org

Motivation

Using nested loops to access the contents of 2D arrays row-by-row is the standard practice/technique in CS1 teaching. However, with a bit of variation in the loops and/or a twist of the order of index-controlling variables, things can become tricky and/or unexpected and thus pose a challenge to CS1 students. As such, being able to correctly recognize the interplay between nested loops and 2D arrays should be an essential benchmark for measuring the quality of students' grasp on 2D arrays. This nifty assignment piece aims to help students' learning in this respect by providing a group of exercises.

The Problem/Assignment

The following 8 (related) Java program fragments are given to CS1 students as an assignment with the stipulation that `a[][]` is a `3x4`[1] properly initialized

[1]The size can actually be `nxm` for any n and m with $n < m$.

2D integer array, and that `i` and `j` are both program-wide[2] `int` variables with initial value 0. The questions are what would be printed from each of the code fragments and an explanation if the code causes an error.

```
// (1)
for (i=0; i<3; i++)
{
   for (j=0; j<4; j++)
      System.out.print(a[i][j]+" ");
   System.out.println();
}
```

```
// (2)
for (i=0; i<3; i++)
{
   for (j=0; j<4; j++)
      System.out.print(a[j][i]+" ");
   System.out.println();
}
```

```
// (3)
for (j=0; j<4; j++)
{
   for (i=0; i<3; i++)
     System.out.print(a[i][j]+" ");
   System.out.println();
}
```

```
// (4)
for (j=0; j<4; j++)
{
   for (i=0; i<3; i++)
      System.out.print(a[j][i]+" ");
   System.out.println();
}
```

```
// (5)
for (j=0; i<4; j++)
{
   for (i=0; i<3; i++)
      System.out.print(a[i][j]+" ");
   System.out.println();
}
```

```
// (6)
for (j=0; j<4; j++)
{
   for (i=0; j<3; i++)
      System.out.print(a[i][j]+" ");
   System.out.println();
}
```

```
// (7)
for (i=0; j<3; i++)
{
   for (j=0; j<4; j++)
      System.out.print(a[i][j]+" ");
   System.out.println();
}
```

```
// (8)
for (i=0; i<3; i++)
{
   for (j=0; i<4; j++)
      System.out.print(a[i][j]+" ");
   System.out.println();
}
```

All code fragments are related to one another in the sense that one is a variation of another. However, each code fragment produces a different result. Code fragment (1) simply prints out the array row-by-row. Code fragment (2) is obtained from (1) by switching the positions of `i` and `j` in `a[i][j]`, and causes an array index out of bound error at run time after printing out the first column. This is due to the fact that `j` dictates the row index now but eventually reaches 3 because of the inner loop. Code fragment (3) is obtained from (1) by switching the inner loop and the outer loop, and prints the array column-by-column. Code fragment (4) is obtained from (3) by switching the positions of `i` and `j` in `a[i][j]`, and will cause an index out of bound error due to `j` (similar to (2)). However, since the loop involving `j` sits outside, code fragment (4) actually prints out the left 3x3 subarray row-by-row prior to `j` reaching 3 and causing the exception.

If we consider code fragments (1)-(4) "conventional" or "standard", then code fragments (5)-(8) can be "unconventional" or "non-standard" where extra

[2] `i` and `j` are purposefully declared as variables outside the loop so that the complications shown in this assignment can occur for learning/training purposes.

thoroughness is needed when examining them. Code fragment (5) is obtained from (3) by *intentionally* changing `j` to `i` in the condition of the outer loop (some students think this is a typo and is done "mistakenly"). While we can see that in this case the value of `i` will never be equal to or larger than 4 so the outer loop potentially will turn into an infinite iteration, we need to remember that the value of `j` increases by 1 every time the outer loop runs and thus will reach 4 quickly. Hence the result of this code fragment is *not* an infinite iteration, but an array index out of bound error following a printout of the entire array column-by-column. Code fragment (6) is obtained from (3) by "mistakenly" typing `i` as `j` in the condition of the inner loop. Similar to (5), although the inner loop condition `j<3` will be always true, the result of this code fragment is not an infinite iteration, but a printout of the first column followed by an index out of bound error caused by `i`. Code fragment (7) is obtained from (1) by "mistakenly" typing `i` as `j` in the condition of the outer loop. Interestingly, unlike others, (7) printouts the first row and then terminates *successfully*. Code fragment (8) is obtained from (1) by "mistakenly" typing `j` as `i` in the condition of the inner loop. Again, no infinite iteration will occur in this case although `i<4` is always true. The code causes an index out of bound error (due to `j`) after printing out the first row.

Classroom observations

These exercises were designed for and given to an introductory Java programming class as a homework assignment. Students' primary responses were that this assignment had clarified many things, and that they had learned significantly through doing this assignment and participating in the ensuing discussions. We claim that this assignment is nifty because it has revealed many issues that would have been hidden otherwise. A few of these issues are listed below.

- Some students thought that code fragment (5) runs into a never-ending infinite loop, which is not true as explained above.
- Some students thought that the trouble in code fragment (4) is caused by `i` in the following way: `i` in this case dictates the column index, the array has 4 columns, but the maximum allowed value of `i` can only reach the 3rd column. So the array would be "under-run". They failed to realize that "under-running" an array is okay but "over-running" is not.
- Some students thought that code fragment (5) prints out the array column-by-column successfully. They realized that `i` controls the row index and the array has only 3 rows but somehow thought that `i<3` and `i<4` assure that `i` will not be over 3 so that situation is "safe" (i.e., `i` will not get out of bound). They failed to realize that `i<4` is a tautology in this case

which can potentially make the loop run forever (and subsequently get into the first issue above).

- Some students thought that `i` and `j` in `a[i][j]` "represent" (or "are") the dimensions of the array and used that to judge the condition of the for-loops. For instance, they thought that `j` in code fragment (6) "represents" the column dimension of the array (4 in this case), and since 4 is not less than 3, so `j<3` is broken causing an index out of bound error. This, unfortunately, is a total mess-up, and shows the complications this assignment may lead to.

We hope that this assignment can be found useful by colleagues.

Teach Me Video Project*

Nifty Assignment

Wen-Jung Hsin
Computer Science and Information Systems
Park University
Parkville, MO 64152
wen.hsin@park.edu

Objective: In psychology, the *protégé effect* refers to the situation in which an individual's comprehension of information is enhanced by engaging in teaching that information to others [1]. In this project, you will try to understand a topic in Discrete Mathematics thoroughly and produce a power point presentation and a video recording (between 5 and 6 minutes in length) to teach other people the concept.

Format: Individual or 2-person group project

Steps for producing a PPT and a video recording:

(1) By the end of week 8, choose a topic in Discrete Mathematics and submit it to the LMS submission box. See a list of possible topics below.

(2) By the end of week 9, obtain the topic approval from the instructor.

(3) By the end of week 11, submit a PPT describing the topic.

(4) By the end of week 12, obtain the instructor feedback for improvement.

(5) During weeks 13 and 14, make the necessary improvement to your presentation by taking the instructor's feedback into account. Produce a video recording between 5 and 6 minutes in length to explain the concept. (Zoom is a great tool to produce such a video recording.)

(6) By the end of week 14, submit the video recording and the improved PPT to the LMS submission box.

(7) During week 15, your PPT and video recording will be made available to the entire class.

Grading: Appendix A provides the grading rubrics.

Possible topics:

(1) Logic Equivalence, Logic Reasoning

(2) Proof by Contradiction

(3) Simplification using Set Identities

(4) Properties of Binary Relations

(5) Algorithm Analysis

(6) Finite State Machine

(7) Recurrence Relation

(8) Induction Proof

(9) Pigeonhole Principle

(10) Bayes Theorem

(11) Graph Connectivity

(12) Graph Coloring

(13) Tree Traversal

(14) Minimum Spanning Tree

(15) Other topics in Discrete Mathematics

References

[1] The protégé effect: How you can learn by teaching others. `https://effectiviology.com/protege-effect-learn-by-teaching/`. Accessed 2024-01-13.

Appendix A

Teach Me Video Project - Grading Rubric

Criteria	Rating				Points
PPT Content	4 pts Substantial, accurate, interesting, creative, and showing thorough understanding of the topic.	3 pts Do not fulfill 1 of the following attributes: substantial, accurate, interesting, creative, and showing thorough understanding of the topic.	2 pts Do not fulfill 2 of the following attributes: substantial, accurate, interesting, creative, and showing thorough understanding of the topic.	0 pt Lack 3 or more of the following attributes: substantial, accurate, interesting, creative, and showing thorough understanding of the topic.	4 pts
PPT Organization	1pt Organized, and logically sequence		0 pt Lacks one or more of the following attributes: Organized, and logically sequenced.		1 pt
PPT Mechanics	1 pt Clear writing, and no grammatical error.		0 pt Lacks one or more of the following attributes: Clear writing, and no grammatical error.		1 pt
Video Content	4 pts Substantial, accurate, interesting, creative, and showing thorough understanding of the topic.	3 pts Do not fulfill 1 of the following attributes: substantial, accurate, interesting, creative, and showing thorough understanding of the topic.	2 pts Do not fulfill 2 of the following attributes: substantial, accurate, interesting, creative, and showing thorough understanding of the topic.	0 pt Lack 3 or more of the following attributes: substantial, accurate, interesting, creative, and showing thorough understanding of the topic.	4 pts
Video Organization	1pt Organized, and logically sequenced		0 pt Lacks one or more of the following attributes: Organized, and logically sequenced.		1 pt
Video Mechanics	1 pt Clear narration, no grammatical error, and high video quality.		0 pt Lacks one or more of the following attributes: Clear narration, no grammatical error, and high video quality.		1 pt
				Total	12 pts

Cyberpalooza: Experiences Presenting Cyber/CS Subjects to High School Students*

Panel Discussion

Charles Hoot, Nathan Eloe,
Matthew Schieber, Zhengrui Qin
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
{hoot, nathane, mschieb, zqin}@nwmissouri.edu

1 Summary

In November 2023, our university hosted high school students from the area for a one day program focuses on cyber computing with four 40 minute sessions. Demand was high enough that a second day was added. Combined, there were 340 students from 29 schools. Students were broken up into eight groups which were assigned to a set of sessions. All students did sessions on (1) Hands-On Coding, (2) Cracking Passwords, and (3) Phishing/Internet Safety. The fourth session depended on the group and was one of (4) Cryptography Unplugged, (5) Robotics, or (6) Virtual Reality/3D Printing. This panel addresses the planning and goals for the program. It will compare and contrast the sessions along with survey results from the students and session materials as available. To conclude, lessons learned and thoughts for the future will be presented.

2 Biographies

Charles Hoot is an Assistant Professor of Computer Science with interests in software development and theoretical computer science.
Nathan Eloe is an Associate Professor of Computer Science and teaches IoT and other core computer science courses.

Matthew Schieber is a Senior Instructor of Computer Science and specializes in introductory courses.
Zhengrui Qin is an Associate Professor of Computer Science and coordinates the cyber security program

Getting Started with Large Language Models for the CS Curriculum*

Conference Workshop

Eric D. Manley
Department of Mathematics and Computer Science
Drake University
Des Moines, IA 50311
eric.manley@drake.edu

With the introduction of ChatGPT in late 2022, popular interest in language-based Artificial Intelligence has exploded. Employers are looking to hire computer scientists who can leverage large language models (LLMs) [2], and student demand for learning about them at many higher education institutions has followed. This one-hour workshop will help computer science educators respond to this demand by introducing the Python *transformers* library and its associated LLM ecosystem [1]. We will discuss how LLMs can be integrated into college computer science curricula from CS 1 through advanced courses in Artificial Intelligence, Machine Learning, or Natural Language Processing. Specific topics include

- Using the *transformers* library with pre-trained models for inference tasks like sentiment analysis, text classification, summarization, translation, and question answering in *only a few lines of code*
- Searching for and using hundreds of thousands of different pre-trained language models hosted by Hugging Face along with datasets that they can be tested on
- Utilizing conversational models to build chat bots

Furthermore, we will briefly discuss the process for fine-tuning LLMs on new data sets and share the following resources:

- Example code for fine-tuning LLMs on new data sets suitable for using with upper-level CS courses
- A repository with code and presentation materials for an undergraduate Natural Language Processing course that utilizes the tools discussed in this workshop
- Additional resources for continued learning about LLMs.

All code for the workshop can be run using free, cloud-based tools, so attendees need not prepare anything other than bringing an Internet-connected laptop.

References

[1] Hugging Face. *Transformers*. Accessed: December 12, 2023. URL: `https://huggingface.co/docs/transformers/index`.

[2] Chloe Taylor. "Gen Z is set to capitalize on AI skills gap as workers want to learn, but businesses aren't teaching, finds new report". In: *Fortune* (May 2023). Accessed: December 12, 2023. URL: `https://fortune.com/2023/09/05/ai-skills-gap-randstad-research-gen-z-chatgpt-bard-training-llm-generative-ai-salary-jobs/`.

Ethical Considerations in Embracing Generative AI in Higher Education*

Conference Workshop

Maria Weber, Annamaria Szakonyi,
Tatiana Cardona, Dhananjay Singh
School of Professional Studies
Saint Louis University
Saint Louis, MO, 63103
{maria.l.weber, annamaria.szakonyi, tatiana.cardona, dsingh21}@slu.edu

Since November 30, 2022, ChatGPT (Chat Generative Pre-trained Transformer) has been embraced by 180 million users [1]. Companies are reimagining how work can be done by adopting new technologies, such as Artificial Intelligence (AI), to existing or new systems. Similarly, Higher Education is looking for ways to integrate Generative AI into current curriculums because nearly half of college students have used AI models to search for information, learn new topics, brainstorm, or request mentoring from this 24 x 7 online tutor [2]. However, one of many challenges in this adoption is understanding the ethical considerations. Therefore, the core topic of the workshop is to understand the opportunities and concerns surrounding the use of AI in education through case scenarios using ChatGPT, resources, libraries, SDKs, and APIs. We will delve into data privacy, bias in algorithms, and the responsibility of educators and technology providers to ensure ethical AI integration. generative AI-based approaches.

Privacy and data security concerns regarding using Generative AI models should be considered when creating accounts, disclosing personal identifiable information (PII), and proprietary data. Case studies of data breaches and sensitive data leaks will open discussions and raise awareness.

Bias in algorithms created with Generative AI models is another challenge area that needs to be studied since they are not bias-free. The created content should be carefully reviewed and critically analyzed. Participants will be

exposed to different Generative AI models to compare outcomes and discover the different approaches in each algorithm.

Generative AI, too, could be used to build students' critical thinking and reflection if students are taught prompt engineering strategies properly. We'll show how to use a variety of practical and engaging prompts to use ChatGPT. We will also analyze the output given using principles of evidence-based decision-making to measure if the information provided by the model is a hallucination or reliable. Finally, participants will learn how to design and create a personalized GPT. We will create assignments and hands-on activities that encourage students' interaction with this tool and explore the veracity of the information provided to enhance the student's skills for future job-related functions.

References

[1] Omar Ali, Peter Murray, Mujtaba Momin, and Fawaz S Al-Anzi. The knowledge and innovation challenges of chatgpt: a scoping review. *Technology in Society*, page 102402, 2023.

[2] Jane Nam. 56% of college students have used ai on assignments or exams. https://www.bestcolleges.com/research/most-college-students-have-used-ai-survey/.