The Journal of Computing Sciences in Colleges

Papers of the 40th Annual CCSC Eastern Conference

October 18th-19th, 2024
Mount St. Mary's University
Emmitsburg, MD

Abbas Attarwala, Editor
California State University, Chico

George Dimitoglou, Regional Editor
Hood College

Volume 40, Number 3 **October 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:

Serving the CCSC: These members are serving in positions as indicated:

Bin Peng, Associate Editor, bin.peng@park.edu, Computer Science and Information Systems, 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.

Ed Lindoo, UPE Liaison, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, Denver, CO 80221.

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
Code Grade
GitHub
CodeZinger
ACM CCECC

Local Vendors

Evapco
Jones and Bartlett Learning

Welcome to the 2024 CCSC Eastern Conference

Welcome to the Consortium for Computing Sciences in Colleges (CCSC) Eastern's 40th Annual Regional Conference, hosted this year by Mount Saint Mary's University in Emmitsburg, MD. We are so happy to welcome all of you to Mary's Mountain for this important anniversary in CCSC Eastern's history.

On behalf of the CCSC Eastern Region and the program committee, we greatly appreciate all the submissions for papers, posters, workshops, tutorials, and nifty ideas. We've been working since the beginning of the year to prepare for the conference. We have participants coming from at least 30 different organizations, including a national publisher and a local business.

While the CCSC committee assessed the materials to ensure the highest quality of the materials selected for presentations and demonstrations, the Mount faculty, staff, and students worked to prepare the locations for the event. We hope you enjoy the activities planned in our brand-new extension to our building as well as the paper sessions throughout the first floor of Coad.

It is also a great honor to have Dr. Adrienne Decker, previous chair of SIGCSE, as the conference's keynote speaker, and Dr. Gary McGraw, co-founder of the Berryville Institute of Machine Learning, as our banquet speaker. We are so pleased to have both of them share their experiences and thoughts for the future with us.

This year, the conference's steering committee received tremendous interest in paper submission, and the committee members assembled in person at the Mount this past summer to finalize acceptance decisions. We received 27 faculty papers and 20 were accepted (74%) after double blind peer review. We also have 10 faculty posters. Student submissions are at 10 papers and over 16 posters. We also accepted 1 panel on multidisciplinary approaches to AI, 1 workshop on curricular design for liberal arts computing programs, and 3 Nifty Ideas/Lighting Talks. We're excited about the wide variety of topics covered at the conference.

Lastly, as co-chairs of this year's conference, we are very grateful and honored to work alongside the committee and the Mount team to ensure this prestigious regional forum for computing sciences is a productive experience. We look forward to meeting you at the Mount and greatly appreciate your participation and attendance.

Ruth Lamprecht
Scott Weiss
Conference Co-Chairs and Hosts

2024 CCSC Eastern Conference Committee

Co-Chairs
Ruth Lamprecht .... Mount St. Mary's University
Scott Weiss .... Mount St. Mary's University

Papers
George Dimitoglou, Regional Editor .... Hood College
Gregory Liu .... Bay Atlantic University
John Wright .... Juniata College

Posters
Ian Finlayson .... University of Mary Washington
Karen Anewalt .... University of Mary Washington
Pranshu Gupta .... DeSales University

Panels, Workshops, and Tutorials/Nifty Ideas
Nooh Bany Muhammad .... Frostburg State University
Jennifer Polack .... University of Mary Washington

Speakers
Melissa Stange .... Laurel Ridge Community College

Programming Contest
Steven Kennedy .... Frostburg State University
David Hovemeyer .... Johns Hopkins University
TJ Highley .... La Salle University
Jorge Silveyra .... Lafayette College

ConfTool Registration and Submission System
Pranshu Gupta .... DeSales University

Regional Board Representative
Michael Flinn .... Frostburg State University

Regional Treasurer
Nathan Green .... Marymount University

Web Site

John Wright . . . . Juniata College

10, 23, 81 — Stacking up the LLM Risks: Applied Machine Learning Security*

Banquet

Gary McGraw, Ph.D., Berryville Institute of Machine Learning

I present the results of an architectural risk analysis (ARA) of large language models (LLMs), guided by an understanding of standard machine learning (ML) risks previously identified by BIML in 2020. After a brief level-set, I cover the top 10 LLM risks, then detail 23 black box LLM foundation model risks screaming out for regulation, finally providing a bird's eye view of all 81 LLM risks BIML identified. BIML's first work, published in January 2020 presented an in-depth ARA of a generic machine learning process model, identifying 78 risks. In this talk, I consider a more specific type of machine learning use case—large language models—and report the results of a detailed ARA of LLMs. This ARA serves two purposes: 1) it shows how our original BIML-78 can be adapted to a more particular ML use case, and 2) it provides a detailed accounting of LLM risks. At BIML, we are interested in "building security in" to ML systems from a security engineering perspective. Securing a modern LLM system (even if what's under scrutiny is only an application involving LLM technology) must involve diving into the engineering and design of the specific LLM system itself. This ARA is intended to make that kind of detailed work easier and more consistent by providing a baseline and a set of risks to consider.

Speaker's Bio

Gary McGraw is co-founder of the Berryville Institute of Machine Learning where his work focuses on machine learning security. He is a globally recognized authority on software security and the author of eight best selling books on this topic. His titles include Software Security, Exploiting Software, Building Secure Software, Java Security, Exploiting Online Games, and 6 other books; and he is editor of the Addison-Wesley Software Security series. Dr. McGraw

has also written over 100 peer-reviewed scientific publications. Gary serves on the Advisory Boards of Calypso AI, Legit, Irius Risk, Maxmyinterest, and Red Sift. He has also served as a Board member of Cigital and Codiscope (acquired by Synopsys) and as Advisor to CodeDX (acquired by Synopsys), Black Duck (acquired by Synopsys), Dasient (acquired by Twitter), Fortify Software (acquired by HP), and Invotas (acquired by FireEye). Gary produced the monthly Silver Bullet Security Podcast for IEEE Security & Privacy magazine for thirteen years. His dual PhD is in Cognitive Science and Computer Science from Indiana University where he serves on the Dean's Advisory Council for the Luddy School of Informatics, Computing, and Engineering.

Transforming Grading Practices in the Computing Education Community*

Keynote

Adrienne Decker, Ph.D., University at Buffalo, NY

It is often the case that computer science classrooms use traditional grading practices where points are allocated to assignments, mistakes result in point deductions, and assignment scores are combined using some form of weighted averaging to determine grades. Unfortunately, traditional grading practices have been shown to reduce achievement, discourage students, and suppress effort to such an extent that some common elements of traditional grading practices have been termed toxic. Using grades to reward or punish student behavior does not encourage learning and instead increases anxiety and stress. These toxic elements are present throughout computing education and computer science classrooms in the form of late penalties, lack of credit for code that doesn't compile or pass certain unit tests, among others. These types of metrics, that evaluate behavior are often influenced by implicit bias, factors outside of the classrooms (e.g., part-time employment), and family life situations (e.g., students who are caregivers). Often, students in these situations are disproportionately from low-socioeconomic backgrounds and predominantly students of color. In this talk, I will present a case for adoption of equitable grading practices in computer science classrooms and issue a call for additional support in classroom and teaching technologies as well as support from administrations both at the department and university level. I will explain the community of practice approach we are taking to both encourage adoption and to study the impact of these practices on students.

Speaker's Bio

Adrienne Decker, PhD, is an Associate Professor in the Department of Engineering Education at the University at Buffalo. She has been studying the teaching and learning of computing concepts for over 20 years. Her work strives

to enrich and improve the student experience in computing education, including work in broadening participation. She has secured $3M in funding for her scholarly work in computing and has authored over 50 publications in computing education. Her currently funded work includes, a 5-year grant to help integrate more equitable computing practices into computer science classrooms, a grant to help develop additional classroom tool support for adoption of equitable grading practices in computer science, a grant to study threshold concepts for intermediate computer science students, a grant to continue the development and dissemination of subgoal labels in the introductory curriculum, and a grant to help prepare faculty to manage neurodiversity in their classrooms. Active in the computing education community, she is currently serving on the ACM Special Interest Group on Computer Science Education (SIGCSE) board as past chair (2022-2025) and was SIGCSE board chair (2019-2022), and board treasurer (2016-2019), program co-chair in 2014 and general co-chair in 2015 for the SIGCSE Technical

Symposium on Computer Science Education and sits on the ACM SIG Governing Board Executive Committee as SIG Viability Advisor. She has been actively involved with the Advanced Placement Computer Science A course since 2011, first serving as a reader, and as part of the development committee for the CSA exam from 2015-2021, serving as higher ed co-chair 2018-2021, and is one of the principal authors of the current Course and Exam Description for CSA. She has also served on various other program and review committees.

AI Intersections: Ethics, Education, and Technological Philopsophy*

Panel Discussion

Michael J. Reno[1], Victoria Russell[2] Taylor J. Nutter[3], P. Anand Rao[4] and
Jennifer Polack (Moderator)
[1]Classics, Philosophy, and Religion
[2]Center for Teaching
[4]Department of Communication & Digital Studies
University of Mary Washington
Fredericksburg, VA 22401
{mreno, vrussel3, arao}@umw.edu
[4]Theology Department Mount St. Mary's University
Emmitsburg, MD 21727
t.j.nutter@msmary.edu

1 Summary

This panel explores the multifaceted intersections of artificial intelligence with ethics, education, and philosophical perspectives on technology. As AI continues to reshape our world, it becomes increasingly crucial to examine its implications across various disciplines. Our panelists will present diverse viewpoints, ranging from innovative pedagogical approaches using AI to philosophical inquiries into the nature of intelligence and technology. The panel will address critical questions surrounding AI explainability, the integration of AI in education, the historical context of AI research, and the ethical considerations that arise from these technological advancements. By bringing together experts from computer science, philosophy, religious studies, and digital humanities, this panel aims to foster a rich, interdisciplinary dialogue on the present and

future of AI in academia and society. In the spirit of the panel topic, this abstract was created using Anthropic's Generative AI platform, Claude.

2 Dr. Michael J. Reno

Classics, Philosophy, and Religion : Using a GPT to Facilitate Student Understanding of Difficult Texts in the Humanities

In this paper, I summarize my work on "Heidegger-bot" using OpenAI's "Create a GPT." I describe a bot designed to advance student understanding of a difficult philosophical piece. I highlight elements from the instructions used in its creation, describe student reactions and feedback, and offer some notes for further development of these sorts of "tutor-bots" in the context of Humanities education.

3 Dr. Victoria Russell

The Soul of Teaching: Harmonizing Relationship-Rich Education with A.I. Innovations

In an era when technology is rapidly transforming education, Dr. Russell will explore the delicate balance between artificial intelligence and the 'human touch' in the classroom. From how we educate and learn about A.I. as faculty to the ways we use it in our classrooms and beyond, this presentation will consider how we innovate while keeping essential human elements of empathy, creativity, and connection in teaching and learning.

4 Dr. Nutter

Will survey the history of the ideas that are presupposed in AI research and cognitive science. This includes discussing the idea that intelligence is a matter of computation over representations, how this has impacted our understanding of what learning and, therefore, education is, and what questions regarding justice are caught up in the history of these ideas.

5 Dr. P. Anand Rao

Communication and Digital Studies: Promoting AI Pluralism: A Debate Model Approach to AI Explainability

This paper proposes a novel approach to AI explainability through the promotion of AI pluralism using a debate model. By leveraging diverse AI agents in structured argumentative exchanges, I aim to enhance the transparency and interpretability of AI systems. This framework draws upon computational and rhetorical argumentation techniques to facilitate more natural human-AI communication and provide clearer explanations of AI decision-making processes. The debate model not only serves as an explainability tool but also offers an innovative method for in-class assessment, allowing educators to evaluate students' understanding of AI concepts through their participation in and analysis of AI-driven debates. This approach addresses the growing need for explainable AI in educational settings while fostering critical thinking skills essential for navigating an AI-driven world.

Using a Distinctive Curricular Design Process for Liberal Arts Computing Programs

Conference Tutorial

Jakob E. Barnard[1], Grant Braught[2], Janet Davis[3], Amanda Holland-Minkley[4], David Reed[5], Karl Schmitt[6], Andrea Tartaro[7], James Teresco[8]

[1]University of Jamestown, Jamestown, ND 58405
Jakob.Barnard@uj.edu

[2]Dickinson College, Carlisle, PA 17013
braught@dickinson.edu

[3]Whitman College, Walla Walla, WA 99362
davisj@whitman.edu

[4]Washington & Jefferson College, Washington, PA 15317
ahollandminkley@washjeff.edu

[5]Creighton University, Omaha, NE 68178
DaveReed@creighton.edu

[6]Trinity Christian College, Palos Heights, IL 60463
Karl.Schmitt@trnty.edu

[7]Furman University, Greenville, SC 29690
andrea.tartaro@furman.edu

[8]Siena College, Loudonville, NY 12211
jteresco@siena.edu

Abstract

As part of its forthcoming article in the Curricular Practices Volume with the new ACM/IEEE-CS/AAAI Computer Science Curricula guidelines (CS2023)[1], the SIGCSE Committee on Computing Education in Liberal Arts Colleges (SIGCSE-LAC Committee) has developed guidance, informed by its sessions at recent CCSC and SIGCSE conferences, to help with the design and/or revision of CS curricula in liberal

[1]https://csed.acm.org

arts contexts [1]. The committee's earlier work found that liberal arts and small colleges approach the design of their computing curricula in unique ways driven by institutional mission or departmental identity. This impacts how faculty at these colleges adopt curricular guidelines. Curricular guidelines like CS2023 inform curriculum design but are balanced with the vision for a program, departmental strengths, locale, student populations, and unique academic experiences. The desire to craft distinctive curricula, combined with the size of curricular recommendations, requires an assessment of trade-offs between achieving full coverage of curricular recommendations and a school's other priorities. SIGCSE-LAC's guidance encourages faculty to reflect on their programs and the role of CS2023, beginning with their institutional and departmental priorities, opportunities, and constraints.

This session will introduce participants to SIGCSE-LAC's guidance to consider curricular development in the context of the unique features of their programs and institutions. Following an overview and brief discussion of CS2023, participants will be guided through an abbreviated design process using the latest version of the committee's reflective assessment process. This process is framed by a series of scaffolding questions that begin from institutional and departmental missions, identities, contexts, priorities, initiatives, opportunities, and constraints. From there, participants will be led to identify design principles for guiding their curricular choices, including the CS2023 recommendations. Examples gathered from the committee's previous CCSC and SIGCSE sessions will be available to help articulate identity and program design principles, which will then be used to identify distinctive program-level learning outcomes. A spreadsheet tool that is being developed to aid in the shaping of curricular choices will be demonstrated. Participants will leave the session with a better understanding of how CS2023 can impact their programs and instruction on how to use the SIGCSE-LACS Workbook outlining our curriculum design process with their departments. Participant feedback will be gathered and used to refine the committee's guidance.

References

[1] Amanda Holland-Minkley, Jakob Barnard, Valerie Barr, Grant Braught, Janet Davis, David Reed, Karl Schmitt, Andrea Tartaro, and James D. Teresco. 2023. *Computer Science Curriculum Guidelines: A New Liberal Arts Perspective.* In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 617–623. https://doi.org/10.1145/3545945.3569793

Presenter Biographies

One or two of this session's eight co-authors will serve as presenter(s)/facilitator(s).

Jakob E. Barnard is an Associate Professor, Chair of the Computing, Design, & Communications Department, and Director of Online Technology Programs at the University of Jamestown. He is a facilitating member of the SIGCSE-LAC Committee, and his research involves how curricula have been integrated into Liberal Arts Computing programs. **Grant Braught** is a Professor of Computer Science at Dickinson College. He is a facilitating member of the SIGCSE-LAC Committee. He has organized committee events focused on curricula and published widely on CS education issues, particularly within the liberal arts. **Janet Davis** is Microsoft Chair and Professor of Computer Science at Whitman College, where she serves as the department's founding chair. She co-organized SIGCSE pre-symposium events in 2020 and 2021 on behalf of the SIGCSE-LAC Committee. **Amanda Holland-Minkley** is a Professor of Computing and Information Studies at Washington & Jefferson College. Her research explores novel applications of problem-based pedagogies to CS education at the course and curricular level. She is a facilitating member of the SIGCSE-LAC Committee. **David Reed** is a Professor of Computer Science and Chair of the Department of Computer Science, Design & Journalism at Creighton University. He has published widely in CS education, including the text *A Balanced Introduction to Computer Science*, and served on the CS2013 Computer Science Curricula Task Force. **Karl Schmitt** is Chair and Associate Professor of Computing and Data Analytics at Trinity Christian College. He has served on the ACM Data Science Task Force and various Computing, Technology, and Mathematics Education committees for the MAA, ASA, and SIAM. His interests explore data science education, and interdisciplinary education between computing, mathematics, data, and other fields. **Andrea Tartaro** is a Professor of Computer Science at Furman University. Her computer science education research focuses on the intersections and reciprocal contributions of computer science and the liberal arts, focusing on broadening participation. She is a member of the SIGCSE-LAC Committee, and has published and presented in venues including the CCSC and the SIGCSE Technical Symposium. **Jim Teresco** is Chair and Professor of Computer Science at Siena College. He has been involved in CCSC Northeastern for over 20 years and currently serves as board chair, and has been involved with the SIGCSE-LAC Committee for 5 years. His research involves map-based algorithm visualization.

Standards-Based Grading in a Wide Variety of Courses*

Lightning Talk

Brian Heinold, Ph.D.
Department of Mathematics & Computer Science
Mount St. Mary's University
heinold@msmary.edu

Last academic year, I switched all my classes over to a standards-based grading scheme. This was for a variety of classes, specifically, CS 1, Theory of Computation, Operating Systems, Computer Security, and Algebraic Structures. This lighting talk will cover the grading scheme I used, what the assessments looked like, and how it all worked out.

Neurodiversity and Computer Science: Working with Neurodiverse Students to Accomplish Their Education Goals*

Nifty Idea

Andrea Wentzell
Computer Science Department
Chestnut Hill College
Philadelphia, PA 19118
wentzella@chc.edu

The transition to college is a significant change for all incoming students, but it can be a bit more complicated for neurodivergent students. Chestnut Hill College's (CHC) Neurodiversity Program recognizes this and works to ensure student success in college through an inclusive environment. Building upon the CHC value and mission of the "Dear Neighbor." Over the last year, the CSIT department has been actively engaged in working with students in the neurodiversity program. This idea will delve into some of the teaching techniques and feedback gathered from students in our program, highlighting our department's commitment to supporting neurodiverse students in their educational journey.

Intended Audience: Faculty members working with neurodiverse students
Material Provided: PowerPoint

Teaching Software Engineering Concepts while Using AI Tools for Programming in Intro Computer Science*

Nifty Idea

Jeffrey Bush
Department of Mathematics abd Computer Science
Moravian University
Bethlehem, PA 18104
bushj@moravian.edu

With the explosion of the usage of AI-assisted tools in programming, both in the professional world and among learners, the approach to teaching students how to use these tools appropriately is of utmost importance. There is little point in denying students access to these tools, as professionals use them and students will seek them out regardless. However, teaching students to use them properly from the very start will allow them to move away from focusing on specific details and nuances of a programming language to be able to focus on larger applications, good design choices, and other aspects of software development besides programming. This means that many software engineering concepts can be introduced earlier in the curriculum, even in a first-semester computer science course, including functional decomposition, data structure decisions, testing, qualitative aspects of programming, stepwise refinement and refactoring, and basic use cases and user stories coupled with software inspections and code reviews. Having students focus on program design, testing, code quality, and building on previous assignments gives students numerous opportunities to develop programming skills. To ensure students are learning a programming language as well, they have to be able to read their own code and other code samples along with being able to research and explain any unfamiliar code generated by an AI tool.

Intended Audience: Introductory CS educators, academic program planners
Material Provided: Example outline and in-class and homework assignments

Persistent Homology Detects Bifurcation in Dynamical Systems*

Poster Session

Chase Demick
Skidmore College
Saratoga Springs, NY
chasedemick@skidmore.edu
Faculty Advisor: Junyi Tu, Salisbury University
jxtu@salisbury.edu

Topological Data Analysis(TDA) is a fast growing research field in data science combining both pure mathematics and computer science. Topology often deals with the overall properties of a space, while the study of dynamical systems deals with the overall behavior of a system without necessarily being preoccupied with numerical details. Key in the study of global behavior is the analysis of bifurcations, which are quantifiable changes to the system's behavior with respect to some set of parameters. We use the tool of persistent homology in TDA to develop a procedure to analyze dynamical systems by comparing the homologies of their trajectories, known as orbits, to those of circles approximating the orbit's long-term behavior. We implement the procedure in open source library in Python, and studies the Hopf bifurcation in FitzHugh-Nagumo (FN) system from neuroscience.

Enhancing Academic Insight: Leveraging Web of Science Data with Large Language Models for Comprehensive Faculty and Article Analysis at Salisbury University*

Poster Session

Spencer Presley and Cole Barbes
Salisbury University
Salisbury, MD
{spresley1,cbarbes1}@gulls.salisbury.edu
Faculty Advisor: Junyi Tu, Salisbury University
jxtu@salisbury.edu

Our project leverages the Web of Science academic database (WoS) alongside Large Language Models (LLMs) to aggregate article metadata and categorize articles. We conducted this process on the last 5 years of Salisbury University (SU) article data, the raw article metadata was exported from WoS. From the raw metadata, we aggregate and extract features on a per-category basis, for each category we construct the following features: Number of faculty, number of articles, number of departments, total citation count across each article, average citations per article, individual article citation count, each faculty members total and average citations, influential faculty, influential articles. Alongside this, we also associate the faculty with the articles they've written or helped write. This project provides a comprehensive analysis of the expertise held by each SU faculty member. The data is made available via an intuitively designed Excel spreadsheet as well as a publicly accessible website. Moreover, this approach is designed to work for any University whose article metadata is available via WoS, not just SU. Future updates will see the addition of additional data features and visualizations. Along with the additional features and visualizations, we will ensure fresh data is always available by empowering users to trigger manual updates as well as allow for automatic updates with user-specified time frames.

Detecting and Analyzing Tennis Ball Movements with AI, Deep Learning, Computer Vision and Large Multimodal Models*

Poster Session

Hunter Liao
Technology and Computer Studies Division
Hagerstown Community College, Hagerstown, MD
hliao@student.hagerstowncc.edu
Faculty Advisor: Dr. Osman Guzide, Shepherd University
oguzide@shepherd.edu

In competitive sports like tennis, analyzing gameplay videos to improve strategies and player performance is crucial. However, tracking the ball's trajectory in tennis poses unique challenges due to the small size of the ball, its high speeds that often exceed 100 miles per hour, and the frequent low quality and pixilation of video footage. These factors complicate the accurate detection and tracking of tennis balls, impacting the effectiveness of traditional video analysis techniques.

The research presented here begins with a comprehensive review of existing methodologies for sports video analysis, mainly focusing on the unique challenges presented by tennis. We then introduce an innovative approach employing Artificial Intelligence, specifically Deep Learning and Large Multimodal Models (LMMs), designed to detect, track, and analyze fast-moving tennis balls with high precision. Our proposed software system aims to surpass current capabilities in sports analytics by providing more accurate identification of tennis ball trajectories and delivering actionable recommendations and feedback directly to players and coaches. This system not only enhances training efficiency but also helps players refine their techniques based on reliable, AI-enhanced visual data.

Building a Voice-Activated RAG Chatbot with Generative AI and LLMs*

Poster Session

Tyler Wills and Cara Burgan
Department of Computer Science, Mathematics and Engineering
College of STEM
Shepherd University
Shepherdstown, WV 25443
{twills02@,cburga02}@rams.shepherd.edu
Faculty Advisors:
Dr. Osman Guzide and Dr. Weidong Liao
{oguzide,wliao}@shepherd.edu

We have developed a voice-controlled Retrieval-Augmented Generation (RAG) chatbot application at Shepherd University that utilizes Generative AI and Large Language Models (LLMs) to transform how students interact with their student handbook. This application combines cutting-edge natural language processing and voice recognition technologies to create a more dynamic and user-friendly experience. The chatbot leverages LLMs to understand and respond to diverse queries, delivering detailed, contextually appropriate responses directly sourced from the handbook. The RAG methodology ensures responses are not only generated by the AI but also precisely retrieved from relevant handbook sections, preserving response integrity and accuracy.

Designed to enhance user experience substantially, this chatbot offers hands-free access to essential information, facilitating increased engagement and ease of use. Students can voice their questions and receive instant, accurate, and concise answers, eliminating the need to manually navigate the handbook. This advancement streamlines information access and supports an inclusive educational environment by catering to various learning preferences and needs. The implementation of this AI-powered chatbot marks a significant progression in integrating modern educational technologies, contributing to a more connected and well-informed student community.

Unveiling the Nexus: AI, Environmental Impact, and Cost*

Poster Session

Vageesh Hegde, Supreetha Bolar
Department of Computer Science & Information Technologies
Frostburg State University
Frostburg, MD
vhegde0@frostburg.edu,supreethabolar61@gmail.com

As we move into 2024, the global landscape is experiencing a significant increase in the adoption of artificial intelligence (AI), which is revolutionizing industries and societies. AI, powered by machine learning (ML) and other advanced programming techniques, represents non-human intelligence capable of learning from large datasets. This transformative technology presents unparalleled opportunities, but also significant challenges, particularly with regards to its environmental impact and economic feasibility. This paper explores the two sides of AI development: Generative AI requires a lot of processing power, leading to high energy consumption and substantial CO2 emissions, affecting its cost-effectiveness and environmental impact. It meticulously examines the complex components contributing to the operational costs of AI models, including computational resources, data storage, energy consumption, and infrastructure requirements. It rigorously analyzes factors influencing these costs, such as model complexity, data volume, and technological infrastructure, to provide a comprehensive framework for cost analysis in AI. Furthermore, the paper explores methodologies for evaluating and quantifying these operational costs, which are essential for calculating return on investment (ROI) in AI initiatives. Real-world case studies illustrate the practical applications of AI, comparing different models to determine their cost-effectiveness and ROI. The paper discusses emerging trends in AI development focused on reducing environmental impact, including green AI initiatives and energy-efficient strategies. It concludes with insights into future research directions, advocating for advancements in cost analysis methodologies and sustainable AI practices to promote responsible AI innovation. In summary, his paper offers a comprehensive analysis of the economic and environmental aspects of AI deployment, providing valuable insights for stakeholders navigating the complexities of AI implementation in a rapidly evolving technological landscape.

An Analysis of the Default Security Postures of Common Linux Distributions*

Poster Session

John M. Roy
Marymount University
Arlington, VA
john.roy.2197@gmail.com

Linux is best-known for being a free and open-source operating system that also has the incredible versatility to be installed on multiple platforms using minimal system requirements.

In the cybersecurity realm, Linux is also often considered to have superior hardening capabilities when compared to the much more dominant Windows and Mac OS operating system families. With Linux becoming an increasingly attractive option around the world to replace these more mainstream operating systems, understanding the vulnerabilities associated with the OS and how to mitigate them is imperative. Unlike Windows and Mac OS however, the open-source nature of Linux has led to the development of multiple distinct distributions with their own purposes and configurations. This can yield wildly different default security postures among similar generation Linux releases. While different Linux distributions have a common core, each variant is developed independently by either a community collective or commercial entity with a certain purpose in mind.

The main purpose of this article is to test the theory of security-by-design by analyzing and comparing the vulnerabilities present in the security configurations across several commonly used distributions of Linux.

Using a collection of vulnerability assessment tools, these Linux distributions are examined on a collection of different hardware platforms at a freshly installed state as well as a fully updated state. While this process does not give a full picture as to how secure each distribution can be when fully configured for operational use, it does help to gain a deeper understanding of the

prioritization of security in their development and deployment.

Keywords: Linux, Security-by-design, Vulnerabilities

Be There or Be Square: Finding the Perfect Shape With a Genetic Algorithm*

Poster Session

Shawn Patrick Harriet and Avery Lamprecht
Mount St. Mary's University
Emmitsburg, MD
{s.p.harriett,j.lamprecht}@email.msmary.edu

This project explores the use of a genetic algorithm to learn to generate a square. The program starts with a list of shapes with four randomly-generated vertices. It then uses a method of scoring each generated shape to determine closeness to a square, crossing over and randomly mutating into new generations until a square is found (or a user-set generation cap is reached). In this program, it was discovered that a larger number of members per generation was more valuable than a higher number of generations, though both were important. This application of a genetic algorithm was created while learning about types of problem-solving algorithms, and displays one in a way that can be easily demonstrated and understood.

Automating Server Deployments with Ansible: Utilizing Automation in DevOps*

Poster Session

Blake Douglas Hatcher
Mount St. Mary's University
Emmitsburg, MD
b.d.hatcher@email.msmary.edu

Automation is increasingly becoming a focal point for analysts and businesses that utilize IT infrastructure. The International Data Corporation predicts that by 2027, 90% of organizations will incorporate automation or AI into operational roles, leading to a projected 30% increase in worker efficiency [1]. Red Hat Ansible is one such automation tool, and in this case, it was used to expedite server deployment. According to a study conducted by the International Data Corporation, the Red Hat Ansible Automation Platform achieved a 67% reduction in the time required to manage network resource configurations [2]. This poster outlines the integration of Red Hat Ansible for Elasticsearch and analytics engine clusters. It highlights the benefits, challenges, and applications for both test and production servers within the Security Operations Center and disaster recovery capabilities for the Mount St. Mary's University network. The Mount Security Operations Center has utilized Red Hat Ansible to reduce the deployment time of a new three-server Elasticsearch stack, complete with appropriate plugins capable of interfacing with existing infrastructure, from an average of three days to approximately two hours. In addition to reducing deployment time, the Security Operations Center now benefits from consistent and shareable standards, ensuring that all changes to server deployment are always reflected in Red Hat Ansible Playbooks. This rapid deployment capability also serves as a disaster recovery system, enabling the Security Operations Center to quickly redeploy servers in the event of an emergency, ensuring minimal disruption and maintaining operational continuity. This poster demonstrates how Red Hat Ansible has significantly decreased overall server deployment time, providing robust support for the Security Operations Center's critical infrastructure.

References

[1] Jensen, J. (2024, March). The Business Value of Red Hat Ansible Automation Platform. Signapore; IDC. https://www.redhat.com/rhdc/managed-files/cl-idc-business-value-ansible-automation-analyst-material-1049194-202403-en.pdf

[2] Shanbhag, P. (2024, May). Automation: The Key to Enterprise Operations Transformation. Singapore; IDC. https://www.redhat.com/rhdc/managed-files/ma-idc-enterprise-automation-transformation-analyst-material-1250246-202406-en.pdf

Modern State Marijuana Legislation and its Effect on Crime*

Poster Session

Joseph Paul Jacobs III
Mount St. Mary's University
Emmitsburg, MD
j.p.jacobs@email.msmary.edu
Faculty Advisor: Dr. Jonathan McCurdy

A prominent political struggle in the 21st century has been centered around state marijuana decriminalization and its potential impact on crime rate. The purpose of this project was to use statistical analysis to determine if decriminalization between 2015 and 2019 impacted property and/or violent crime rates. States were partitioned into decriminalized and non- decriminalized groups and paired t-tests were performed to determine if a significant statistical difference in violent or property crime rates presented themselves. Analyzing the data across 5 years and comparing the groups, there was evidence to suggest a statistically significant difference in crime rates between the two groups, where crime rates in decriminalized states tended to be lower than those in states that had not been decriminalized. Additional testing was performed to determine if states that decriminalized marijuana in 2016 saw a change in crime rates by looking at the 2015 and 2017 years. Based on this testing, there was a difference in property crime rates (lower in decriminalized states) but no difference in violent crime rates for states which decriminalized marijuana in 2016. Further analysis showed that the national average for property crime decreased after 2016, lining up with what we saw for states that decriminalized marijuana in 2016. Overall, the extensive testing done in this project seems to indicate that marijuana decriminalization can be correlated to a decrease or static movement in violent and property crime rates nationally.

Creating a Jguardrail Plugin for IntelliJ IDEA*

Poster Session

Mackenzie Swain
Department of Computer Science
University of Mary Washington, Fredericksburg, VA
mswain2@mail.umw.edu
Faculty Advisor: Dr. Ian Finlayson
ifinlay@umw.edu

The jguardrail program was created "...to provide more warnings for potentially incorrect code than the javac compiler does, even with warnings enabled."[1] The plugin for IntelliJ IDEA was created to allow more access to this program across different code editors. IntelliJ IDEA is a popular IDE among beginning computer science students and debugging is a particularly difficult skill for beginning programmers. Having this plugin to highlight errors is preferable to the program behaving differently than the programmer expects and allows students to recognize those errors and find the proper solutions.

The goal of this project was to create a plugin for IntelliJ IDEA that implements jguardrail, a program that provides more warnings for potentially incorrect Java code. Jguardrail identifies common mistakes that beginning computer science students make that are not identified by the traditional Java compiler or IntelliJ IDEA. Jguardrail analyzes code to identify these errors, but it cannot be used directly with an Integrated Development Environment (IDE), a code editor where programs are written, compiled, and run. A plugin is a software addition that can be installed on an app to add new features, providing a way to allow jguardrail to run on the IntelliJ IDE. The plugin aims to run jguardrail to identify errors, visually highlight them, and prevent projects from compiling when such errors are present. Research and testing were conducted to develop a plugin with this specific implementation that also integrated with the IntelliJ IDEA visual interface. The resulting plugin successfully runs jguardrail to identify ten different errors, highlight them in the editor, and prevent the project from compiling when these errors are detected.

This plugin identifies and highlights common beginner errors that IntelliJ alone does not catch allowing students to recognize the errors and find the appropriate solutions.

Works Cited

[1] JGuardrail GitHub Page, `https://github.com/IanFinlayson/jguardrail`.

Exploring the Architecture and Application of Transformer Models in Natural Language Processing and Media Generation*

Poster Session

Zhairui Shen, Tianwei Wang, and Vitaly Ford
Department of Computer Science
Arcadia University
Glenside, PA
{zshen_01,twang_05,fordv}@arcadia.edu

This project investigates the detailed architecture of Transformer models, including components such as self-attention mechanisms, multi-head attention, and positional encoding. The research further explores the application of Transformers in text processing, covering tasks like tokenization, stemming, and word embedding. Additionally, the project examines the use of Transformer models in generating images and videos from textual descriptions, specifically through the integration of Generative Adversarial Networks (GANs) with Transformers. The expected outcome of this research is a comprehensive analysis that illustrates the significant impact of Transformer models (and their enhancements) on both natural language processing and AI-driven content creation.

By thoroughly optimizing the Transformer architecture—focusing on attention mechanisms, positional encoding, and multi-head attention—we plan to achieve more precise and contextually accurate visual outputs. Our work also concentrates on accelerating the training process of Transformer models without sacrificing accuracy via:

1. Mixed Precision Training: speeding up model training while maintaining numerical stability.

2. Distributed Training: model training in multi-GPU or multi-node environments.

3. Learning Rate Adjustment: exploring dynamic learning rate adjustment strategies and comparing optimization algorithms (e.g., AdamW, LAMB) to enhance training efficiency.

Additionally, this research will explore various model improvement strategies:

4. Model Architecture Optimization: applying pruning, quantization, and knowledge distillation to enhance the model structure.

5. Attention Mechanism Refinement: investigating techniques like sparse attention and low-rank approximations to optimize the attention mechanism.

6. Pre-training and Fine-tuning Strategies: exploring advanced pre-training strategies and fine-tuning methods to improve downstream task performance.

We plan to continue contributing to AI-driven content generation and laying a foundation for further advancements, where improved Transformer models can be applied to a wider range of tasks, leading to more sophisticated and accurate AI-generated media. Through these efforts, we aim to push the boundaries of what is possible in text-to-image and text-to-video generation.

A Shortcut to Parallel Programming in Java*

Poster Session

Connor Hairfield
Department of Computer Science
University of Mary Washington, Fredericksburg, VA
chairfie@mail.umw.edu
Faculty Advisor: Dr. Ian Finlayson
ifinlay@umw.edu

While modern computer processing units rely on multiple cores to gain efficiency, the parallel programming required to utilize these efficiencies is less widely used than it should be. With increases in single-core performance diminishing, manufacturers are placing more cores in their chips in response. However, writing programs that utilize multi-core performance is less common largely because of the heightened awareness required from developers and the increased difficulty for beginners to learn. In Java, one of the most popular programming languages, writing parallel programs requires the developer to add a considerable amount of code to an existing program. Aside from the integral sections that give a program a distinct function, much of this code is similar between different parallel Java programs. Eliminating the need to type out the shared structure of parallel programs in Java is one step to encourage increased use of parallel programming. To achieve this, a new front end to Java is introduced that adds syntax to specify concise parallel blocks. Using parsing, the Java code containing the new parallel syntax is transformed and expanded into traditional Java code which is then compiled and run. This removes the requirement to write fixed parallel Java code, relieving much of the burden for the developer and making parallel programming easier to learn for beginners. This shortcut to writing parallel programs can encourage greater use of parallel programming in Java.

Trends and Privacy Challenges in Mental Health Apps: A Data-Driven Analysis*

Poster Session

Xingyu You, Wang Wang, Yanxia Jia
Department of Computer Science
Arcadia University
Glenside, PA
{xyou,wwang_04,jiay}@arcadia.edu

According to the National Alliance on Mental Illness (NAMI), an estimated 48.3 million adults in the U.S. deal with some type of mental health condition each year. In recent years, the number of mental health apps available to consumers has grown significantly. These apps can help address various mental health issues, such as stress, depression, and anxiety, particularly for individuals with milder symptoms. However, several challenges remain, including low user engagement, inconsistent quality, and concerns over data privacy and security.

In this study, we utilize data science techniques to analyze a dataset of mental health-related mobile apps. App data was collected through the Apple Store API, and privacy document content was extracted through python tools. We performed exploratory analysis and clustering analysis to derive insights from general app data, and ChatGPT was employed for an initial analysis of privacy policy documents.

Our findings reveal a consistent rise in the number of mental health apps since 2009, with a significant increase beginning around 2020. This suggests heightened awareness of mental health-related issues in the post-pandemic period. Among the different categories of mental health apps, Health and Fitness, Education, Lifestyle, Medical, and Games are the most prevalent. A comparison of apps released before and after the COVID-19 pandemic shows a marked increase in user ratings in the post-pandemic period. Cluster analysis identifies four distinct clusters, each characterized by differences in app ratings, pricing, release years, update frequencies, and user engagement levels. Lastly, a preliminary examination of app privacy policies was conducted to assess compliance with privacy policy requirements and awareness of privacy laws and regulations, such as GDPR and HIPAA.

SightAssist: Mobile App for Object Recognition and Navigation for the Visually Impaired*

Poster Session

Tarron Montgomery, Brandon Wiggins, Malachi Gray,
Staphord Bengesi, Md Kamruzzaman Sarker
Bowie State University
Bowie, MD
{montgomeryt0919,wigginsb006,graym0830}@students.bowiestate.edu,
{sbengesi,msarker}@bowiestate.edu

Visually impaired individuals face significant challenges navigating due to a lack of visual cues, relying on auditory and tactile input, which can make even simple tasks difficult. Hazards like clutter or changes in layout can further complicate navigation. While traditional aids such as canes and guide dogs provide some assistance, they have limitations. Advancements in assistive mobile technology, such as navigation and object recognition apps, are helping to increase the independence of visually impaired individuals. These apps offer real-time data and audio feedback, reducing reliance on others and boosting confidence in various environments. To address these needs, we developed SightAssist, a user-friendly mobile application designed to recognize objects in the user's surroundings and assist with navigation. SightAssist uses advanced technologies like the YOLO (You Only Look Once) object recognition framework and a real-time camera feed via a Raspberry Pi to deliver accurate, timely readings of the environment. The app also includes text-to-speech functionality to convert visual data into audio cues, enhancing usability and enabling easy interaction. The live feed from the external camera is broadcast to the user's smartphone through the app, ensuring seamless navigation. The SightAssist prototype has shown promising results in experimental testing. By combining text-to-speech and real-time object recognition, the app significantly improves the user experience, offering accurate and reliable navigational assistance for the visually impaired.

Evaluating Stigma: Experiences Modifying the Assembly Code of Closed-Source Android Apps*

Poster Session

Shokhinabonu Tojieva, Chelsea Rao, Ed Novak
Franklin & Marshall College
Lancaster, PA
shokhinatojieva@gmail.com, {crao,enovak}@fandm.edu

A difficult but potentially powerful advanced software engineering concept is to modify existing, compiled, closed-source applications to identify and potentially remedy security and privacy issues. This technically challenging concept is very applicable to the Android ecosystem, as users have little insight on the possible vulnerabilities in these app designs. However, existing approaches are bespoke, use-case specific implementations. In this poster we present the evaluation of Stigma, an open-source software tool which can make modifications to commodity Android applications. Our tool allows researchers and skilled users to define their own desired modifications for a range of purposes such as security and privacy analysis, improving app functionality, removing unwanted features, debugging, proling, and others. We evaluate Stigma in terms of compatibility, efficacy, and efficiency on approximately 100 commodity Android applications. This involved measuring the LOC, memory, and CPU overhead. Stigma could potentially be used in the future for various tasks, including detecting app usage of sensitive information, identifying security vulnerabilities, fixing bugs, and even enhancing app functionality without requiring the original developer's involvement

Cost-Effective Detection of Invasive Ductal Carcinoma Using CNNs and Explainable AI on Histology Images*

Poster Session

Dylan Jayabahu
Laurel Heights Secondary School
Waterloo, Ontario, Canada
dylanjayabahu@gmail.com

Invasive Ductal Carcinoma (IDC) is the most prevalent form of breast cancer, making its early and accurate detection crucial for patient prognosis and treatment. Traditional deep learning approaches for IDC detection often rely on expensive imaging techniques such as CT scans and X-rays, which can be inaccessible in resource-limited settings. This study addresses this challenge by exploring a method that utilizes affordable optical microscope images to achieve competitive accuracy.

Our study aimed to develop and evaluate a cost-effective deep learning model for detecting IDC in breast histology images. We sought not only to surpass existing accuracy benchmarks but also to enhance the interpretability of the model through the integration of explainable AI (XAI) techniques.

We used a dataset of 277,524 histology image patches (198,738 IDC-negative and 78,786 IDC-positive) obtained from Andrew Janowczyk's research. Several convolutional neural network (CNN) models were trained using this dataset, with data augmentation employed to boost model generalizability. Additionally, XAI techniques such as saliency maps were implemented to visualize IDC-affected areas both at the individual patch level and across full histology slides.

Our approach achieved a minimum testing accuracy of 86.2%, outperforming existing methods, which generally achieve around 84%. The model was able to generate detailed heatmaps that highlighted IDC-affected regions at a granular level using only histology images, eliminating the need for more costly imaging technologies.

This study demonstrates that CNN models, combined with data augmentation and XAI, can achieve state-of-the-art accuracy for IDC detection using cost-effective optical microscope images. The enhanced transparency provided by XAI methods offers valuable insights into model predictions, making this approach a viable and accessible alternative for breast cancer diagnosis, particularly in settings where advanced imaging technologies are not available.

A Sample Lecture at College-Level for Optimizing Linear Search Algorithm*

Poster Abstract

Penn Wu
Cypress College
Cypress, CA 90630
pwu@cypresscollege.edu

When searching for a value in an unsorted list, the fastest approach is to sequentially iterate through the list to find the target. Although there are techniques that can potentially improve the searching efficiency on unsorted data, optimizing a linear search is not commonly discussed in a college-level course of data structure and algorithm.

The presentation will describe how to integrate methodologies, such as parallel search with sharding, to possibly optimize linear searches. In the classroom, students often experience different execution results. They are: (1) the optimized versions run faster than the simple linear search, (2) an $O(n^2)$ algorithm run quicker than an $O(n)$ algorithm on the same input, (3) the simple linear search runs faster than the optimized versions. These deviations are results of selected algorithms with a few relevant factors: (1) hardware related issues, (2) properties of dataset, (3) implementation of algorithm, (4) testing condition, and (5) code complexity.

The provide graphs and visual aids will show how the optimization efforts affect linear searches. Simple sharding, odd-even sharding, and remainder sharding, with the arranged mechanisms of parallelization, show the potential to shorten the searching time. The in-class testing of programs also confirms that the speed of execution is closely tied to the efficiency of the algorithm being used. Well-designed algorithms can minimize execution time by optimizing resource usage such as CPU cycles, memory, and I/O operations. The quality of the code, compiler optimizations, and parallelization techniques affect execution time. After the lecture, students generally show a better understanding

that the quality of the code, compiler optimizations, and parallelization techniques affect execution time.

The pedagogy could engage students in critical thinking about code optimization, fostering efficient problem-solving strategies, and preparing them for real-world scenarios. Students can also seek opportunities to improve scalability, such as handling larger datasets or more complex problems.

Leveraging ChatGPT for SQL Learning: An Interactive Approach*

Poster Abstract

Ching-yu Huang and Paolien Wang
Department of Compute Science and Technology
Kean University, Union, NJ 07083
{chuang, pawang}@kean.edu

In today's data-driven world, SQL proficiency is essential for data analysts, software developers, and database fields. However, traditional SQL learning methods often rely heavily on classroom lectures and textbooks, resulting in repetitive activities where students tackle the same exercises and questions. This approach can be inefficient, hindering student learning as instructors must individually address syntax errors and runtime issues, limiting their ability to assist all students effectively. Additionally, student engagement and comprehension may suffer, particularly for those with weak foundations who heavily depend on instructor guidance.

To address these challenges, we propose an innovative approach to SQL learning utilizing ChatGPT, a state-of-the-art natural language processing model developed by OpenAI. ChatGPT provides an interactive platform where users can engage in natural language conversations to ask questions, seek explanations, and practice SQL queries in a simulated real-world environment. It fosters a conversational learning experience, allowing users to ask questions and receive immediate responses, practice SQL queries, and get instant feedback to refine their skills. Additionally, ChatGPT adapts to the user's pace, offering tailored explanations and examples.

The instructor can provide real-world questions, allowing students to utilize ChatGPT to find answers and comprehend explanations. For instance, students can ask ChatGPT to generate SQL queries to create an employee table with primary and possible foreign keys, including several records, and another query to display the name and salary of the highest-paid individual.

Additional SQL questions related to this employee table can also be given within the same session. By leveraging conversational AI with ChatGPT for SQL learning, students won't rely solely on the instructor, empowering them to study new topics independently after the class. This self-learning skill is crucial for their transition into the workforce.

Approaches to Identifying Preteen Suicide Risk and Protective Factors*

Poster Abstract

Osei Hyiamang
Marymount University
Arlington, VA 22201
ohyiaman@marymount.edu

Preteen suicide has become a concerning issue in recent years, necessitating a concerted effort to identify the factors that contribute to this menace while also exploring potential protective factors. Drawing from a comprehensive literature review and analysis of existing studies, this paper identifies key factors that contribute to preteen suicide risks, such as mental health disorders, adverse childhood experiences, social isolation, and personality traits. This research provides a comprehensive understanding of approaches to identify preteen suicide risks and protective factors. The research identifies protective factors that could be developed to change the risk trajectory, both proximal and distal. The study also identifies intervention targets and timing, as well as scalable assessment approaches that would have clinical utility for characterizing both risk or protective factors and suicide risk status and outcomes. The study finally uses a comprehensive method to identify suicide risks and protective factors associated with preteen. Assessment instruments like self-report questionnaires can be employed in educational institutions, healthcare facilities, and community groups to recognize individuals who are potentially vulnerable. The study identifies intervention targets and timing, as well as scalable assessment approaches that would have clinical utility for characterizing both risk or protective factors and suicide risk status and outcomes. The first section of this paper discusses the current prevalence of preteen suicide and highlights the importance of early intervention and prevention strategies. It then delves into the various methods that can be employed in identifying risk factors, ranging from psychological assessments, behavioral observations, and social analyses to genetic and neurobiological explorations. The strengths and limitations of each approach are

critically evaluated to determine their effectiveness in accurately identifying preteen suicide risk. The subsequent section focuses on the investigation of protective factors that mitigate suicide risk among preteens. A wide range of protective factors, including strong social support systems, adaptive coping skills, and resiliency are examined to understand their role in reducing the risk of suicide in this vulnerable population. The study seeks to underscore the importance of interdisciplinary collaborations among psychologists, psychiatrists, educators, and community stakeholders to develop comprehensive prevention programs that address both risk and protective factors effectively. The study uses three models and mixed method approach, and the findings to contribute to the existing knowledge offering valuable insights for policymakers, mental health professionals, and researchers. Ultimately, the outcomes of this study inform the development of evidence-based strategies that can prevent preteen suicide and promote mental well-being among this vulnerable population.

Keywords: preteen suicide, protective factors, suicide risk status, outcomes, risk trajectory, proximal and distal, scalable assessment

Enhancing Undergraduate Computing Education with LMMs and ChatGPT-4o*

Poster Abstract

Weidong Liao, Osman Guzide
Department of Computer Science, Mathematics and Engineering
College of STEM
Shepherd University Shepherdstown, WV 25443
{wliao,oguzide}@shepherd.edu

Large Language Models (LLMs) and ChatGPT have significantly impacted programming practices and computer science education. The rapid advancements in natural language processing, recurrent neural networks, and Transformer architectures have captured the attention of students and educators alike. These tools aid students in brainstorming, coding, analyzing code, and writing reports. Although concerns about cheating and plagiarism persist, these tools also provide educators with novel ways to create and assess assignments. Despite some hesitancy among educators to integrate these AI tools into the classroom, the advert and development of Large MultiModal Models (LMMs), the enhancement of LLMs that can deal with multimedia inputs and outputs, illustrates a significant evolution in generative AI capabilities.

This poster explores our use of LMMs and ChatGPT-4o in educating undergraduate students across various computing disciplines, including computer science, engineering, and technology. LMMs' ability to handle multimedia information makes them applicable in diverse fields like web design and robotics. ChatGPT-4o, enriched with new features from LMMs, enhances both teaching and learning. Our approaches include integrating LMM-related concepts into classrooms and utilizing ChatGPT-4o to improve educational outcomes.

Evaluating the Pedagogical Impact of Large Language Models on Programming Skills in Data Science Programs in Higher Education*

Poster Abstract

Giancarlo Crocetti, Seonwoo Bak,
Daena D. Vautor-Laplaceliere, Naqib A. Noory
St. John's University, Jamaica, NY
crocettg@stjohns.edu,
{seonwoo.bak21,daena.vautorlaplaceliere20,naqib.noory22}@my.stjohns.edu

The integration of GenAI (GenAI), such as large language models (LLMs), in education has raised the question of how it will alter the students' training and learning outcomes. To better understand the phenomenon, this empirical study explores whether college students find GenAI tools helpful in advancing their skills, particularly Python programming proficiency.

Throughout a graduate-level course in data science, students were taught the concepts of intelligent web systems, formally trained to use large language models like ChatGPT and Gemini, and encouraged to use them during the execution of a project assignment. The pretest-posttest methodology was employed to investigate the changes in students' self-assessment of programming ability for a data science application using a paired measurement in the form of a survey taken before and after the execution of the project. Factors such as students' prior knowledge, familiarity with other programming languages, and experience with online Python classes were considered to ensure the reliability of the findings.

The results support the positive pedagogical impact of LLMs in improving students' ability to write quality code, analyze data with Python libraries, and build complex applications like web scraping. However, there were no significant changes in students' understanding of Python's basic syntax and constructs or their comfort level in using Python. GenAI tools specifically enhanced more advanced and practical aspects of programming rather than basic

knowledge, suggesting that fundamental programming knowledge is rooted in the student's understanding, which LLMs cannot implant.

Innovative Career-Focused Curriculum for Computer Science and Information Technology*

Poster Abstract

Ching-yu Huang
Department of Compute Science and Technology
Kean University, Union, NJ 07083
chuang@kean.edu

Undergraduate students from disadvantaged backgrounds often lack guidance when it comes to selecting college majors and exploring career paths. As a result, they may be unprepared for internships or job searches during their freshman or sophomore years. Many choose majors like computer science or information technology because of the promising job market, but they may not fully understand the types of jobs, roles, and skills required in these fields.

While attending ACM or IEEE club activities and seminars can help students gain career insights and develop skills, those from disadvantaged families often need to work 20 or more hours per week. As a result, they leave campus after classes and miss out on valuable extracurricular opportunities. Therefore, it is crucial to offer a free elective course in Career Education for CS/IT students, helping them explore the field, understand various job functions, identify the necessary skills, and prepare for job interviews early on. This course would enable them to earn credits toward their degree while learning how to craft strong resumes, prepare for job searches, and succeed in interviews.

This paper presents a comprehensive curriculum for career education in computer science, covering topics such as academic and industry roles, the alignment between coursework and job functions, an overview of research opportunities in CS/IT, a four-year study plan, building a professional resume and e-portfolio using course projects, mock interview practice, job search strategies, work ethics, NSF REU programs, graduate study options (MS and Ph.D.), and job offer negotiations. Students will participate in peer review of resumes, conduct research, and receive constructive feedback from the instructor. Additionally, they will learn how to effectively use online resources such as ChatGPT, YouTube, Glassdoor, Indeed, Monster, and Levels.fyi to stay updated on job requirements, roles, and salary ranges.

Introducing First-year Students to Computing through Class OSSD*

Poster Abstract

Lily Liang
Department of Computer Science & Information Technologies
University of the District of Columbia
lliang@udc.edu

We developed a Minecraft Open Source Software Development (OSSD) project for the Computer Science I class, leveraging the experiential learning of OSSD to engage first-year computing students. Participants built upon a Minecraft game for teaching K-12 students programming, which an upper-class student team had previously developed. We then investigated the project's impact on the student participants' sense of community and motivation to pursue their disciplines. The students reported positive experiences during the end-of-semester interviews.

We chose this Minecraft game because of the following:

1. Many first-year students aspire to become game developers;

2. No special knowledge is needed except knowledge of Minecraft, which many first-year students already have due to the popularity of Minecraft games;

3. Practicing block-based programming and developing course software reinforces what is learned in the first-year programming course;

4. Block-based programming of Minecraft lowers the bar for OSSD participation, allowing legitimate peripheral participation (LPP) in a programming community;

5. The Minecraft education platform is freely available to programmers and gamers;

6. There's a large population of potential users of the software developed.

Participation in the project was voluntary and was encouraged with extra credit. We created learning activities that simulate programmers' community of practice and developed not only students' programming skills but also their team communication, project management, and leadership.

The project provided situated learning in a traditional decontextualized curriculum and prepared students to participate in professional organizations, computing communities, and internships as they advance in their disciplines. We plan to expand this OSSD group to a larger community in the department, which works with the Computer Science I class each semester on this and additional OSSD projects.

Understanding Immediate Weekend Hits and Cult Classics using Natural Language Processing Script Analysis*

Poster Abstract

Heather Cassel[1], AnNM. Phaml[2],
HungT. Nguyenl[3], Linh B. Ngo[1]

[1]Computer Science Department
West Chester University of Pennsylvania
West Chester, PA
{hc946859, lngo}@wcupa.edu
[2]Highschool of Education and Sciences
Ha Noi, Vietnam
Ngocminhanp@gmail.com
[3]Ly Thai To Highschool
Bac Ninh, Vietnam
kin.sit2k6@gmail.com

There are different ways of measuring a movie's successes, including box office performance, critical acclaim, cultural impact, online reviews, and franchise potential. These measurements of successes could be completely in contradiction with one another. For example, a movie could fail financially during the opening weekend but eventually become a classic (e.g. Blade Runner in 1982). On the other hand, a movie could be harshly criticized but ended up making millions (e.g. the Sharknado series). Even online reviews, including both critics and users, have demonstrated to have no relationships with movie success. In other words, being liked is different from being talked about and doing well in the box office. It is therefore not appropriate to use one measure of success to directly predict another. Instead, a different measure of success would be understanding the actual creative content of the movies, expressed via script analysis. A similar process was proposed in the movie production green-lighting process, where prospective scripts are evaluated using natural language processing. Subsequent literature has focused mainly on the script generation process itself rather than measuring and predicting movies' success. In this work, we will investigate this gap by utilizing natural language processing to study movie scripts to identify possible relationships between language attributes and other measures of success. Our work will start with acquiring all relevant data, including manual data downloading activities (box office results), automated mining (using libraries to mine movie scripts), and large-scale data processing (mining online user reviews from Reddit's movie subreddits). Next, data mining techniques are applied to consolidate these various data sources using strings of predefined movie titles as the keys. Statistical and textual analysis using natural language processing will be applied to the acquired data to identify related patterns between movies' metrics of success.

Inclusive by Design: How Accessibility Regulations are Shaping the Future of Software Development*

Poster Abstract

Peter A. Hartnett
Department of Computer Science & Information Technologies
Frostburg State University, Frostburg MD, 20882
phartnett0@frostburg.edu
Faculty Advisor: Professor Michael B. Flinn
mflinn@frostburg.edu

This presentation aims to illustrate the importance of the Americans with Disabilities Act (ADA) as it intersects with software development. It explores the legal, historical, and technical implications of compliance with ADA regulations in web and application development. Approximately 30% of the U.S. population has a disability. The CDC estimates 6 million Americans have vision loss and 15.5% of adult Americans have hearing loss. The risk of developing these conditions increases with age. However, many existing websites and applications lack basic accessibility tools. As the population of computer users ages, the need for accessibility tools in technology – and education on how to implement them effectively – will continue to expand.

The 2024 Final Rule for Title II of the ADA requires all state and local government websites and applications to comply with Web Content Accessibility Guidelines 2.1 level AA. Legal scholars believe that Title III of the ADA will soon extend these requirements to commercial websites and applications, resulting in a massive demand for accessibility solutions. This research highlights the implications of expanding legal requirements for ADA compliance, as well as exploring the universal benefits of accessible design. The findings contribute to the growing discourse on the importance of integrating accessibility features from the outset of software development to foster inclusive digital experiences.

Big Interview, Big Deal: Connecting Academic Experiences to Careers with Interview Practice*

Poster Abstract

Karen Anewalt and Jennifer Polack
Department of Computer Science
University of Mary Washington
Fredericksburg, VA 22401
{anewalt,polack}@umw.edu

Colleges are increasingly facing pressure to assist students in transitioning to careers after graduation. We explored the use of assignments requiring students to practice interview skills while reflecting on experiences in completing coding assignments in undergraduate courses. Using an online mock interview tool, students in a CS1 course and a senior-level software engineering course were required to film video responses to interview-style questions related to course programming assignments. Data was collected from 24 CS1 students and 29 software engineering students. The survey data provides insights into how mock interview practice can bridge the gap between academic experiences and career readiness, enhance students' interview skills, and increase their confidence in discussing their work.

The data collected in the initial semester of this assignment shows that most students reported a satisfactory experience and found the mock interview tool to be user-friendly. Approximately 45% of students said that completing the assignment improved their confidence in discussing a coding project in a professional manner. Approximately 47% said the assignments helped them to improve their interview skills. And 53% said that the mock interview experience made them feel more prepared for future interviews. Students shared several suggestions for future consideration including having an option to answer questions and receive feedback in person, which they felt would also be beneficial.

Malware Detection using Deep Learning*

Student Research Paper

Roxan Rockefeller

Department of Computer Science
Bowie State University
Bowie, MD 22030
rockefellerr0515@students.bowiestate.edu

Malware requires complex detection techniques because it poses serious hazards to both individual systems and networks. In order to demonstrate how well deep learning can adapt to new threats by learning from a variety of datasets, this research investigates the effectiveness of deep learning for static malware detection. When machine learning and deep learning methodologies are compared, it can be seen that deep learning produces malware classification results with more accuracy.

The paper trains a neural network model improved with k-nearest neighbors (k-NN)for preprocessing and ensemble techniques using the SoReL-20M dataset, which consists of around 20 million samples. This method makes use of k-NN's advantages in local trend analysis and data augmentation to enhance the model's performance. The findings highlight deep learning's benefits for cybersecurity, especially its resilience and versatility in detecting and categorizing malware.

Implementing a FashionBot Curriculum in High School Classroom To Improve Student Engagement and Motivation in Computing*

Student Research Paper

Onyinye Ukeneru-Steve and Lily Liang, Ph.D.†

Department of Computer Science and Information Technology
University of the District of Columbia
Washington DC 20016
onyinye.ukenerusteve@udc.edu

Computing often involves abstract concepts that can be difficult for students to grasp, leading to disengagement if not adequately contextualized or related to real-world applications. We implemented a Socio-Cultural STEM curriculum that uses FashionBots to engage and motivate underrepresented minority high school students in computing. FashionBots is a low-cost and easy-to-program robot that integrates fashion into robotics. The curriculum is part of a manual script that will be published in a book chapter on socio-cultural STEM education written by the second author.

We implemented the FashionBot curriculum successfully in the first author's classroom. The students researched the history, tradition, and culture of FashionBots' clothing and learned programming. The highly interactive and hands-on instructions allowed the students to build, program, and experiment with the FashionBots. The students were taught the parts of the robot, how they work together to create the movements, and the Fashionbots' source code. Then, students were asked to identify a change they could make to the code so that the Fashionbots moved differently. Students felt welcomed, celebrated culture, and made real-world connections to their learning.

We observed that the FashionBot socio-cultural STEM curriculum addressed students' challenges regarding grasping the abstract concept of computing,

†Advisor - lliang@udc.edu

which led to disengagement. The students were more engaged and motivated in computing. We found that this curriculum empowered underrepresented minority students and enhanced their sense of community and shared learning experiences, broadening computer science participation.

Investigating Deepfake Detection using LIME *

Student Research Paper

Sandra L. Delancy and Lily Liang, Ph.D. †

Department of Computer Science and Information Technology
University of the District of Columbia
Washington DC 20016
sandra.delancy@udc.edu

The proliferation of deepfake images and videos has progressed exponentially in the past decade. Detecting deepfakes is more important than ever due to the potential damage these images can cause in medical, political, and social settings. Deepfake detectors typically flag an image or video as fake, but they may not explain how the prediction was achieved, which is important in establishing trust in the prediction. In this paper, we investigated Local Interpretable Model-agnostic Explanations (LIME), an open-source objects classification explainer, in detecting deepfake images of individuals. We found that we can effectively detect deepfakes by tuning LIME's parameters to highlight facial regions of an individual that positively or negatively contribute to a prediction. We configured LIME with nine classification and segmentation algorithm combinations and compared their performance in identifying deepfakes. We used the following classification algorithms: Basic Convolution Neural Network, Inception V3, ResNet50, and the following segmentation algorithms: Quickshift, Simple Linear Iterative Clustering (SLIC), and Felzenswalb. The ResNet50 classification algorithm using transfer learning and the Simple Linear Iterative Clustering (SLIC) segmentation algorithm outperforms others in this task.

†Advisor - lliang@udc.edu

Multi-Party Computation in a United States-based E-Voting System*

Student Research Paper

Ephraim N. Govere[1], *Syed Rizvi*[2]
[1]*School of Computing*
Queen's University
Kingston, ON K7L 3N6
ephraim.n.govere@queensu.ca
[2]*College of Information Sciences and Technology*
The Pennsylvania State University
State College, PA 16802
srizvi@psu.edu

Electronic voting (e-voting) has become an efficient method to process votes securely and maintain voting confidentiality, integrity, and availability. Although e-voting provides fair, fast, and private votin, the vulnerabilities associated with e-voting systems raise concerns about current digital voting landscapes, specifically the premature release of votes. Multi-Party Computation (MPC) offers a solution to e-voting concerns by securing the vote tally and ensuring results remain confidential until the election closes. This paper aims to create an MPC-based e-voting environment, showcasing its ability to secure the vote tallying process and prevent the premature release of voting results. Additionally, this study will evaluate the scalability and time efficiency of the MPC-based environment compared to an environment without MPC. This study found that an MPC-based environment is an effective method to help secure voting results and prevent the release of those results before they are authorized to be viewed by the public. Although a non-MPC-based environment is faster at tallying the voting results, the difference between the two environments is minimal, even while increasing voting participants; this shows the effectiveness of an MPC-based environment to scale itself effectively against a significant increase in participants without sacrificing computational speed.

Unveiling the Deception: Understanding the Urgent Need to Combat Deep Fake Videos*

Student Research Paper

Jada A. Danner

Computer Science Department
Bowie State University
Bowie, MD, USA
DANNERJ0901@students.bowiestate.edu

The exponential growth of deep fake technology presents a profound threat to the fabric of modern society, undermining trust in digital content and eroding the foundations of truth and authenticity. In this study, we delve into the motivations driving the creation and dissemination of deep fake videos, recognizing the urgent imperative to address this pressing global challenge. By dissecting the intricate web of incentives behind deep fake production, including political manipulation, fraud, and misinformation campaigns, we elucidate the farreaching consequences for democratic processes, public discourse, and individual privacy. The proliferation of deep fake videos exacerbates existing societal fissures, perpetuating disinformation, and sowing discord in an already polarized world. Moreover, we underscore the critical need for proactive measures to counteract the pervasive influence of deep fakes, advocating for interdisciplinary collaborations between policymakers, technologists, and civil society. Through the development of robust detection algorithms, enhanced media literacy initiatives, and regulatory frameworks, we can safeguard the integrity of digital content and fortify societal resilience against emerging threats. This study serves as a clarion call to action, highlighting the imperative of collective vigilance and concerted efforts to mitigate the deleterious impacts of deep fake videos. By fostering a deeper understanding of the motivations driving this phenomenon, we pave the way towards a more resilient and trustworthy digital ecosystem, preserving the integrity of information in an increasingly intercon-

nected world.

Keywords – Deep fake technology, Digital trust erosion, Disinformation campaigns, Detection algorithms, Media literacy initiatives

An Analysis of Blockchain Approach in AI & Cyber-Physical Systems*

Student Research Paper

Jared Robinson

Computer Science Department
Bowie State University
Bowie, MD, USA
RobinsonJ0627@students.bowiestate.edu

As the implementation of Artificial Intelligence grows, the potential vulnerabilities associated with these innovations have increased. Application of AI and Deep Learning have permeated through various avenues including Connected and Autonomous Vehicles (CAVs), Smart Cities, and other Cyber-Physical Systems (CFS). Integration of networking, computation, and physical processes are all regulated and communicated using sensors. The communication between these sensors opens the door for various threats and we seek to investigate the Blockchain (BC) approach and if its impact can reinforce exploitations presented within a CPS.

Keywords – Cyber-Physical Systems, Artificial Intelligence, Deep Learning, Smart Cities, ICRS, Blockchain.

Malware Detection in Android Phone*

Student Research Paper

Dawn Marshall, Md Kamruzzaman Sarker

Computer Science Department
Bowie State University
Bowie, MD, USA
marshalld1126@students.bowiestate.edu, ksarker@bowiestate.edu

Malware is defined as software that is specifically intended to disrupt, damage, or gain unauthorized access to a computer system. Malware detection aims to detect and prevent the harmful effects of malware using different techniques. Among different detection techniques Machine Learning is one of the most efficient because of its pattern recognition capability. Machine learning is the process of machine learning itself using algorithms and models to produce a prediction-based output for users. We propose a Machine Learning Model to detect Malware in Android Phones.

Mobile Application for Object Recognition for Visually Impaired People*

Student Research Paper

Isha Salma Renner[1], Omobolanle Favour Niyiowoeye[1], Christopher Miller[1], Maysha Fahmida[2], Md Kamruzzaman Sarker[1]

[1] Computer Science Department
Bowie State University
Bowie, MD, USA
[2] Department of Computer Science and Engineering
United International University
Dhaka, Bangladesh

Mobile applications for object recognition have become increasingly popular in recent years. These apps use image recognition technology to identify objects in images taken with a smartphone camera. They have a wide range of potential applications, from helping the visually impaired to assisting with inventory management in retail settings. Research on mobile applications for object recognition on smartphones is still ongoing.The widespread adoption of smartphones globally has created unparalleled opportunities for connectivity and communication. However, the elderly population and visually impaired individuals have been relatively underserved in terms of mobile app development. Object recognition technology uses computer vision algorithms to analyze images and classify objects based on their features, which is helpful to these populations. We propose developing an Android app to assist visually impaired individuals in recognizing and verbally identifying nearby objects as they navigate.

Designing and Prototyping a Parking Space Monitoring System with Generative AI and Large Multimodal Models*

Student Research Paper

Rachael Poffenberger, Chaz Cornejo, Weidong Liao

Department of Computer Science, Mathematics and Engineering
Shepherd University
Shepherdstown, WV 25443
wliao@shepherd.edu

In recent years, Generative AI and Large Multimodal Models have demonstrated significant advancements in image recognition, detailed responses, and complex reasoning. Our project harnesses these capabilities to monitor parking space occupancy by placing a camera above a parking lot and using Generative AI to assess the number of cars present. This system aims to provide real-time information to users, helping them determine if they should seek alternative parking. Additionally, with sufficient data, the system could predict peak hours and busy days for the parking lot.

Despite their potential, current models sometimes struggle to produce accurate and consistent vehicle counts. This study focuses on improving the reliability and accuracy of the parking space monitoring system by employing various designs, prototypes, and prompt engineering techniques with Generative AI and large multimodal models.

In this research, we have investigated the feasibility of using Generative AI and Large Multi-modal Models to develop a parking space monitoring app. As part of our investigation, a comparative study has been done to evaluate the accuracy of a variety of LMM models in the aspect of vehicle detection and counting. Our future research will focus on testing more open-source LMM models, especially small LMM models because we plan to deliver the final application on tiny computers such as Raspberry Pi. In addition, we plan to test

the responsiveness of LMM models in vehicle detection and counting.

References

1. Cameron Vu, Daria Panova, Josiah Kowalski, W. Liao, O. Guzide. Smart Parking Space Detection with Generative AI and Large Language Models. Poster presented at West Virginia Academy of Science; April 6th, 2024; Glenville, WV.

Demystifying the RSA Algorithm: An Intuitive Introduction for Novices in Cybersecurity*

*Zhengping Jay Luo[1], Ruowen Liu[2],
Aarav Mehta[1] and Md Liakat Ali[1]
[1]Department of Computer Science and Physics
[2]Department of Mathematics
Rider University
Lawrenceville, NJ, 08648
{zluo,rliu,mdali}@rider.edu

Abstract

Given the escalating importance of cybersecurity, it becomes increasingly beneficial for a diverse community to comprehend fundamental security mechanisms. Among these, the RSA algorithm stands out as a crucial component in public-key cryptosystems. However, understanding the RSA algorithm typically entails familiarity with number theory, modular arithmetic, and related concepts, which can often exceed the knowledge base of beginners entering the field of cybersecurity. In this study, we present an intuitively crafted, student-oriented introduction to the RSA algorithm. We assume that our readers possess only a basic background in mathematics and cybersecurity. Commencing with the three essential goals of public-key cryptosystems, we provide a step-by-step elucidation of how the RSA algorithm accomplishes these objectives. Additionally, we employ a toy example to further enhance practical understanding. Our assessment of student learning outcomes, conducted across two sections of the same course, reveals a discernible improvement in grades for the students.

1 Introduction

The three most widely accepted security goals of cybersecurity are shorted as "CIA triad", which stands for **C**onfidentiality, **I**ntegrity and **A**vailability. Cryptographic algorithms play a pivotal role in achieving confidentiality through private-key and public-key cryptographic algorithms. Public-key cryptographic algorithms, exemplified by the RSA algorithm, also contribute significantly to attaining another vital security goal—non-repudiation, particularly crucial in scenarios like electronic mail, where digital signatures are employed. Remarkably, the RSA algorithm was originally designed to address both confidentiality and non-repudiation goals in electronic mail [11, 16].

Developed by Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman at the Massachusetts Institute of Technology (MIT) in 1976, the RSA algorithm stands as a pioneering implementation of the public-key cryptosystem, conceptualized by Diffie and Hellman [3]. Operating with two keys—a private key and a public key—the RSA algorithm facilitates secure communication. For instance, when two parties, Alice and Bob, aim to exchange messages covertly, Alice encrypts the message M using Bob's public key, creating ciphertext C. This ciphertext is then sent to Bob, who decrypts it with their private key to retrieve the original plaintext M.

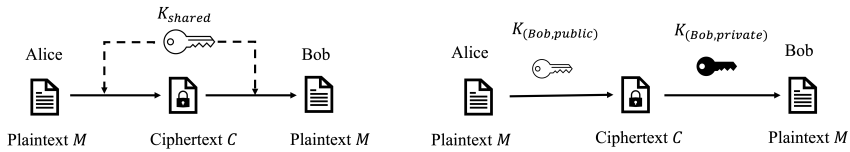
While this process may appear straightforward, generating the public and private keys involves intricate mathematical concepts such as number theory and modular arithmetic. These topics often pose challenges for beginners in cybersecurity, especially undergraduate students. In our work, we offer an intuitive and accessible perspective on understanding the RSA algorithm. Beginning with the three primary goals the RSA algorithm aims to achieve, we employ a student-oriented approach to elucidate the step-by-step design of the system. We acknowledge the potential lack of background knowledge in readers regarding number theory, modular arithmetic etc., and hence, we aim to simplify the mathematical rigor to make the content more approachable.

Additionally, we provide a practical toy example of the RSA algorithm to enhance readers' understanding. Towards the end of the paper, we present a real-world student learning outcome assessment conducted on students from two different sections of the same course. Our results demonstrate that the proposed student-oriented approach outperforms the traditional method of explaining the RSA algorithm in terms of assignment grades.

The paper is organised as follows: the necessary foundational information of the RSA algorithm is provided in Section 2. Then the detailed student-oriented style introduction of the algorithm is elaborated in Section 3. In Section 4 we employed a specific toy example to demonstrate how to encrypt and decrypt the message in RSA from a practical perspective. We concluded the paper in Section 6.

2 Background and Preliminaries

In this section, we provide necessary background that gives the context and mathematical foundations of the RSA algorithm. Readers can also skip this section and use this section as a reference while reading Section 3.



(a) Symmetric-key cryptography (b) Public-key cryptography

Figure 1: The information flow when Alice sends a message to Bob using symmetric and public key cryptography.

2.1 Symmetric-key and Public-key Cryptosystems

One of the major challenges modern cryptographies want to address is how to ensure two end users, let's say Alice and Bob, could secretly exchange messages M in an open and potentially unsafe environment. We have two strategies to tackle this challenge[7].

The first strategy is to let both Alice and Bob share a secret key K_{shared} and make sure any one of them can encrypt the plaintext M into ciphertext C using K_{shared}, while the other can recover M from C using the same key K_{shared}. This strategy is also known as symmetric-key cryptography [1]. It is similar with a real-world padlock example in which we use a key to lock a cabinet. When someone wants to open the cabinet, they need to get the same key to unlock the padlock. The process of Alice using the symmetric-key cryptography to send a message to Bob is shown in Fig. 1(a).

One of the major problems with the symmetric-key cryptography is that end users have to share the same key in advance, which is often impractical in modern communication systems such as computer networks due to: :

- In computer network systems, communication connections are usually random and instantaneously. Requiring a shared key among all the communication connections would be costly;
- Any information of the shared key sent over the open environment could be intercepted by malicious attackers, which will put the encryption out of work. Therefore, it is unrealistic to require all end users to share the same secret key in advance when they want to exchange information.

In 1976, Diffie and Hellman [3] proposed the second strategy named as public-key cryptosystems to tackle these challenges. The basic idea is that both Alice and Bob will still share the same cryptograhic algorithm, but they no longer need to share the same secret key. Instead, the system will maintain two keys: a private key and a public key. The private key is only known to the owner while the public key can be accessed by anyone who wants to communicate with the owner.

Every time if Alice wants to send a message to Bob, Alice will use Bob's public key $K_{(Bob,public)}$ to encrypt the message M. On Bob's side, the cipher-text C can be decrypted using Bob's private key $K_{(Bob,private)}$. Since only Bob has $K_{(Bob,private)}$, thus no one else could recover M. The process of Alice using the public-key cryptosystem to send a message to Bob is shown in Fig. 1(b).

In this system, the two communication entities no longer need to communicate a shared key in advance, which addresses the major problem in symmetric-key cryptography. However, one of the major disadvantages is the public-key cryptography algorithms is usually more computationally costly than symmetric-key cryptography algorithms [8, 5, 9].

The public-key cryptosystem is similar with our self-service drop box mechanism used in shipping industry. Anyone can put an envelope or a package (messages) into a public drop box (public key) provided by the shipping company (anyone could use the receiver's public key to encrypt the message in public-key cryptosystems). However, only authorised personnel (receiver) from the shipping company that has the key (private) could open the drop box to get the mails/packages.

Using public-key cryptosystems, two end users will no longer be required to share a secret key in advance when they need to exchange information. All the sender needs to know is the public key of the receiver and the cryptographic algorithm the receiver used, both of which are public information. The RSA algorithm is an implementation of the public-key cryptosystem concept.

2.2 Modular Arithmetic

Modular arithmetic is a branch of arithmetic for integers, where numbers "wrap around" when reaching a certain value. If we have a modulus n, which is an integer larger than 1, $a \bmod n$ is the remainder of a divided by n. For example, $7 \bmod 3 = 1$. The result of $a \bmod n$ for any number a will always be less than n and greater than or equal to 0, i.e., $0 \leq a \bmod n < n$. In our $7 \bmod 3 = 1$ example, obviously $1 < 3$. If $a < n$, then $a \bmod n$ will always equal to a itself. For example, $5 \bmod 9 = 5$. In the case where integers a and b have the same remainder when divided by n, i.e., $a \bmod n = b \bmod n$, we have the following definition:

Definition 2.1. If a and b are integers and m is a positive integer, then a is *congruent* to b modulo m if m divides $a-b$. We use the notation $a \equiv b (\text{mod } m)$ to indicate that a is congruent to b modulo m.

For example, as 24 and 14 have the same remainder when divided by 5, we call 24 and 14 are congruent modulo 5, which can be represented as $24 \equiv 14 (\text{mod } 5)$. In modular arithmetic, we use "$\equiv$" rather than "$=$" to denote the equivalence of modulo results. There is an important theorem of congruence that we will use in explaining the RSA algorithm:

Theorem 2.1. *If $a \equiv b(mod\, m)$ for integers a, b and m, then $ak \equiv bk(mod\, m)$ and $a^k \equiv b^k(mod\, m)$ for any integer k.*

Proof. This can be proved by the definition of congruence. Since $a \equiv b(\text{mod } m)$, then $a \text{ mod } m = b \text{ mod } m$, i.e., $a - c_1 m = b - c_2 m$ for integers c_1 and c_2. Further this can be written as $a - b = cm$ for an integer c. We multiply both sides by an integer k to get $ak - bk = ckm$, and perform modulo m on both sides will get $ak \text{ mod } m = bk \text{ mod } m$, i.e., $ak \equiv bk(\text{mod } m)$, which completes the proof. We can use similar strategies to prove $a^k \equiv b^k(\text{mod } m)$ for any integer k. $\square$

Another important theorem that we will use in proving the RSA algorithm is Bézout's theorem,

Theorem 2.2 (Bézout's theorem)**.** *If a and b are positive integers, then there exist integers s and t such that the greatest common divisor of a, b, i.e., $gcd(a, b)$, can be represented as $gcd(a, b) = sa + tb$.*

The detailed proof of this theorem can be found in [13]. The pair of s and t could be found using the *Extended Euclidean Algorithm.* For example, $gcd(24, 14) = 3 \times 24 + (-5) \times 14$. Now we give the definition of modular multiplicative inverse.

Definition 2.2. If there exist integers a, b such that $ab \equiv 1(\text{mod } m)$, then b is said to be an inverse of a modulo m and vice versa.

Based on this definition of modular multiplicative inverse and Bézout's theorem, we can derive the following theorem:

Theorem 2.3. *An inverse of a modulo m is guaranteed to be existed whenever a and m are relatively prime.*

Proof. As a and m are relatively prime, $gcd(a, m) = 1$. According to Bézout's theorem, there are integers s and t such that $gcd(a, m) = sa + tm = 1$. This implies that $sa + tm \equiv 1(\text{mod } m)$. As $tm \equiv 0(\text{mod } m)$, it follows that $sa \equiv 1(\text{mod } m)$. Consequently, s is an inverse of a modulo m. $\square$

To simplify the readability, we leave the proofs of these properties, such as the Extended Euclidean Algorithm in modular arithmetic, to the reader's interest. For those who wish to explore modular arithmetic and related theorems and proofs in greater depth, please refer to [12] for a detailed explanation.

2.3 Prime Factorisation

Prime factorization means the decomposition, if possible, of a positive integer into a product of prime integers. For example, the prime factorization of 15 is 3×5, in which both 3 and 5 are prime numbers. Prime factorization is an important problem in number theory because still no efficient enough way has been discovered to find the prime factorization of an extremely large integer with existing classical computer systems.

The RSA algorithm embeds prime factorization in its design to ensure there exists no efficient way to decipher the ciphertext in non-quantum computing systems. However, it does not mean that we would not find an efficient way to perform prime factorization in the future based on nowadays computer technology (a lot of mathematicians are still working on this problem); it also does not mean that we would not find an efficient way on future computers, such as quantum computing [14, 6, 4]. In fact, an efficient way to perform prime factorization on quantum computers has already been found [15]. The problem is that a workable quantum computer is still estimated to be at least decades away [2]. Therefore, we can safely say the RSA algorithm is secure at least for the time being.

2.4 Euler's Theorem

Before introducing Euler's theorem, let's first provide the definition of Euler's totient function:

Definition 2.3. The Euler's totient function $\phi(.)$ is the number of positive integers that are less than and relatively prime to this integer, i.e., $\phi(n) =$ the number of integers in$\{1, 2, 3, \ldots, n-1\}$which are relative prime to n.

For example, given an integer 8, there exist four integers $1, 3, 5, 7$ that are relatively prime to 8, thus Euler's totient function value $\phi(8) = 4$. You might have already realised that Euler's totient function value for a prime number n is always $n-1$, i.e., $\phi(n) = n-1$, as all the $n-1$ positive integers less than n are relative prime to n. An important mathematical property of Euler's totient function is that:

Theorem 2.4. *If m and n are relatively prime integers, then $\phi(mn) = \phi(m) \times \phi(n)$.*

For example, $\phi(6) = \phi(2) \times \phi(3) = 1 \times 2 = 2$. We'll skip the proof here and the detailed proof of this theorem can be found in [13]. This property offers a convenient way to calculate Euler's totient function value if an integer can be factorized into the product of two prime numbers m and n. In this case $\phi(mn) = \phi(m) \times \phi(n) = (m-1)(n-1)$ as m, n are also relatively prime to each other, which we will use later in proving the RSA algorithm. The challenge here is that no efficient way has been found on modern computers to do prime factorization (as discussed in Section 2.3).

It is worth noting that the complexity of prime factorization and computing the Euler's totient function is equivalent for arbitrary integers. Essentially, both require evaluating whether the integer is relative prime to all the positive integers less than it. Therefore, it is also computationally difficult to calculate Euler's totient function for large enough integers. Now we're ready to introduce Euler's Theorem.

Theorem 2.5 (Euler's Theorem)**.** *If two integers a and n are relatively prime, i.e., $gcd(a, n) = 1$, and $n > 0$, then $a^{\phi(n)} \equiv 1 \ (mod\, n)$.*

For example, let $a = 3$ and $n = 4$, then they are relatively prime and we have $\phi(4) = 2$. Further we have $3^{\phi(4)} = 3^2 = 9$, thus, $3^{\phi(4)} \equiv 9 \equiv 1 \pmod 4$. We leave the proof of Euler's theorem to the readers due to the abundance of online resources on this topic [13]. It is worth noting that Euler's theorem provides a fast way to calculate $a^{\phi(n)} \bmod n$ when a, n are relatively prime. This property plays a significant role in the RSA algorithm as we will see in the following section.

After all the background information introduction, now we're ready to start the introduction of the RSA algorithm, which is an implementation of the public-key cryptosystem.

3 The RSA algorithm

The RSA system was introduced in 1976. Now it is one of the most widely used public-key encryption methods in computer networks. To materialise a public-key cryptosystem, as we introduced in Section 2.1, we want to achieve the following three basic goals [11]:

1. **Efficiency:** The encryption and decryption process should be easy to compute for legitimate users who have the required key information.

2. **Plaintext recovery:** We should be able to get the original plaintext M through decrypting the ciphertext C.

3. **Computational difficulty:** Without the private key information, there is no known efficient way to perform the decryption process.

These three goals are critical in the success of the public-key systems. With these three goals in mind, we introduce the core encryption and decryption process of the RSA algorithm. The corresponding ciphertext C of the plaintext M is computed from

$$C \equiv M^e \pmod{n}. \tag{1}$$

e and n is the public key information of the receiver. The decryption process is similar, which is

$$M' \equiv C^d \pmod{n}. \tag{2}$$

The private key information consists of d and n. We use M', not M directly in Eq. (2) because we want to highlight that this is the result we obtained from the decryption process. We will ensure $M' = M$ in the *plaintext recovery* goal.

Suppose Alice wants to send a secret message $M = 2$ to Bob using the RSA algorithm. Bob's public key (e, n) is $(113, 143)$ and the corresponding private key (d, n) is $(17, 143)$, which means that the ciphertext $C \equiv M^e \pmod{n} \equiv 2^{113} \pmod{143} \equiv 19$. Alice will send out $C = 19$ to Bob. Bob can then decrypt the ciphertext to recover the plaintext through $M^{'} \equiv C^d \pmod{n} \equiv 19^{17} \pmod{143} \equiv 2$, which achieved the goal of $M^{'} = M$. The detailed encryption and decryption process of the RSA algorithm is shown as follows in Algorithm 1.

Algorithm 1 The encryption and decryption process of the RSA algorithm.

The Receiver:
- Choose two large random prime numbers p and q privately.
- Obtain n and $\phi(n)$ through $n = p \cdot q$ and $\phi(n) = (p-1)(q-1)$, then keep p and q in private or destroy them.
- Choose a large number e that is relatively prime to $\phi(n)$.
- Compute d such that $ed \equiv 1 (\mathrm{mod}\ \phi(n))$.
- Release (e, n) to the public and keep (d, n) as the private key.

The Sender:
- Encrypt the message M using the receiver's public key (e, n), $C \equiv M^e \pmod{n}$, and send the ciphertext C to the receiver.

The Receiver:
- Decrypt the received ciphertext C using their own private key (d, n) to recover $M' \equiv C^d \pmod{n}$.

We now need to understand what conditions must be satisfied and how this process could achieve the three goals mentioned above. We will explain each goal with the associated conditions as follows.

3.1 Goal 1: Efficiency

Both encryption and decryption procedures are identical from an implementation perspective, making them straightforward to implement in practice. Additionally, private and public keys can be determined using standard and efficient methods on modern computers [10].

We also need to be able to find $M^e \pmod n$ and $C^d \pmod n$ efficiently without using an excessive amount of memory given that e, d, n are all large numbers. Directly computing the exponentiation operation of M^e or C^d is impractical, as their results can be very extremely large and require significant memory to store. Fortunately, this problem can be addressed using the fast modular exponentiation algorithm, which reduces the computational complexity to a logarithmic level. The detailed algorithm is provided in [12].

However, despite the RSA algorithm's careful design for efficiency, it is generally accepted that public-key cryptosystems are usually less efficient than symmetric-key cryptosystems. Therefore, in real-world scenarios, the RSA algorithm is primarily used for delivering the pre-shared key in symmetric-key cryptosystems, which is often a short message. When encrypting large amounts of information, symmetric-key cryptosystems are still preferred for their efficiency [8].

3.2 Goal 2: Plaintext Recovery

The second goal is to guarantee the accurate recovery of original plaintext M from ciphertext C using receiver's private key (d, e), i.e., to ensure $M' = M$. Substituting C in the encryption process as shown in Eq.(1) to the decryption process as shown in Eq.(2), it yields

$$M' \equiv [M^e \pmod n]^d \equiv M^{ed} \pmod n. \tag{3}$$

As we know from Section 2.2, M could also be written as

$$M \equiv M \pmod n, \text{ if } M < n. \tag{4}$$

Therefore, the goal can be reinterpreted as finding the conditions to guarantee

$$M^{ed} \equiv M \pmod n, \text{with } M < n. \tag{5}$$

As long as $M < n$, the above equation will hold. According to Euler's theorem (Section 2.4), if M and n are relatively prime, then $M^{\phi(n)} \equiv 1 \pmod n$. By the modular arithmetic properties (Section 2.2), we can raise both sides to the k-th power, with k being a positive integer, to get $M^{k\phi(n)} \equiv 1^k \equiv 1 \pmod n$. Multiplying both sides by M yields,

$$M^{k\phi(n)+1} \equiv M \pmod n. \tag{6}$$

Comparing Eq.(3) to Eq.(6), to ensure the correct recovery $M' = M$, we would now require

$$M^{ed} \pmod n = M^{k\phi(n)+1} (\text{mod } n) \tag{7}$$

i.e., we need

$$ed = k\phi(n) + 1 \text{ , } k \text{ is a positive integer.} \tag{8}$$

Up until now, we found that we have two conditions need to be satisfied in order to make above equations hold: (1) $M < n$ and (2) M and n are relatively prime. As long as these two conditions are satisfied, the above derivation from Eq.(3) to Eq.(8) will hold. To satisfy the first condition, in real world, after choosing the large positive number n, we need to break long messages into small blocks such that each block can be represented as an integer M that is less than n. We will explain how to ensure the second condition in Section 3.3.

We now know that if we could find a pair of e, d such that $ed = k\phi(n) + 1$,k is a positive integer. The two conditions for M and n are satisfied, then we're confident that the original plaintext M could be recovered from C. In the next section, we'll see how these conditions are met and at the same time the *computational difficulty* goal is also achieved.

3.3 Goal 3: Computational Difficulty

Now the challenge is reduced to a problem of finding appropriate values of e and d, which are the major components of the public and private key respectively. The only clue we have now is $ed = k\phi(n) + 1$, where k is a positive integer.

To achieve the third goal of computational difficulty, we will start with the challenge of how to choose e and d. Let's first manipulate the equation a little bit. Given that $ed = k\phi(n) + 1$, when the modulus is $\phi(n)$, we have

$$ed \equiv k\phi(n) + 1 \equiv 1 \pmod{\phi(n)}, \tag{9}$$

where the last congruent relation comes from the fact that k is a positive integer. The congruence we get from the above manipulation $ed \equiv 1 \pmod{\phi(n)}$ reveals that if e and $\phi(n)$ are relatively prime, then d is an inverse of e modulo $\phi(n)$ and the existence of d is guaranteed according to the Bézout's theorem (Section 2.2).

Now we just need to find a number e that is relatively prime to $\phi(n)$, and the corresponding inverse modulo n, denoted by d. Finding a number e that is relatively prime to $\phi(n)$ should not be a difficult problem if given $\phi(n)$. Finding the corresponding inverse d of e modulo $\phi(n)$ could be done through *the Extended Euclidean Algorithm* efficiently as $gcd(e, \phi(n)) = 1$.

We have successfully found a way to find an appropriate e and d. However, this does not conclude the problem. In the third goal of public-key cryptosystems, it requires that there exists no known efficient way to calculate d given the information of e and n. Obviously, we still have not reached that goal. If n is not chosen carefully, an attacker might be able to easily figure out the value of $\phi(n)$ and further efficiently figure out d based on e.

Achieving the last goal of the public-key cryptosystems is one of the most elegant parts of the RSA algorithm. We know that there exist no known efficient method to perform prime factorisation(Section 2.3). If the receiver can first find two large random prime numbers p and q privately and let $n = p \cdot q$, then there will exist no efficient way to reverse this process to get p and q from only n. Further, it will be computationally difficult to get the value of $\phi(n)$ as stated in Section 2.4.

However, it will be super easy for the valid receiver to calculate $\phi(n)$ as $\phi(n) = (p-1)(q-1)$. This is also known as the "trap-door one-way function", which is similar with how our shipping drop box works.

Finally we have achieved all the three goals mentioned at the beginning. The receiver just needs to first choose two large enough prime numbers p and q, and get $n = p \cdot q$ and $\phi(n) = (p-1)(q-1)$. Then p and q can be destroyed to prevent potential leaks. The receiver can further get the public key (e, n) by choosing a large enough e that is relative prime to $\phi(n)$ and then the private key (d, n) could be computed based on $ed \equiv 1(mod\ \phi(n))$. As there's no efficient way to compute $\phi(n)$ based on n as it requires a prime factorization, thus the third goal of computation difficulty will be achieved.

We still have one last question left unanswered from Section 3.2. How can we ensure n and M to be relatively prime? Unfortunately, we cannot ensure it directly. However, we know that $n = p \cdot q$ with p, q being prime, which means n will be relatively prime to all numbers less than n except p, q and their multiples. The only case in which M and n are not relatively prime is when M is a multiple of p or q or both, which has an extremely low chance in terms of probability considering we also require $M < n$ in Goal 2.

Up until this point, all the requirements to achieve the three goals of public-key cryptosystems are satisfied. In the following section we provide a toy example to sort out the process.

4 A Toy Example

The detailed implementation specifications of the RSA algorithm in real world can be found in [10]. Suppose Alice wants to send a message "Tue 7PM" to Bob secretly using the RSA algorithm. First, Bob needs to decide his private key (d, n) and public key (e, n) for the communication. Bob will choose two large random prime numbers p and q. Let's assume $p = 1721$ and $q = 1801$. In real world, these two numbers should be much larger such that it is unrealistic for modern computers to obtain the prime factors p and q from n. n can be computed as $n = p \cdot q = 3099521$. We can also obtain Euler's totient function of n as $\phi(n) = (p-1)(q-1) = 3096000$.

The next step for Bob is to choose a public key e, which is a number relatively prime to $\phi(n)$. For example, the standard sizes for RSA keys starts from 512 bits. To get a very high-strength key, the key size requires 4096 bits. Here in our toy example we choose $e = 1012333$. Now Bob needs to compute the private key d. Based on the equation $ed \equiv 1 \pmod{\phi(n)}$, we could get the inverse of e modulo $\phi(n)$ as $d = 997$ using the Extended Euclidean Algorithm. After e and d are determined, p and q can be destroyed or hidden for the sake of security. Bob can release his public key (e, n) to the public while keep d private.

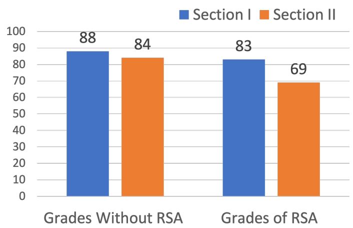
From Alice's perspective, Alice needs to first obtain Bob's public key (e, n), then she could convert the message she wants to send into its numerical representations. Here we use ASCII (American Standard Code for Information Interchange) to convert "Tue 7PM" into numerical representation as: 084 117 101 032 055 080 077.

If the message is too long, Alice could divide the message into smaller blocks, then encode each block separately. Here we divide the message into blocks that has 3 digits in each of them. There are seven blocks in the message including the space. With the public key $(e, n) = (1012333, 3099521)$, Alice could obtain the ciphertext through $M^e \pmod{n}$ to get $084^{1012333} \equiv 469428 (\bmod\ 3099521), 117^{1012333} \equiv 547387 (\bmod\ 3099521), \ldots\ldots.$ The complete ciphertext C is shown as "0469428 0547387 2687822 1878793 0330764 1501041 1232817". When Bob receives the ciphertext, he will decrypt the ciphertext using his own private key $(d, n) = (997, 3099521)$ to get $0469428^{997} \equiv 84 (\bmod\ 3099521), 0547387^{997} \equiv 117 (\bmod\ 3099521), \ldots\ldots, 1232817^{997} \equiv 77 \pmod{3099521}$. Finally he recovers the original message by looking up the ASCII table to get the plaintext message "Tue 7PM".

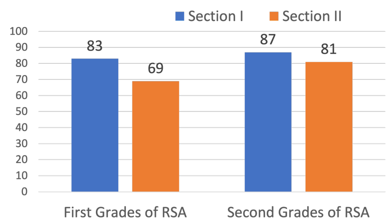
5 Student Learning Outcome Assessment

To study the effectiveness of the proposed student-oriented approach in explaining the RSA algorithm, we conducted a comparative analysis with the traditional method outlined in [12]. In the traditional method, the encryption and decryption process are presented upfront to the students, followed by the corresponding proof utilising number theory knowledge to enhance comprehension of the algorithm. The explanatory style from [12] presents the conventional approach to teaching the RSA algorithm.

The comparison involved two sections of the same course, namely *CSC 140 Discrete Structures* at Rider University. These sections comprised 24 and 26 undergraduate students, respectively, all majoring in computer science or cy-



(a) "Grades Without RSA" refers to the average grades of assignments unrelated to the RSA algorithm, which are taught in the same manner; "Grades of RSA" represents the average grades related to the RSA algorithm, which are taught differently.



(b) "First Grades of RSA" represent the averaged grades of the assignment related to the RSA algorithm for the two sections; "Second Grades of RSA" refer to the averaged grades students received after the alternative way is offered.

Figure 2: Students learning outcome comparison in terms of assignment grades from two sections of the same course.

bersecurity. Given that this is a 100-level course and a prerequisite for several higher-level courses, the majority of students are either freshmen or sophomores, aligning with the target readership of this paper.

In these two sections, all course content, excluding the RSA algorithm section, followed the same instructional format. Equal lecture time was allocated to each topic in both sections. Student performance was compared based on related assignment grades. Both sections were presented with identical assignment problems and grading criteria.

The study involved initially employing the proposed student-oriented method outlined in this work for students in Section I and the traditional method from [12] for students in Section II. Subsequently, a related assignment was administered. Following this, both sections were exposed to an alternative introduction method—Section I students were presented with the traditional explanation, while Section II students were introduced to the proposed student-oriented approach. Finally, a makeup opportunity for the assignment was extended to all students. Detailed results are presented in Fig. 2.

In Fig. 2 (a), we initially compared two categories of student grades: "Grades Without RSA" and "Grades of RSA." The former represents the averaged grades for all assignments throughout the semester, excluding the one related to the RSA algorithm. With a total of 9 assignments for the entire semester, all topics pertaining to these assignments are taught in the same way. Our analysis revealed that students from Section I performed, on average, 4 points higher than those from Section II (each assignment is out of 100 points).

On the other hand, "Grades of RSA" focuses solely on the assignment related to the RSA algorithm, considering a single assignment. Our findings indicated that students in Section I outperformed those in Section II by an impressive average margin of 14 points. If the effectiveness of the teaching methods were equal for both sections, we would anticipate a much smaller average grade difference than the observed 14 points. Consequently, these results underscore the effectiveness of the student-oriented approach in explaining the RSA algorithm compared to the traditional method.

Upon offering both sections the alternative teaching method, we observed an improvement in grades for both groups (Fig. 2 (b)). However, the gap in grades between the two sections narrowed from 14 points to 6 points. This reduction further validates the efficacy of the student-oriented teaching approach.

6 Conclusion

As the significance of cybersecurity continues to rapidly increase across various facets of society, comprehending the fundamental logic behind widely used security mechanisms becomes essential not only for cybersecurity students but also for a broader audience. In this study, we present a self-contained and student-oriented interpretation of the RSA algorithm, a cornerstone in public-key cryptosystems. Beginning with three goals of public-key cryptosystems, we guide readers through a step-by-step explanation of how the RSA algorithm satisfies and implements each of these three goals. Our student learning outcome assessment, conducted across two different course sections, demonstrated the effectiveness of our approach, with an average grade difference of 14 points compared to the traditional method of teaching the RSA algorithm.We envision this work serving as a more approachable channel for readers to grasp the intricacies of the RSA algorithm.

References

[1] R Anusha et al. “Symmetric Key Algorithm in Computer security: A Review”. In: *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE. 2020, pp. 765–769.

[2] Daniel J Bernstein and Tanja Lange. “Post-quantum cryptography”. In: *Nature* 549.7671 (2017), pp. 188–194.

[3] Whitfield Diffie and Martin E Hellman. “New directions in cryptography”. In: *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*. 2022, pp. 365–390.

[4] Chuck Easttom. “Quantum computing and cryptography”. In: *Modern Cryptography: Applied Mathematics for Encryption and Information Security.* Springer, 2022, pp. 397–407.

[5] Reza Fotohi, Somayyeh Firoozi Bari, and Mehdi Yusefi. “Securing wireless sensor networks against denial-of-sleep attacks using RSA cryptography algorithm and interlock protocol”. In: *International Journal of Communication Systems* 33.4 (2020), e4234.

[6] Jack D Hidary and Jack D Hidary. *Quantum computing: an applied approach.* Vol. 1. Springer, 2019.

[7] Raza Imam et al. “Systematic and critical review of rsa based public key cryptographic schemes: Past and present status”. In: *IEEE Access* 9 (2021), pp. 155949–155976.

[8] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography CRC Press.* 2020.

[9] Dwi Liestyowati. “Public key cryptography”. In: *Journal of Physics: Conference Series.* Vol. 1477. 5. IOP Publishing. 2020, p. 052062.

[10] Kathleen Moriarty et al. *PKCS# 1: RSA cryptography specifications version 2.2.* Tech. rep. 2016.

[11] Ronald L Rivest, Adi Shamir, and Leonard Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[12] Kenneth H Rosen. *Discrete mathematics and its applications.* The McGraw Hill Companies, 2019.

[13] Kenneth H Rosen. *Elementary number theory.* Pearson Education London, 2011.

[14] Engineering National Academies of Sciences, Medicine, et al. “Quantum computing: progress and prospects”. In: (2019).

[15] Peter W Shor. “Algorithms for quantum computation: discrete logarithms and factoring”. In: *Proceedings 35th annual symposium on foundations of computer science.* Ieee. 1994, pp. 124–134.

[16] Osama Fouad Abdel Wahab et al. “Hiding data using efficient combination of RSA cryptography, and compression steganography techniques”. In: *IEEE access* 9 (2021), pp. 31805–31815.

Design and Development of the FlexBE WebUI with Introductory Tutorials*

Samuel Raymond, Grace Walters, Josh Luzier, David C. Conner

Capable Humanitarian Robotics and Intelligent Systems Lab
Department of Physics, Computer Science and Engineering
Christopher Newport University
Newport News, VA 23606

{samuel.raymond.21, grace.walters.21, joshua.luzier.19, david.conner}@cnu.edu

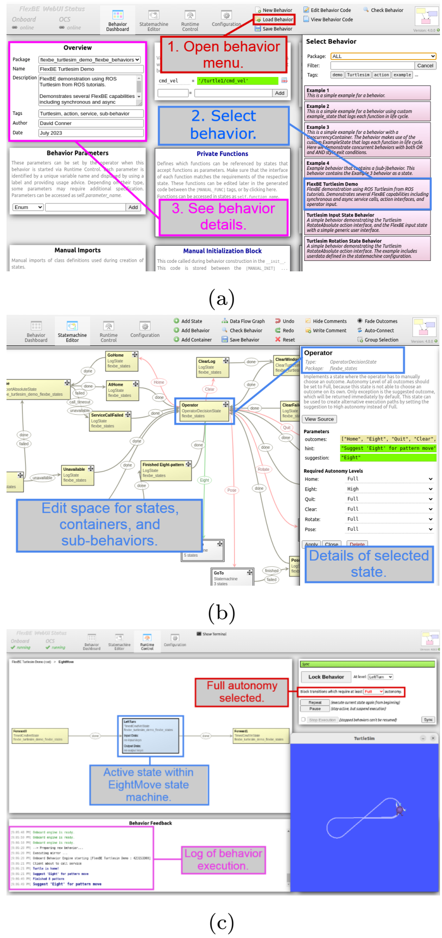
Abstract

Hierarchical Finite State Machines (HFSMs) continue to be a popular strategy for high-level behavioral control in robotics. Within the Robot Operating System (ROS) ecosystem, the Flexible Behavior Engine (FlexBE) provides an easy to use Python implementation with an intuitive user interface for runtime monitoring and "collaborative autonomy." In this paper, we describe the latest evolution of the FlexBE UI, that leverages FastAPI to provide a Python-based native ROS node interface, to a web-based user interface. Additionally, the paper describes a "quick-start" introductory tutorial to state machines using FlexBE. This system provides a natural introduction to HFSM-based techniques.

1 Introduction

Robotic systems are typically implemented using distributed interacting processes that require high-level coordination to induce the desired overall system behavior. As seen in the DARPA Urban (2007) and Robotics (2015) Challenges, hybrid architectures that include low-level reactive controllers with high-level deliberative mission planners are commonly used. These different components are coordinated using *behavioral control systems*, which are commonly implemented using either Hierarchical Finite State Machines (HFSM) or Behavior Trees (BT).

In this paper we focus on HFSM, and describe recent work improving the user interface for the Flexible Behavior Engine (FlexBE[1]) that was first developed in the DARPA Robotics Challenge[1, 2, 3]. Furthermore, we introduce an accessible introductory tutorial. Figure 1 shows views from our tutorial demonstration of the capabilities of FlexBE in particular and HFSM in general. This tutorial is being used in our senior level *Introduction to Robotics* course at Christopher Newport University.



(a) (b) (c)

Figure 1: FlexBE UI showing Turtlesim demonstration. (a) Loading a predefined behavior from the FlexBE behavior dashboard. (b) The behavior editor view. (c) The FlexBE runtime monitor view overlaid with the ROS Turtlesim window. In Full Autonomy mode, as shown here, the EightMove (sub-)state machine is repeatedly activated.

Section 2 begins with an overview of ROS and the Flexible Behavior Engine, along with related works. Afterwards, Section 3 describes the design and development of a new user interface for FlexBE. Section 4 presents our "quick-start" tutorials for FlexBE. While the open-source distribution provides details for installing and running

[1] `https://github.com/FlexBE`

the demonstrations,[2] this paper documents their relationship to key FlexBE design features, with specific changes relevant to the FlexBE Web UI. Finally, Section 5 concludes the paper and outlines future work for the project.

2 Background

2.1 Robot Operating System

The Robot Operating System (ROS) is a framework for writing robot software initially released in 2007 by Willow Garage[4, 5, 6]. ROS, which is not an operating system, provides a structured communications *middleware* layer above the host operating system. In addition to this middleware definition and messaging conventions, ROS provides an ecosystem of utilities, libraries, and drivers to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. ROS's ecosystem of loosely coupled tools and libraries enables software distribution and reuse of cutting edge algorithms to facilitate robotics research and development.

The original middleware release, now called ROS 1, used TCP and UDP broadcast communications. The final ROS 1 release, dubbed Noetic, has been supplanted by ROS 2, which uses a Data Distribution Service (DDS) protocol for its middleware[6].[3] As of this writing, ROS 2 Jazzy Jalisco is the latest long-term support release.[4]

In addition to the base ROS middleware, ROS has numerous open-source packages for common robotic software tasks such as planning, navigation, and control.

2.2 Behavioral Control Systems

Robotic systems are complex systems of systems most commonly implemented with interacting asynchronous software components in a multilayer hybrid framework that mixes slower deliberative software components with faster reactive components[7, 8, 9]. Two common approaches to implementing the behavioral control coordination are Hierarchical Finite State Machines (HFSM) and Behavior Trees (BT)[10].

Starting with the latter, Behavior Trees are the newer technology initially developed in the computer gaming industry to model non-player characters (NPCs) [11, 12]. Like robotic systems, NPCs are complex *hybrid dynamical systems* that mix continuous movement components with discrete decision

[2] https://github.com/FlexBE/flexbe_webui
[3] https://docs.ros.org/en/jazzy/Concepts/Intermediate/About-Different-Middleware-Vendors.html
[4] https://docs.ros.org/en/jazzy/Releases/Release-Jazzy-Jalisco.html

making components. As implied by the name, a BT is a *directed acyclic graph* or *tree* [11, 12]. Several BT libraries, such as `BehaviorTree.CPP`[5], `py_trees`[6], and `ros_bt_py`[7] have been created for robotics and incorporated in ROS[13].

Both HFSM and BT approaches are often implicit within other ROS packages. The ROS 1 basic `navigation` system used HFSM to coordinate planning, control, and recovery behaviors; the Move Base Flex navigation system used `py_trees`-based BTs [14]. With the introduction of ROS 2, the `navigation2` package uses BTs with `BehaviorTree.CPP` for task orchestration [15].[8]

In this paper, we focus on the use of HFSM. While HFSM are an older technology, popularized under the *State Chart* formalism[16], they are widely applicable. It has been argued that HFSM have advantages over BT in some applications, especially for cyclical operations and as a *script* for supervisor/operator interaction[2, 17]. The early ROS 1 package SMACH (State MACHine) allowed users to define and execute HFSMs on a robot.[9]

2.3 HFSM and FlexBE - the Flexible Behavior Engine

In the DARPA Robotics Challenge, from 2012 to 2015, it was necessary for the supervisor/operator to interact with a robot system during execution to preempt behaviors and reconfigure the robot behaviors at runtime in an evolving disaster response scenario; this was called *collaborative autonomy* [1, 2, 3]. As SMACH was not designed to provide collaborative autonomy, Team ViGIR developed the Flexible Behavior Engine (FlexBE) as a major extension to SMACH. FlexBE supports adjustable autonomy, preemptive state transitions, and online adjustments to behaviors that support collaborative autonomy. Like SMACH, FlexBE enables the passing of *user data* from one state to the next.

The FlexBE Onboard Behavior Executive (OBE) runs "onboard" the robot and coordinates execution of the Python-based state implementations. The desired system *behavior* is induced via the invocation of specific Python state implementation class instances that encode the HFSM according to the desired transitions. The OBE manages how these states publish and subscribe to various topics and invoke ROS actions and services [1, 3]. Separately on an Operator Control Station (OCS), a desktop graphical user interface (FlexBE UI) enables development of HFSMs and monitors their execution in real time. The UI allows the user to edit and define behaviors as state machines, and then write the Python script that defines the overall state machine. These Python scripts are executed by the OBE. Meanwhile, the OCS *mirror* node functions

[5] `https://github.com/BehaviorTree/BehaviorTree.CPP`
[6] `https://github.com/splintered-reality/py_trees`
[7] `https://github.com/fzi-forschungszentrum-informatik/ros_bt_py`
[8] `https://github.com/ros-planning/navigation2`
[9] `http://wiki.ros.org/smach`

as an intermediary between the OBE and FlexBE UI by "mirroring" the status of the OBE and reporting it to the FlexBE UI.

The FlexBE User Interface was originally written in JavaScript as a Chrome browser application by Philipp Schillinger[1, 18]. In 2017, as Google stopped supporting Chrome Apps,[10] Schillinger developed the FlexBE App[11] as a standalone application using the `NW.js` runtime library.[12] This library was not part of the standard ROS setup and required a separate install step; this prevented the `flexbe_app` from inclusion in the standard ROS binary distributions.

At the time of the `flexbe_app` development, there did not exist a native interface between JavaScript and ROS. As a workaround, the `flexbe_app` uses custom Python scripts run as hidden subprocesses to send and receive ROS messages, and redirects the standard output to enable processing by the JavaScript code. This approach worked, but made debugging new messages a challenge. The new `flexbe_webui` presented in this paper addresses this issue and introduces new features.

Since its release in 2016, FlexBE has been widely used[10]. Several extension packages support integration with ROS navigation and manipulation packages[19, 20, 21]. These aim to decouple the tightly integrated nature of existing ROS navigation and manipulation packages with their implicit behaviors to enable collaborative autonomy. These have the added educational benefit of allowing students to better appreciate the multiple steps that are required for simple navigation tasks (e.g., receiving goal, planning path, and path following). By following the common ROS *action* paradigm, these extensions also allow for flexible integration of future technologies. Recent work proposed the integration of FlexBE-based HFSM with Behavior Trees[17].

3 FlexBE WebUI

To support the continuing work on FlexBE, over the past year our group has developed a new FlexBE WebUI tool and has publicly released the source code.[13] The FlexBE WebUI concept was initially proposed by Schillinger in early 2023 and transferred to our group for development. This approach supports the standard ROS dependency model and will target a ROS 2 Jazzy Jalisco binary release in late 2024.

The FlexBE WebUI uses native Python nodes with `FastAPI`[14] and `uvi-`

[10] `https://github.com/pschillinger/flexbe_chrome_app/issues/11`
[11] `https://github.com/FlexBE/flexbe_app`
[12] `https://nwjs.io/`
[13] `https://github.com/FlexBE/flexbe_webui`
[14] `https://fastapi.tiangolo.com/`

`corn`[15] to define a web server in Python and allows the use of native ROS 2 Python nodes communicating with DDS. This also allows all of the standard ROS 2 and Python 3 debugging tools, while retaining the existing user interface defined by the core FlexBE App JavaScript code. As such, the interface between the JavaScript code and ROS has been simplified. Rather than creating Python subprocesses locally, a client browser sends HTTP requests to the web server for specific ROS functionality. In addition to leveraging the communication advantages of native Python nodes, transitioning to a client-server model is beneficial for those seeking to deploy FlexBE on a network; a client can access the FlexBE OCS software running on a separate device. This setup supports client access via a standard web browser or as a standalone client using the `PySide6 QtWebEngine`.[16]

We retained the basic user interface but added several enhancements. One common issue in the original FlexBE App (`flexbe_app`) was overlapping transition labels between single states. The new FlexBE WebUI now allows the user more flexible control of the curves used to define transitions between states, and more precise control over the position of their labels as shown in Figure 2.

In the older `flexbe_app`, state-to-state transitions were auto-drawn using a cubic-bezier S-curve. Start and end points were calculated based on the shortest path between two states, and the transition label was automatically placed in the middle of the curve. However, since transitions and labels were placed automatically, they would frequently overlap, and the only potential user solution was to rearrange their states. With many interconnected states, this could get messy very easily.

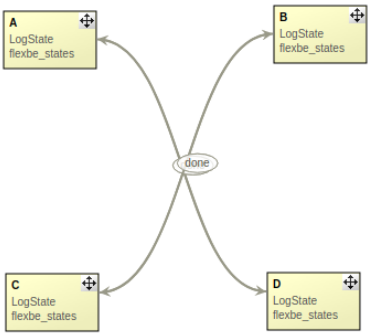
In the new `flexbe_webui`, transitions are drawn using a Catmull-Rom Spline [22]. A unique feature of the Catmull-Rom spline is that it goes through all of its control points, which the Bezier curve did not. This allows the transition label itself to be a control point. The initial curve still auto draws based on shortest path when the transition is initially connected, but the `flexbe_webui` user has more control over the shape of the curve. They can move the end-points to be anywhere along the box of the start or end state. They can also drag the transition label to change the shape of the curve.

Similarly, during runtime, distinct outcomes are now shown, where in the original version outcome labels between common states would overlap.

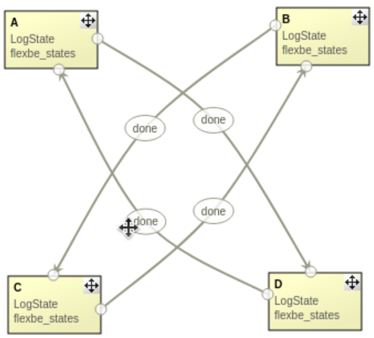
Another quality-of-life issue from the FlexBE App was input validation for state parameters. Now, input fields have color coded backgrounds based on the input type, as shown in Figure 1b, and the UI performs additional data validation checks prior to saving a behavior. There are additional improvements

[15] `https://www.uvicorn.org/`
[16] `https://doc.qt.io/qtforpython-6/quickstart.html`

to key stroke bindings and tab handling between edit fields. For further information and user guide see the official documentation at `https://flexbe.readthedocs.io/en/latest/`.



(a) FlexBE App



(b) FlexBE WebUI

Figure 2: (a) The state transition curves and labels can overlap in the original FlexBE App. (b) The FlexBE WebUI allows the user to modify positioning of curves and labels.

The August 2024 official release of the WebUI coincides with the official release of version 4.0.0 of the FlexBE Behavior Engine[17].

This version of the FlexBE WebUI includes extensions to the FlexBE Behavior Engine that improves mirroring and synchronization of concurrent states between the onboard and OCS mirror[1, 18]. A key feature of the FlexBE design is the ability of a remote operator to monitor and control the onboard state machine; therefore, keeping the OCS aware of the current state of the onboard behavior is crucial. FlexBE does this with a "mirror" state machine that follows the discrete transitions of the onboard state machine without the functionality; the active states and potential transitions are then shown on the UI. A design goal was to meet this functionality while keeping the required bandwidth low. Earlier versions of FlexBE did this with an "outcome" message that sent the outcome label as a string. The latest 4.+ version uses an unique 23-bit numeric hash value assigned to each state as the `state_id`; the outcome of a state is encoded using an additional 8-bits. This allows a single 32-bit number to encode the exact state and current outcome (or none) during operation. Both the transition outcomes and status heartbeat messages now use this `state_id` for consistency. In the case of rapidly changing states

[17]Version 4.0.0 changes the API and is no longer compatible with earlier 3.0.x versions of the `flexbe_behavior_engine` and the `flexbe_app`. Pre-release versions of `flexbe_webui` are available that are compatible with the 3.0.x versions.

that may be nested in hierarchical containers, inconsistencies arising due to out of order messages can be quickly identified and corrected to keep the OCS "mirror" synchronized with the onboard behavior.

4 ROS 2 FlexBE Turtlesim Tutorials

To demonstrate FlexBE's capabilities, we have developed a set of simple introductory tutorials based around the ROS 2 release of FlexBE. Complete installation and operation directions are available open-source.[18] Here we provide an overview and motivation for each demonstration of FlexBE's capabilities.

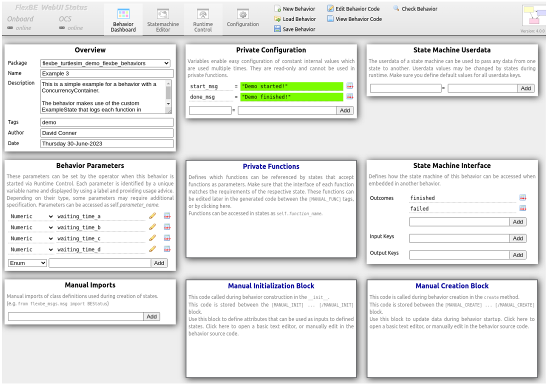
The distribution includes four basic examples built using a simple text-based `LogState`. These examples only require a basic ROS installation with the FlexBE Behavior Engine and user interfaces (whether the classic FlexBE App or the new FlexBE WebUI). Example 3 is shown in Figure 3; the other examples are not pictured but can be viewed and run from the open-source distribution.

Example 1 includes the "hello world" equivalent with FlexBE. This simple state machine includes a `LogState` to display a message and a `WaitState` that delays before completing the state machine execution. The Example 1 behavior demonstrates the use of the FlexBE Behavior Dashboard to load pre-existing behaviors, define specific variables, and provides a minimal demonstration of state transitions. The Example 1 documentation describes the structure of the Python state implementations.

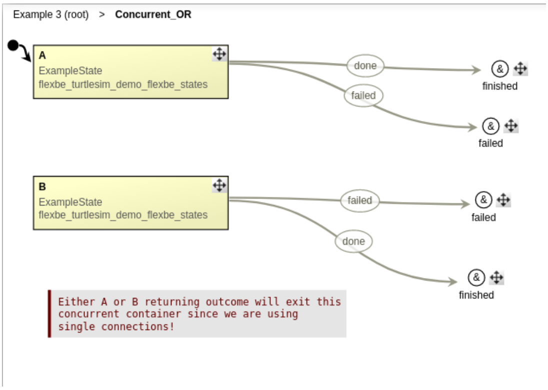
Example 2 demonstrates the use of a custom `ExampleState` implementation with multiple outcomes. This demonstrates how to construct new customized state implementations.

Example 3 constructs a new state machine that includes two `ConcurrencyContainer`s that each contain simple state machines using the `ExampleState`. The first concurrency container demonstrates an `OR` behavior where either `ExampleState` instance in the `ConcurrencyContainer` returning an outcome will cause the container to transition. The second demonstrates an `AND` behavior where all states in the `ConcurrencyContainer` tied to a specific outcome must have an outcome transition before the `ConcurrencyContainer` returns that specific outcome. Example 4 demonstrates a `BehaviorContainer` where the container invokes the Example 3 behavior. The `BehaviorContainer` can invoke entire behaviors that are defined and written by the FlexBE UI (whether the classic FlexBE App or the new FlexBE WebUI). These examples describe the impact of varying autonomy levels.

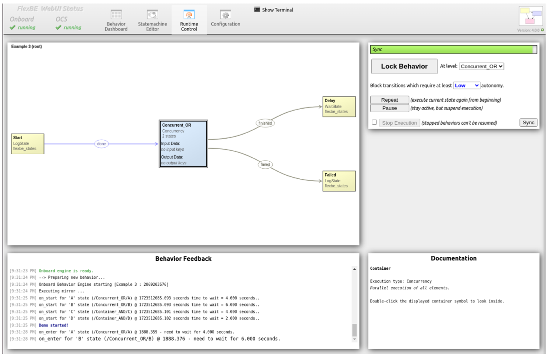
[18] `https://github.com/FlexBE/flexbe_turtlesim_demo`



(a)



(b)



(c)

Figure 3: Example 3: (a) The FlexBE dashboard, (b) Example 3 behavior sub-state machine, (c) Runtime monitoring view of a high level state machine during execution

The main example uses the classic ROS Turtlesim[19] and demonstrates a more complex HFSM that includes an `OperatorDecisionState` that allows operators "collaborative autonomy" with user defined inputs, as well as `StateMachineContainer`, `ConcurrencyContainer`, and `BehaviorContainer`. Figure 1 shows the Turtlesim-based demonstration in action.

These demonstrations will continue to evolve, for the latest see `https://github.com/FlexBE/flexbe_turtlesim_demo` and the official documentation at `https://flexbe.readthedocs.io/en/latest/`.

5 Conclusion

This paper presents the design and development of a new web-based UI for the Flexible Behavior Engine (FlexBE) based on `FastAPI` and `uvicorn` within native Python ROS nodes. Additionally, the paper describes a set of "quick-start" tutorials that allow new users an accessible introduction to FlexBE and hierarchical finite state machine-based control of robotic systems. The tutorials start with simple text-based outputs and then include a simple ROS Turtlesim-based interactive tutorial; this provides an easy to install and start simulation without the complexity of full navigation simulations.

The new FlexBE WebUI is designed to be extensible and future improvements to the UI are planned. These include upgrades to the FlexBE synthesis interface; see [23] for an introduction to synthesis of FlexBE state machines. Further planned enhancements include support for automated layout of the state machine, as well as more intuitive pan and zoom behaviors.

Acknowledgments

This work was supported by US Navy contract N00174-23-1-0018. The views expressed are those of the authors.

References

[1] P. Schillinger, S. Kohlbrecher, and O. von Stryk. "Human-robot collaborative high-level control with application to rescue robotics". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016, pp. 2796–2802. DOI: `10.1109/ICRA.2016.7487442`.

[19] `https://docs.ros.org/en/jazzy/Tutorials/Beginner-CLI-Tools/Introducing-Turtlesim/Introducing-Turtlesim.html`

[2] Stefan Kohlbrecher et al. "A Comprehensive Software Framework for Complex Locomotion and Manipulation Tasks Applicable to Different Types of Humanoid Robots". In: *Frontiers in Robotics and AI* 3.31 (2016). ISSN: 2296-9144. DOI: 10.3389/frobt.2016.00031.

[3] Alberto Romay et al. "Collaborative Autonomy between High-level Behaviors and Human Operators for Remote Manipulation Tasks using Different Humanoid Robots". In: *Journal of Field Robotics* (2016), pp. 333–358. ISSN: 1556-4967. DOI: 10.1002/rob.21671.

[4] Morgan Quigley et al. "ROS: an Open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. Vol. 3. May 2009, p. 5.

[5] Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. Sebastopol, CA: O'Reilly Media, 2015. ISBN: 9781449323882.

[6] Steven Macenski et al. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66 (2022), eabm6074. DOI: 10.1126/scirobotics.abm6074.

[7] Michael Montemerlo and *et al.* "Junior: The Stanford Entry in the Urban Challenge". In: *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 91–123. ISBN: 978-3-642-03991-1. DOI: 10.1007/978-3-642-03991-1_3.

[8] Charles Reinholtz and *et al.* "Odin: Team VictorTango's Entry in the DARPA Urban Challenge". In: *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 125–162. ISBN: 978-3-642-03991-1. DOI: 10.1007/978-3-642-03991-1_4.

[9] Chris Urmson and *et al.* "Autonomous Driving in Urban Environments: Boss and the Urban Challenge". In: *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–59. ISBN: 978-3-642-03991-1. DOI: 10.1007/978-3-642-03991-1_1.

[10] Razan Ghzouli et al. "Behavior Trees and State Machines in Robotics Applications". In: *IEEE Trans. Softw. Eng.* 49.9 (Sept. 2023), pp. 4243–4267. ISSN: 0098-5589. DOI: 10.1109/TSE.2023.3269081.

[11] Michele Colledanchise and Petter Ögren. "How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees". In: *IEEE Transactions on Robotics*. Apr. 2017, pp. 372–389. DOI: 10.1109/TRO.2016.2633567.

[12] Michele Colledanchise and Petter Ogren. *Behavior Trees in Robotics and AI: An Introduction*. July 2018. ISBN: 9781138593732. DOI: 10.1201/9780429489105.

[13] Georg Heppner et al. "Distributed Behavior Trees for Heterogeneous Robot Teams". In: *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. 2023, pp. 1–8. DOI: 10.1109/CASE56687.2023.10260300.

[14] Sebastian Pütz, Jorge Santos Simón, and Joachim Hertzberg. "Move Base Flex A Highly Flexible Navigation Framework for Mobile Robots". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 3416–3421. DOI: 10.1109/IROS.2018.8593829.

[15] Steve Macenski et al. "The Marathon 2: A Navigation System". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 2718–2725. DOI: 10.1109/IROS45743.2020.9341207.

[16] David Harel. "Statecharts: a visual formalism for complex systems". In: *Science of Computer Programming* 8.3 (1987), pp. 231–274. ISSN: 0167-6423. DOI: https://doi.org/10.1016/0167-6423(87)90035-9.

[17] Joshua M. Zutell, David C. Conner, and Philipp Schillinger. "Flexible Behavior Trees: In search of the mythical HFSMBTH for Collaborative Autonomy in Robotics". In: (2022). eprint: 2203.05389 (cs.RO). URL: https://arxiv.org/abs/2203.05389.

[18] Philipp Schillinger. "An Approach for Runtime-Modifiable Behavior Control of Humanoid Rescue Robots". Master's Thesis. Darmstadt, Germany: Technische Universität Darmstadt, 2015. URL: https://www.sim.informatik.tu-darmstadt.de/publ/da/2015_Schillinger_MA.pdf.

[19] David C. Conner and Justin Willis. "Flexible Navigation: Finite State Machine-based Integrated Navigation and Control for ROS Enabled Robots". In: *IEEE SoutheastCon 2017*. Mar. 2017, pp. 1–8. DOI: 10.1109/SECON.2017.7925266.

[20] David C. Conner et al. "Flexible Manipulation: Finite State Machine-based Collaborative Manipulation". In: *SoutheastCon 2018*. 2018, pp. 1–8. DOI: 10.1109/SECON.2018.8478933.

[21] Joshua M. Zutell, David C. Conner, and Philipp Schillinger. "ROS 2-Based Flexible Behavior Engine for Flexible Navigation". In: *SoutheastCon 2022*. 2022, pp. 674–681. DOI: 10.1109/SoutheastCon48659.2022.9764047.

[22] Soroosh Tayebi Arasteh and Adam Kalisz. "Conversion Between Cubic Bezier Curves and Catmull--Rom Splines". In: *SN Computer Science* 2.5 (July 2021), p. 398. DOI: 10.1007/s42979-021-00770-x.

[23] Josh Luzier and David C. Conner. "Solving the Farmer's Dilemma with FlexBE Using GR(1) Synthesis". In: *SoutheastCon 2024*. 2024, pp. 1189–1196. DOI: 10.1109/SoutheastCon52093.2024.10500189.

Stigma: A Tool for Modifying Closed-Source Android Applications*

Ed Novak[1], Shaamyl Anwar[2], Saad Mahboob[3]
Shokhinabonu Tojieva[4], and Chelsea Rao[5]

Computer Science Department
Franklin and Marshall College
Lancaster, PA 17604

[1]enovak@fandm.edu, [2]mshaamylanwar@gmail.com, [3]saadmahboob3@gmail.com
[4]shokhinatojieva@gmail.com, [5]crao@fandm.edu

Abstract

A difficult but potentially powerful advanced software engineering concept is to modify existing, compiled, closed-source applications to identify and potentially remedy security and privacy issues. This technically challenging concept is very applicable to the Android ecosystem, but existing approaches are bespoke, use-case specific implementations. In this paper we present Stigma, an open-source software tool which can make modifications to commodity Android applications. Our tool allows researchers and skilled users to define their own desired modifications for a range of purposes such as security and privacy analysis, improving app functionality, removing unwanted features, debugging, profiling, and others. We evaluate Stigma in terms of compatibility, efficacy, and efficiency on approximately 100 commodity Android applications.

1 Introduction

Android smartphone applications are most commonly pre-compiled and closed-source. This makes their functionality rigid and somewhat opaque, leading to security concerns and contributing negatively to the general trend of concerns about user privacy. We seek to provide tools that modify and analyze the behavior of these closed-source applications. Such tools could one day be used to perform a variety of tasks such as identifying app uses of sensitive information, searching for security vulnerabilities, fixing bugs, and even improving app functionality independently of the original developer.

Unfortunately modifying commodity Android apps is generally only done in very limited ways via custom fit, temporary, and largely manual processes. Although reverse engineering and "cracking" / "modding" exists in a limited sense in the Android community, these efforts are largely disconnected and lack big picture strategies. Many of the existing research projects in this area [16, 11, 17] do not seem to be available, and therefore cannot be feasibly replicated, embraced, or extended. Furthermore, these legacy projects have usability and compatibility limitations [4].

Modifying apps is difficult due to the fact that the source code is not available. Instead users and tools must operate directly on the byte-code (machine code for a virtual machine). Writing byte-code directly is difficult due to numerous esoteric constraints, non-obvious syntax rules, and few conveniences most programmers take for granted (e.g., for loops). Additionally, the Android framework, runtime environment, and build system are complex leading to bespoke techniques and tools.

In this paper we present Stigma [13]; an open source (GPLv3), command line, python program which can help users make modifications to Android applications for security and privacy analysis. Users first obtain the Android Package (APK) of an app, which is used as input. Plugins, written in python for Stigma, determine what alterations will be made to the app. A new, modified APK is output, which can be run on any target device that the original APK was compatible with. No modifications are necessary to the device or OS itself making for easier reproducibility and long term compatibility. The contributions of this paper are as follows:

- We present the design and prototype implementation of Stigma, our open source and extensible software tool for modifying Android apps.
- As an exemplary use of Stigma, we implement (and also distribute in open source) two Stigma plugins. The first is a dynamic information flow tracking (DIFT) plugin and the second is a "SharedPreferences" extraction plugin.
- We highlight the esoteric details of the "reference pools" in the DEX file format, which to the best of our knowledge, have not been documented extensively elsewhere.

- We evaluate Stigma and our two plugins on approximately 100 popular Android applications. We seek to measure the compatibility of Stigma with arbitrary Android applications as well as the overhead incurred on the application.

2 Related Work

The most closely related works are “Dr. Android and Mr. Hide” [10], and “SIF” [9] in which the smali assembly code of Android apps is modified directly and automatically. “Dr. Android” makes limited modifications in the narrow scope of implementing a more fine grained permission system. SIF asks the user to specify their desired modification in a language called “SIFScript.” The SIFScript includes the functionality itself as well as the general places and times in which the functionality should be inserted (called the “workload”). These works were published in 2012 and 2013 respectively. Due to their age, it is very likely that they are no longer compatible with modern Android. They seem to be orphaned and don’t appear to be readily available online.

2.1 DIFT Systems

One relevant subfield is that of dynamic information flow tracking (DIFT), in which code is added into the target app to track and alert the user about the use of their own sensitive and/or personal, identifiable information (PII). This is the idea implemented by our Stigma plugin described in Section 4. Some of the most relevant works in this area include ViaLin [1], TaintMan [16], TaintArt [15], TaintDroid [8], AppCaulk [11], ConDySTA [18], and Capper [17].

2.2 Community Projects

Some less formal, community based efforts include the Cydia Substrate for Android [5], the (apparently defunct) Xposed Framework, and Android DDI [7] which allows the user to write their modifications using the Java Native Interface (JNI). None of these projects seem to have accompanying publications in peer reviewed conferences or journals.

2.3 Limitations of the Related Work

The existing works in this area all suffer from at least one of two critical flaws. Either the software described was never released to the public, or the system design has significant compatibility and usability concerns. Some projects exhibit both problems. Compatibility and usability concerns include requiring substantial changes to the Android OS, the Android Framework, the dex2oat compiler, rooting, or changing other aspects of the platform / device itself instead of the app. These approaches are difficult for others to setup, brittle, and become outdated quickly as the Android OS is continually updated. In contrast, our system is carefully designed to follow the semantics of smali / dex byte-code itself, which is a standard that hasn't changed substantially since the introduction of Android.

3 Stigma System Design

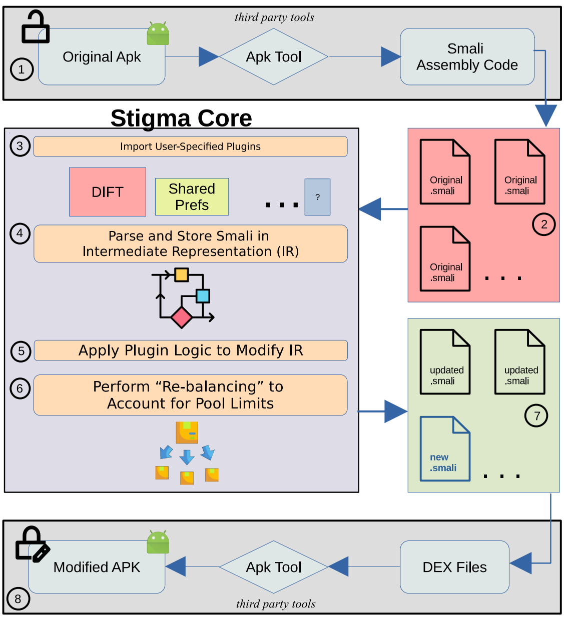


Figure 1: Stigma system architecture.

Stigma [13] is a fully open-source python program that accepts an APK file as input, and outputs a modified version of that APK file. The modifications made are specified by plug-ins, which are written by the user. As a proof of concept, Stigma comes with two pre-written plugins. One performs DIFT, which seeks to track the use of location (GPS coordinate) data. The other inserts code such that the app prints the keys and values of the default SharedPreferences database when the app is launched. Further information about these two plugins is given in Sec. 4 and 5. Other plugins can be imagined and implemented that allow researchers and other power users to specify precisely what modifications should be made to the app.

An overview of the architecture of the system can be seen in Fig. 1. First ① a third party tool `apktool`[3], is used to extract the Dalvik byte-code (DEX[1]) from the application and convert it to the assembly-like `smali` [12,

[1] DEX is designed to be run on a Java Virtual Machine (JVM), but in the modern Android ecosystem, it probably never will be. Instead, the DEX2OAT compiler is invoked at install

2] language ②. Stigma then parses these smali files into an intermediate representation (IR) ④. Stigma maintains in-memory representations (objects) for smali classes, smali methods, registers, and basic data-types (32-bit, 64-bit, and object references). Plugin logic is applied to the IR. Then in step ⑥, the code must be "re-balanced" to account for the constant pool limits as described in Section 3.1. Finally, the modified IR is written back to smali files on disk. The modified smali files ⑦, along with any new smali classes added by the plugin(s), are re-packed using the same third party tool `apktool` ⑧. The end result is an APK, digitally signed by Stigma, which can be installed on any device for which the original, input APK was compatible.

Stigma parses and allows the user to modify the smali assembly code of the target app. The original Java or Kotlin source code is not available, due to most apps being distributed close-sourced. And the immediately available DEX byte-code is not human readable, making it near impossible for users to define plugins for DEX directly.

As mentioned previously, the smali classes, methods, instructions, registers, and types of the original app are all represented in-memory by python objects. Stigma also builds a control flow graph for every method in the app, and does type analysis such that it can determine the known type of every data value stored in every register at any point in the execution. Of course, there are many points where the type information is unknown or undefined. For example, at the very beginning of a method most of the temporary, general purpose registers are empty and therefore have no type.

3.1 Reference Pools

As mentioned very briefly in the official Dalvik documentation "There are separately enumerated and indexed constant pools for references to strings, types, fields, and methods." [6]. This means that in a single DEX file (comprised of many smali files) all *references* to (1) strings, (2) types, (3) class fields, and (4) methods are collected into respective sets. Each set may contain at most 65,535 entries since the pools are enumerated using an unsigned short.

As applications have grown larger and larger, it has become increasingly common for a single Android application to exceed the 65k limit on one or more of the pools. To alleviate this the code is distributed into multiple DEX files (`classes.dex`, `classes2.dex`, `classes3.dex`, etc.) such that none of the pools are overloaded. Normally this is done by the `dx` converter (from the Android SDK) which converts Java `.class` files to Android `.dex` files. But, Stigma modifies the code *after it has been converted to DEX*, and so any

time to convert the DEX code to machine code matching the architecture of the device. The app is then run on that device via the Android RunTime (ART).

```
1  .class public Lcom/example/stigmatestapp/MainActivity;
2  .super Landroidx/appcompat/app/AppCompatActivity;
3  .source "MainActivity.java"
4
5  # static fields
6  .field static final GREEN_TRANSPARENT:I = 0x6600ff00
7
8  # instance fields
9  .field sputgetText:Landroid/widget/TextView;
10
11 # direct methods
12 .method public constructor <init>()V
13     .locals 0
14
15     .line 19
16     invoke-direct {p0}, Landroidx/appcompat/app/
           AppCompatActivity;-><init>()V
17
18     return-void
19 .end method
```

Listing 1: Sample code used to demonstrate how the constant reference pools are tabulated. This listing contains 9 string references, 5 type references, 2 field references, and 2 method references.

changes it makes that alter the amount of items in the pools may cause those pools to overflow and for the app to fail to compile / run.

Properly organizing a collection of smali files into an appropriate number of DEX files is not straightforward, since it is not clear how various smali instructions impact the four pools. To our knowledge, this relationship is not explained in any pre-existing documentation. The `smali` command line tool [12] can be used to create a DEX file from a single smali file. The resulting DEX file will likely be unable to run, since it does not even contain some of the necessary foundation code such as the Android support libraries. But, such a DEX file can be analyzed by the `dexdump` tool, from the Android SDK, to precisely determine how the code therein contributes to each pool. Using this tool-chain we are able to reverse engineer the relationship between smali code and the four pools.

Consider the code sample in Listing 1. According to `dexdump`, the DEX file containing *only this class* contains 9 string references, 5 class/type references, 2 field references, and 2 method references.

- **Strings** (9 total) - The class name and parent class name on lines 1 and 2 account for two strings. And, line 3 contains a literal string. These strings are used, presumably, for debugging purposes such as stack traces, and compiler warnings as well as for Java reflection. Each of the fields declared on lines 6 and 9 contribute two string references each (one for the identifier, and another for the type). Finally, the method declaration on line 12 contributes two string references, (one for the method name, and the other for the method return type).

- **Types** (5 total) - The class and parent class on lines 1 and 2 are types. The `I` in the field declared on line 6 is a type (integer). The `TextView` referenced on line 9 is a type. And, the `V` indicating that the `<init>` method returns `void`, is a type. Note that the reference to `AppCompatActivity` on line 16 is not counted, because it is redundant with the reference on line 2.
- **Fields** (2 total) - This class references only two fields (as declarations) on lines 6 and 9.
- **Methods** (2 total) - This class references only two methods. The declaration of `MainActivity.<init>` on line 12 and the call to `AppCompatActivity.<init>` on line 16.

Stigma uses this logic to distribute smali files into a number of `classesX.dex` files appropriately. It is important to note that smali instructions need no modifications or special access rights in order to reference the classes, fields, and methods of smali files in other DEX files. Access rights are restricted only by the traditional Java access modifiers `public`, `private`, and `protected`.

3.2 Extensible Plugin Framework

Without any active plugins, Stigma will not make any changes to the target app. How should intended changes be specified? In our system, plugins specify callback handler functions. The callback handler function itself is written by the plugin author / user, specifying which key points it should be invoked on. These functions return new smali code that is inserted into the app at key point(s).

The handlers are invoked as the original application code is linearly iterated over. They can be called or triggered at various key points in the target app. First they can be called at the point in the app at which the app is launching. This is similar to "the start of the main()" in a traditional program. Second they can be called at the start of each original smali method. Finally, the most intricate trigger is applied individually to each of the 200+ types of smali instructions.

Stigma provides an "Instrumenter" class, which has a method `sign_up()`. Plugins can call the `sign_up()` method in order to register callbacks as shown in Fig. 2.

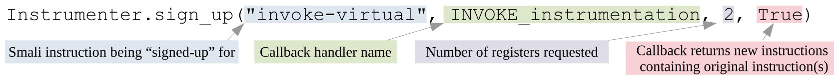


Figure 2: Example of a call to `sign_up()` made by a plugin registering a callback handler for the `invoke-virtual` smali instruction.

3.3 Other Implementation Challenges

Although smali assembly is the most user friendly form of the code to work with, there are still several technical details that must be accounted for when writing or modifying smali code. Stigma accounts for many of them automatically. Specifically, extracting the code from an APK file and converting it to smali, allocating and identifying machine registers that are free to use by the plugin code, accounting for code offset value limits, avoiding unintentional changes to control flow, correctly allocating "reference pools" among DEX files (as discussed in Sec. 3.1), and re-packing the modified smali code back into a usable APK with a valid cryptographic signatures.

Many of these esoteric and complex details of the smali and DEX languages are not well documented. Interested readers can see our technical report [14].

4 DIFT Plugin

As a proof of concept, we design a plugin for Stigma that implements dynamic information flow tracking (DIFT) of sensitive user information. Our plugin registers a variety of handlers for many smali instructions to (1) originate, (2) propagate, and (3) terminate tags that mark sensitive data. It also registers a handler for the start of each method to propagate tag values from the function parameters / inputs. The plugin essentially specifies new smali instructions, which Stigma inserts amongst the original app instructions, to implement the logic of sensitive information marking and tracking. Our plugin has several limitations (it is only a prototype), which are given in our technical report [<**empty citation**>].

4.1 Tag Origination

For tag origination, Stigma identifies several key functions from the Android API that can be used to obtain sensitive data. For example, `LocationManager.getLastKnownLocation(String provider)`, which is one method used to obtain the device's GPS coordinates. When this instruction/method call is identified in the smali assembly code, our Stigma plugin interleaves instructions to store a tag value on the register used to store the return value.

4.2 Tag Propagation

When a tag is applied to a register, that tag value should flow as the data in that register flows. For example, if the data is copied to another register or passed to a function, the tag should also flow. Our DIFT plugin interleaves new smali instructions to move the tag values, triggered by roughly 85 of the almost 250 smali instructions that move data.

4.2.1 Propagating Tags Across Function Calls

Well written Java code makes heavy use of methods. In order to track sensitive information in and out of method calls, we split all methods into two categories: "internal" and "external". Internal methods are all those defined in the smali code contained in the target APK file. External methods are those for which their smali source code is unattainable. For example, `java/lang/StringBuilder` is provided by the runtime so the code is not included in the APK.

For internal method calls, the tags for the arguments (at the call site) need to be propagated to the parameters (at the definition / callee site). At the call site, new instructions are added just before the function call. The tags for the method arguments are read (e.g., `public static CallingClass_foo_v1_TAG:F`) and then written into the tag locations for the method parameters at the callee site (e.g., `public static CalleeClass_bar_p0_TAG:F`).

When a method returns (keeping in mind there may be more than one return point) the returned value may contain sensitive information. Therefore, for every `return` instruction, the tag value of the returned register is copied into the special global tag field `public static return_field_TAG:F`. At the call site, the tag value is extracted from that field and propagated to the specified destination register.

4.3 Tag Termination

For (3) tag termination, Stigma identifies certain functions which indicate data transmission. In the current implementation this is limited to the various `write()` methods of `java/io/OutputStreamWriter;` and `java/io/OutputStream;`. These are used in network socket I/O and file I/O. When such a method is identified, our plugin writes new instructions into the application which retrieves the tag value associated with each of the input parameters. If the tag value of any parameter is not zero, an entry is written to the Android system log (logcat) alerting the user that sensitive information is being leaked.

5 SharedPreferences Extraction Plugin

"SharedPreferences" is an often used API in the Android framework. It allows app developers to store key-value pair information. Traditionally, it is intended to store the user's innocuous preferences that are specific to an app (e.g., repeat or shuffle in a music player app). Occasionally, developers store things that they should not, introducing security risks and vulnerabilities. Examples include plaintext passwords, private information, booleans to control paid only features, and encryption keys.

We wrote a plugin for Stigma that forces the app to print the entire contents of the default SharedPreferences database when the app is launched. This allows the user to search for suspicious or obviously improper use. Although there are other methods of obtaining the SharedPreferences contents, ours does not require rooting or modifying the OS / device. Additionally, our approach operates during runtime. Which is important, since applications that have not been run in a realistic way likely will not have added any actual values to the SharedPreferences database.

5.1 Implementation Details

Our plugin adds roughly 50 smali assembly instructions to application that invokes the Android SharedPreferences API. Since our code is inserted into the app itself, it runs with same privileges that app has. We simply iterate over the returned HashMap and print the values using the built-in Android logging system (logcat).

Finding the starting point of the application is not straightforward since Android applications follow an event driven architecture and there is no traditional `main()` function. To find the starting point of the application, Stigma parses the associated `AndroidManfiest.xml` file for `activity` and `activity alias` instances that specify the "`LAUNCHER`" attribute. The new code is then inserted into those activities at the start of the `onCreate()` method.

6 Evaluation & Case Studies

To evaluate the compatibility across many commodity Android applications, we acquired 100 random popular applications from `https://APKMirror.com` and ran Stigma on them.

First, we selected approximately 31 applications and, for each, we processed it with Stigma using our prototype DIFT plugin. If successful we installed and ran that app on an Android device. We found that approximately 45% of the apps we tested (14/31) had some sort of compatibility problem with the 3rd party depedency `apktool`, making it impossible to fully evaluate Stigma on that app. Of the remaining 17 apps, only 11.76% of them (2/17) had compatibility problems with Stigma itself. Of those 15/31 apps that appeared to be fully compatible, Stigma was able to identify and track the use of GPS location information in 6 apps.

Similarly, for the remaining 67 random applications we processed each of them using Stigma with our prototype SharedPreferences plugin. We found that approximately 26% of the apps (18/67) had some sort of compatibility problem with `apktool`. From the remaining 49 apps, we were able to extract SharedPreferences data from 59% of them (29/49).

Stigma is not able to obtain GPS data or SharedPreferences data from every app. In the vast majority of cases where it was not, the reason is simply because the app in question doesn't appear to utilize location data or the SharedPreferences API at all.

6.1 LOC Overhead

To measure the overhead of the new instructions added by Stigma, we compare the number of lines of code in an application before and after Stigma is run with the DIFT plugin (which adds many more lines of code than our other prototype plugin). We compiled a short list of 5 "case study" applications. Three applications were selected from a list of popular Android applications. One (Open Chaos Chess) was selected from the open-source FDroid application market. And the final application is a simple, self made application used for testing and development of Stigma. All five are listed below. The LOC analysis is shown in Fig 3a.

1. "**Weather**" weather forecasts - `com.macropinch.swan`
   version: 5.1.7

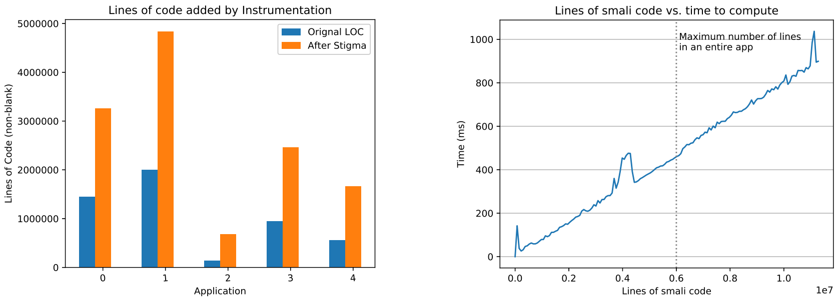
2. "**GroupMe**" messaging application - `com.groupme.android`
   version: 5.54.4

3. "**Open Chaos Chess**" chess game - `dev.corruptedark.openchaoschess`
   version: 1.7.0

4. "**Office Documents Viewer**" office suite
   `de.joergjahnke.documentviewer.android.free`
   version: 1.29.13

5. "**Stigma Test App**" internal test application -
   version 0.1

The lines of code, both before and after modification, are measured as any non-blank lines of *smali assembly code*. This includes comments, function



(a) Lines of code added by Stigma. (b) Time incurred by smali instructions.

signatures, field declarations, etc. in addition to actual opcodes / instructions. Therefore, some lines of code added don't incur much, if any, computational overhead. As a byproduct of the design of Stigma, many class fields are added to the application, which forms the bulk of the new lines of code shown in this analysis. As is shown in Fig. 3a our implementation increases the lines of code by about a factor of 2.5x.

6.2 Memory Overhead

We also measure the memory usage of applications on launch. For each application, we launched and performed basic functionality (logging in, starting a game, etc.) The `top` command on the Android command line (`adb shell`) is used to examine the memory usage. This was done before any instrumentation and again after. Results are shown in Table 1. `VIRT` represents the total size of the virtual address space for that process. `RES` represents the actual physical memory in use ("**RES**erved") for that process. Although there is some memory overhead, it is relatively small. This is likely due to the fact that in-memory data structures and code are insignificant compared with common assets such as audio, and images that already exist in most apps and which Stigma add none.

6.3 CPU Overhead

A significant concern of Stigma is overhead it might incur in responsiveness and performance of the application. Empirically, this impact is imperceptible. The computational complexity of the original app code is generally not changed, since our prototype plugins do not introduce any loops, recursion, or new function calls.

Process	**VIRT** before/after	**RES** before/after
Weather	4.3 / 4.4G	130M / 173M
Weather “remote”	4.1G / 4.1G	32M / 38M
Groupme	4.2G / 4.3G	111M / 143M
Groupme “sync”	4.1G / 4.2G	56M / 69M
Open Chaos Chess	4.2G / 4.2G	135M / 135M
Document Viewer	4.4 / 4.4G	139 / 151M
Stigma Test App	4.1G / 4.1G	62M / 67M

Table 1: Memory overhead before and after instrumentation.

To estimate the CPU overhead more precisely we wrote a simple Android application, which executes arbitrary assembly code similar to the code that might be inserted by Stigma to perform DIFT. We executed these instructions repeatedly in larger and larger batches, each time measuring the amount of time it took to complete the batch. The experiment was carried out on a Nexus 5x, which is a modest device released in 2015 running a Hexa-core CPU: (4x1.4 GHz Cortex-A53 & 2x1.8 GHz Cortex-A57). The results can be seen in Fig.3b.

Generally, smali instructions incur a negligable amount of overhead. For an app of reasonable size of 2 million lines of (smali) code, *the entire codebase* could be executed in only ~2/10 of a second. Usually, Android app performance is not CPU-bound, but rather I/O bound.

7 Conclusion

Modifying the code of commodity Android applications is a promising, foundational technique. Unfortunately, most of the tools and systems created in this area have a singular purpose and many are never released. Recent published works from the literature usually don't reveal the critical details necessary to safely modify smali code. In this work we uncover critical concepts for modifying smali code, highlight pitfalls that researchers will experience, and offer an open-source prototype implementation called Stigma. Our work aims to support and inspire future research efforts in this area.

References

[1] Khaled Ahmed et al. “ViaLin: Path-Aware Dynamic Taint Analysis for Android”. In: *Proceedings of the 31st ACM Joint European Software En-*

gineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE 2023. <conf-loc>, <city>San Francisco</city>, <state>CA</state>, <country>USA</country>, </conf-loc>: Association for Computing Machinery, 2023, pp. 1598–1610. ISBN: 9798400703270. DOI: 10.1145/3611643.3616330. URL: https://doi.org/10.1145/3611643.3616330.

[2] *Android Instrumentation with Smali: A survival guide*. Available At: http://paulsec.github.io/posts/android-smali-primer/. 2020.

[3] *APKtool Project Source Code*. Available At: https://ibotpeaches.github.io/Apktool/. 2022.

[4] Fabian Berner. and Johannes Sametinger. "Dynamic Taint-tracking: Directions for Future Research". In: *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications - Volume 2: SECRYPT,* INSTICC. SciTePress, 2019, pp. 294–305. ISBN: 978-989-758-378-0. DOI: 10.5220/0008118502940305.

[5] *Cydia Substrate for Android*. Available At: http://www.cydiasubstrate.com/. 2014.

[6] *Dalvik Byte-Code Documentation*. Available At: https://source.android.com/devices/tech/dalvik/dalvik-bytecode. 2022.

[7] *DDI, Dynamic Dalvik Instrumentation Toolkit*. Available At: https://github.com/crmulliner/ddi. 2022.

[8] William Enck et al. "TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones". In: *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*. OSDI'10. Vancouver, BC, Canada: USENIX Association, 2010, pp. 393–407. URL: http://dl.acm.org/citation.cfm?id=1924943.1924971.

[9] Shuai Hao et al. "SIF: A Selective Instrumentation Framework for Mobile Applications". In: *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys '13. Taipei, Taiwan: Association for Computing Machinery, 2013, pp. 167–180. ISBN: 9781450316729. DOI: 10.1145/2462456.2465430. URL: https://doi.org/10.1145/2462456.2465430.

[10] Jinseong Jeon et al. "Dr. Android and Mr. Hide: Fine-Grained Permissions in Android Applications". In: *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*. SPSM '12. Raleigh, North Carolina, USA: Association for Computing Machinery, 2012, pp. 3–14. ISBN: 9781450316668. DOI: 10.1145/2381934.2381938. URL: https://doi.org/10.1145/2381934.2381938.

[11] J. Schutte, D. Titze, and J. M. de Fuentes. “AppCaulk: Data Leak Prevention by Injecting Targeted Taint Tracking into Android Apps”. In: *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. 2014, pp. 370–379. DOI: 10.1109/TrustCom.2014.48.

[12] *Smali Project Source Code*. Available At: https://github.com/JesusFreke/smali. 2022.

[13] *Stigma Source Code*. Available At: https://github.com/fmresearchnovak/stigma. 2022.

[14] *Stigma Technical Report*. Available At: http://ednovak.net/documents/stigma_tr.pdf. 2024.

[15] Mingshen Sun, Tao Wei, and John C.S. Lui. “TaintART: A Practical Multi-level Information-Flow Tracking System for Android RunTime”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’16. Vienna, Austria: ACM, 2016, pp. 331–342. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978343. URL: http://doi.acm.org/10.1145/2976749.2978343.

[16] W. You et al. “TaintMan: An ART-Compatible Dynamic Taint Analysis Framework on Unmodified and Non-Rooted Android Devices”. In: *IEEE Transactions on Dependable and Secure Computing* 17.1 (Jan. 2020), pp. 209–222. ISSN: 1941-0018. DOI: 10.1109/TDSC.2017.2740169.

[17] Mu Zhang and Heng Yin. “Efficient, Context-Aware Privacy Leakage Confinement for Android Applications without Firmware Modding”. In: *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*. ASIA CCS ’14. Kyoto, Japan: Association for Computing Machinery, 2014, pp. 259–270. ISBN: 9781450328005. DOI: 10.1145/2590296.2590312. URL: https://doi.org/10.1145/2590296.2590312.

[18] Xueling Zhang et al. “ConDySTA: Context-Aware Dynamic Supplement to Static Taint Analysis”. In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 796–812. DOI: 10.1109/SP40001.2021.00040.

Addressing the Gap Between How Students and Professionals Read Code*

Matthew Woerner, David Socha, Mark Kochanski
Computing & Software Systems
School of Science, Technology, Engineering & Math
University of Washington Bothell
Bothell, WA 98011
{woerner,socha,markk}@uw.edu

Abstract

This paper reports on the design and evaluation of a Canvas module to help students learn how to do what professionals commonly have to do: manually read and understand code by tracing calls into and through multiple files in novel codebases in order to understand how the code actually works before modifying the code. This work was motivated by our prior research that showed that students, but not professionals, exhibited a common set of mistakes when reading novel codebases. Our goal in this current study was to see if we could create a module that would help students move toward professional practice in this respect. We based the module design upon a set of lessons derived from our analysis of the videos of ten students doing code reading exercises. Our evaluation of the results of the nine students who went through the Canvas module suggest that the module indeed improved their ability to read novel codebases. The module is available online for anyone to use.

1 Introduction

In a prior paper [7] we uncovered three particular ways that undergraduate computer science students and professional software developers differ in the way that they went about understanding novel codebases. We identified three anti-patterns exhibited almost exclusively by students but not professionals:

1. Making and not self-correcting misinterpretations of the code.
2. Not following the call stack through files.
3. Examining files not critical to the current execution and/or searching randomly through files.

Those results led us to design a Canvas module with a scaffolded series of lessons to teach students a structured code reading strategy so as to help students grow closer to a professional level of code reading ability. This paper reports on the design and evaluation of that Canvas module.

Section 2 briefly discusses the previous work and research. Section 3 describes and provides some of the design rationales behind the Canvas module we created. Section 4 describes the three codebases we created to teach and assess student's knowledge and ability about how to follow the trace of code through a codebase. Section 5 describes the Canvas module we created. Section 6 describes how we assessed the module's efficacy and what the assessment revealed. Section 7 concludes by discussing possible future avenues of research and practice, including ways that others could use the open-source Canvas module we created.

2 Background

As described in our prior paper [7], this work emerged from a Software Engineering Studio course that the second and third author had co-designed to provide an "on-campus internship-like experience" for students before moving into the workplace proper. That course was organized in a sequence of badges that onboarded the students into teams using a common set of development tools and an execution environment which they used to write, test, and execute code, and then had them do progressively more complex and less structured collaborative development of new features in a multi-year codebase. Each badge consisted of: instructions and resources to read and refer to; a low-stakes self-assessment quiz to ensure students understood key terminology; a body of work for the student to complete; a portfolio that the student constructs in order to visually show and describe the work they did for this badge; a 30-minute badge challenge session during which the instructor met via Zoom with each student to assess the student's badge work, uncover gaps in the student's knowledge, and help fill in those gaps; and a final summative challenge reflection the student writes to reflect upon their experience in this badge.

The badge challenges were done as semi-structured interviews [2] during which we faculty based our questions off what we saw in the student's work, ask questions to uncover gaps in the student's knowledge, and then help the student bridge that gap. We typically started each badge with one of a few common questions, but each conversation was unique in the details. The format allowed us to meet the student where they were at and revealed many surprises.

One surprise was that students did not know how to read novel codebases. As noted in our prior paper [7], only a handful of the over 100 students were able to successfully follow a call stack to comprehend what source code in a novel codebase actually did. Students tended to require direct instruction and guidance to help them fully understand the function of the codebase. The first author noticed these same dynamics during the semi-structured interviews of 10 other students from different universities, but did not observe them in the 11 semi-structured interviews with professional software developers [7].

The existing literature suggested that experts are far more efficient in how they read code [3, 4]. This could in part be due to more familiarity with how codebases are typically structured, which could lead to less cognitive overload and a faster and more comprehensive reading of the code. Some eye tracking studies suggest that professionals engage in more complex processing of code than novices [5]. These results hold a strong correlation to our interview findings in which students spent a longer amount of time reading code and did not understand the code as well as the professionals given the same tasks [7].

None of the professionals had taken a course that taught them how to read novel codebases. They all had learned this on the job. Nor could we find anyone who had developed materials to teach students how to navigate code call stacks across multiple files. This motivated us to develop a course module that would help students learn how to read code like professionals, which is what this current paper covers.

3 Making sense of what we saw

Before starting to design the module, we wanted to be clear on what lessons we wanted to teach, and we wanted those lessons to be grounded in observations of actual issues students encountered when trying to read codebases. We wanted to use backward design [6] to create a learning module and its codebases, exercises, and final assessment. In particular, we wanted to help students avoid the three anti-patterns we had observed in our prior study [7]: examining files not critical to the current execution, not following call stacks, and making and not correcting misinterpretations of code.

We started by reanalyzing the videos of the ten students from our prior study. This revealed sixteen key observations of problems they exhibited related to reading code, as shown in the left-most column of Figure 1. Grouping these by affinity produced seven issues (column 2) each supported by multiple observations. As shown in the vertical scope of the cells for O6 and O13 in Figure 1, each of those two observations supported two separate issues.

Further thinking about those issues led to three key hypotheses or interpretations (column 3) about the underlying causes of the students' difficulty reading code:

H1: Students appear to have trouble navigating through larger codebases, perhaps because they are unfamiliar with the format or how to find what they are looking for.

H2: Students end up missing and glossing over important details in code leading to misinterpretations, perhaps because they don't examine code with a critical mindset.

H3: Students appear to misunderstand important segments of the code they read, perhaps because they are unfamiliar with some computational basics.

Using all of that information, we came up with a list of six key concepts (column 4) that we conjectured students did not understand sufficiently well and that might relate to how well one performs in a code reading exercise. These concepts motivated a set of twelve lessons (column 5) that we could teach in the module. We determined that addressing hypothesis H3, that students had a knowledge gap preventing them from understanding the code, would be out of scope for our module as it touches more on students prior computer science education rather than teaching them new skills to aid in the process of code reading. This left us with nine lessons to fulfill in the learning module, as shown in the shaded cells in Figure 1.

4 Formative and summative codebases

Knowing the set of lessons to teach, the first author designed and created a set of three new C# codebases to use for teaching and assessing students on those nine lessons: a Note Taking app, a Meal Planner app, and a Social Media app. To allow us to compare the results of the learning module to the results from our prior study, we designed the codebases to be like those used in our prior study, in that each codebase provided an application programming interface (API) for a simple application with multiple files spread across multiple directories with no documentation to assist in the students reading. The codebases were written in C#, but the lesson and codebase designs are applicable to any object-oriented language. The code for these applications is available online,

Observations	Issues	Hypotheses	Concepts	Lessons
O1: Students not using namespaces to find source code.	I1: Students having difficulty navigating to and tracking code into different files	H1: Students appear to have trouble navigating through larger code bases, perhaps because they are unfamiliar with the format or how to find what they are looking for	C1: Navigating a larger codebase	L1: Highlight that namespaces can mimic folder structure of the codebase
O2: Students randomly clicking on files to find source code.				
O3: Students not examining subdirectories for source code.				L2: Illustrate the relationship between class names and filenames to students
O4: Students not using IDE tools to help themselves navigate.	I2: Students have difficulty understanding how codebases are structured		C2: Understanding codebase structure	L3: Showcase how to distinguish between library and written functions
O5: Students not understanding how unit tests are structured in the project directory and meant to be navigated.				
O6: Students not stepping down more than one level deep in a call stack.				L4: Provide an example of an IDE tool that automates the process
O7: Students do not think about what the name of a method might imply about its function.	I4: Students do not examine code critically and tend to assume the code is 100% correct	H2: Students end up missing and glossing over important details in code leading to misinterpretations, perhaps because they don't examine code with a critical mindset	C3: Code misinterpretations	L5: Provide examples of cases where taking a method at face value can lead to a massive misinterpretation of the code
O8: Students do not think deeply about the connections of methods and their implications related to data structures (i.e. first element in a list).				L6: Teach avoiding misinterpretations by relating code to the application (does what I'm seeing make sense in context)
O9: Students do not think deeply about conditional return values and what potential results from a called method could be (i.e. null).				
O10: Students do not step through and explain every statement of code encountered, but instead try to make assumptions and gloss over the code.	I3: Students making assumptions about code rather than reading it		C4: Reading code critically and drilling down into problems	L7: Build on prior examples of distinguishing between library and user written functions by showcasing searching for online documentation
O11: Students assume the name of a method completely describes its functionality without looking deeper.				
O12: Students do not look up unfamiliar outside library functions for their purpose.	I5: Students do not typically search for documentation on unfamiliar patterns in code			L8: Emphasize the importance of following the call stack step by step and retaining info from previous levels of stack
O13: Students do not look up language syntax that is unfamiliar to them.				L9: Work through reading and commenting undocumented code
O14: Students read code from left to right (order of assignment), not right to left (order of execution).	I6: Students lack the background knowledge to read some codebases	H3: Students appear to miss important details about the code they read, perhaps because they are unfamiliar with some computational basics	C5: Lack of knowledge	L10: Provide examples of effective knowledge searching strategies
O15: Students do not connect files to disk storage and variables to memory.	I7: Students have difficulty understanding hardware constraints on code		C6: Hardware constraints	L11: Showcase scalable and unscalable code
O16: Students do not connect code to performance and scalability.				L12: Provide examples of issues that could lead to scalability concerns

Figure 1: Shows how our analysis of 16 observations from our video analysis of students doing the code reading exercise led to 7 issues, 3 hypotheses or interpretations, 6 concepts, and 12 lessons to consider teaching. The final module addressed the nine shaded lessons, L1 through L9.

and additional details about these codebases can be found in the first author's master's thesis [8].

The module uses the Note Taking app's codebase[1] to instruct students in the basics of how to trace through a codebase. It consists of 6 files and roughly 222 total lines of source code. This module was designed to get students used to exploring files that do not contain source code as a part of their tracing. Within the critical code path students would be required to examine a JSON file read by the program to determine the proper flow of execution.

The module uses the Meal Planner app's codebase[2] to introduce students to the process of looking up unfamiliar functions and navigating through a codebase with dozens of files. It contains 51 files and 828 total lines of source code, which is deceptively large due to 44 of the files defining small classes with roughly 10 lines of code each. The codebase includes various code reading pitfalls for students such as method overloads, similarly named methods, and a liberal use of outside functions that students may not have been previously exposed to. These gave students ample opportunities to demonstrate the anti-patterns that the module was designed to help correct.

The module uses the Social Media app's codebase[3] for the final knowledge check. It consists of 18 files containing 734 lines of source code, resulting in the largest average lines of code per file of the three codebases, and has a higher density of code reading pitfalls. To make it easier to track the progress students made in avoiding anti-patterns, we designed the codebase to be easy to misinterpret for readers who only "skimmed" through the code.

To help us understand what lines of code each student viewed when tracing through the files, and how their understanding of the code changed during their tracing, we inserted into each codebase a set of comment blocks, empty except for a unique identifier. Each student was instructed to read though the codebase and for every blank comment block they come across, describe what they think the section after the comment block is doing and add it to a list of comments that they had already filled out to show the order in which they had done so. Figure 2 shows a portion of the Social Media app's commented code and a snippet of one student's submission. Students were instructed to not erase any comments that they felt they may have filled in erroneously; instead students were asked to remark within the comment that the prior comment was made in error and give context as to why it was filled out in the first place and how the student realized their mistake. Students then submitted their answers in the form of an ordered list of their comments from the start of the code path to the end, along with what they believe to be the output of the

[1] https://github.com/CodeReadingModule/CodeReadingNotesApp
[2] https://github.com/CodeReadingModule/CodeReadingMealPlannerApp
[3] https://github.com/CodeReadingModule/CodeReadingSocialMediaApp

codebase with the provided input.

Analyzing the student submissions helped us detect anti-patterns by identifying comments that they had filled in that were outside of the actual code path, incorrect descriptions in comments, or errors in what the student's prediction of the final output.

The code used meaningful names for classes, methods, and variables as per best practice for professional code [1], but did not contain any comments beyond those blank comment blocks, since a) the goal of the code reading exercises was to see how well the students could understand the code based upon reading the code, not reading code comments, and b) the code design was simple enough to not merit additional code comments.

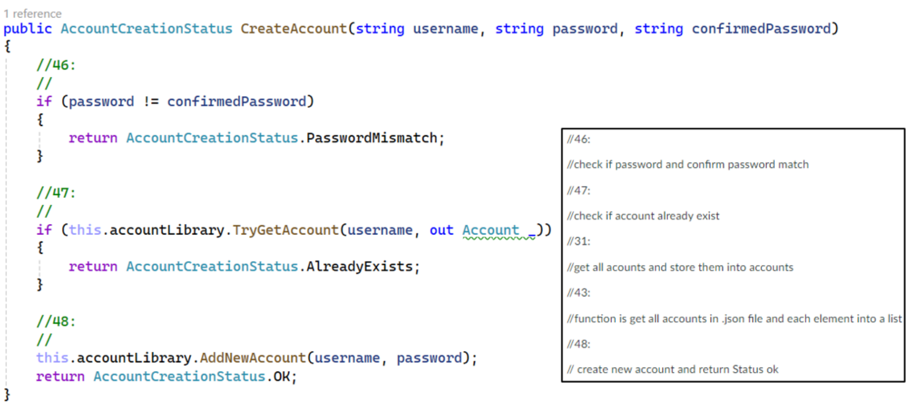


Figure 2: Example of empty comment blocks in a codebase and a student filling in each comment block in order of execution within the box (blocks 31 and 43 are elsewhere in the code).

5 Lesson plans and exercises

The Canvas module consists of six sections. The first section provides information about codebase structure, naming guidelines and conventions common in C# code or any similar object-oriented language, and ends with a short self-assessment quiz requiring students to analyze the file structure in the Notes codebase to answer questions on that information. The second section describes how to find the particular files that implement an Application Programming Interface (API), and ends with a short self-assessment quiz where students have to analyze the file structure of the Meal Planner codebase.

The third section includes information about common pitfalls to avoid when tracing code, based upon the issues we had observed in students doing the exercises in our first study [7]. It also includes a quiz which required students to trace through the Notes codebase with a given input in order to fill in a table documenting every step in the call stack. By documenting every step, students are made to see how the code path steps through multiple classes and utilizes outside libraries to complete its task.

Since the purpose of this Canvas module is to allow students to learn on their own, the fourth section of the module provides text and videos that show example solutions to the tracing exercises the students had just completed.

The fifth section is another quiz that requires students to trace through the Meal Planner codebase to document the flow of execution with a given input. The Meal Planner codebase contained a set of empty comment blocks that students fill in with their best interpretation of the code segment tied to each empty block. Students then submitted their answers in the form of an ordered list of their comments from the start of the code path to the end, along with what they believe to be the output of the codebase with the provided input. Once students had completed the quiz, we present them with text and videos that illustrate successful traces like those they were just tested on.

The sixth and final section is the final knowledge check. It is a quiz that requires students to trace through the Social Media codebase to answer a set of questions and to document the flow of execution through their own comments. Students were expected to trace through the application starting at various endpoints in the controller with different provided inputs. Much like the fifth section, students were asked to fill in comment blocks and submit an ordered list of comments for each endpoint they traced through. After submitting their ordered list of comments, students were also asked to provide the response from the endpoint they had just examined given the same input they used in their trace, as well as answer some additional questions on the flow of execution of each endpoint.

6 Results

To assess the code reading Canvas module, we provided it to the seventeen students in the autumn 2023 instance of the Software Engineering Studio course. Nine of the students completed the full module. The first author manually graded the student responses for the two tracing exercises and the final knowledge check. Responses to tracing questions started off as perfect scores and had points subtracted when different mistakes were made. The amount subtracted was dependent on the severity of the mistake, as defined in a rubric that listed 23 different mistakes observed. More details about the scoring and rubric are available in the first author's master's thesis [8]. One student was scored for the knowledge check, but not the two tracing exercises.

The resulting scores shown in Figure 3 suggest a general improvement across the two Canvas self-assessment exercises and the final Canvas knowledge check.

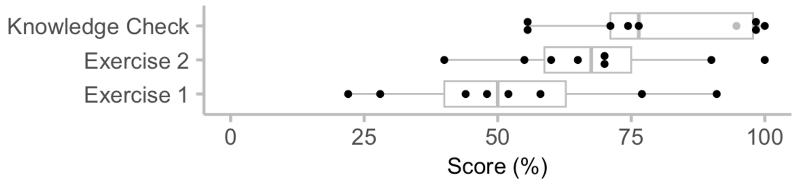


Figure 3: Results of the eight students who completed Exercise 1, Exercise 2, and the final knowledge check; and the one student (S4; grey point) who only completed the final knowledge check.

To see whether the Canvas module may have helped students avoid the three anti-patterns (non-critical/irrelevant files examined, call stacks not followed, uncorrected misinterpretations of code) documented in our prior study [7] we counted the number of each anti-pattern exhibited in their final knowledge check trace records. Figure 4 shows these on the right side. Exercise 1 and Exercise 2 show the number of anti-patterns exhibited by the professional software engineers and students in the prior study [7]. The correct trace path for each test required students to navigate through a different number of call stacks (2 for Exercise 1, 3 for Exercise 2, and 12 for the Knowledge Check); thus, for a better comparison we normalized the "call stacks not followed" metric to be a percentage of the total number of call stacks in the exercise.

Figure 4 shows that the number of anti-patterns exhibited by the nine students in the current study was lower than what we observed in the students in the prior study, though still not as good as the professionals in the prior study.

While not definitive due to the small sample size and other possible confounding factors, Figures 3 and 4 strongly suggest that the new learning module was effective at helping students learn how to read novel codebases.

Given the positive results of the assessment of the Canvas module's impact on student learning, we decided to package the Canvas module in a way that other faculty would be more likely to use in their courses, or that students could take on their own outside the context of a course. The main impediment was the amount of effort required to assess a student's code reading ability. It took roughly 90 minutes to grade and evaluate each student's responses. Performing this process manually was essential during the development, user testing, and evaluation of the Canvas module, but is too labor intensive to be

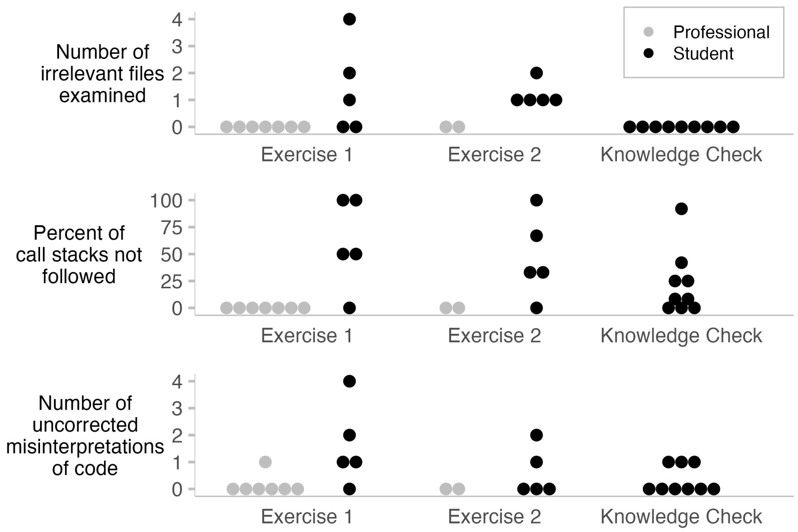


Figure 4: The number of anti-patterns exhibited by professionals and students in Exercise 1 and Exercise 2 of the prior study [7], and by the students in the Knowledge Check of this current study.

done as part of teaching a course using this module.

We thus revised the module to use completely automated self-assessment quizzes. We created a question bank for each quiz, and require students to score at least 75% before they can proceed to the next section. We packaged this module into its own Canvas course which is available online[4] and as a Canvas Course Export Package[5]. This new revision was used in the spring 2024 instance of the same course for the same related badge challenge. The instructor found that sixteen of the eighteen students were better able to navigate the code more directly without hints or help compared to prior to these modules being available.

7 Conclusions and Future Work

The performance of students throughout the code reading Canvas module suggests that the module can substantially improve students ability to read and comprehend unfamiliar code. Integration of such modules into computer sci-

[4]https://canvas.instructure.com/courses/9479990
[5]https://drive.google.com/file/d/1zsejLAR2P6JwFWCGXOtEUJcmWoCEWikq

ence courses early in the curriculum may benefit students throughout their undergraduate studies and into their futures.

The improved ability of students to read code may indicate that the modules were successful in lowering the cognitive load novice programmers may experience when presented with code reading problems for complex code [2, 3]. Future neurological studies conducted with students before and after such a module could help bolster the findings of studies like Peitek et al. [2] and Siegmund et al. [3] if student performance improved after the module and markers for cognitive load also decreased. Such an outcome would lend further support to the idea that increased cognitive load is a key driver for lowered performance in reading unfamiliar complex code for novice programmers.

We continue to use this Canvas module in our Software Engineering Studio course, and anecdotally it continues to seem to be helping students improve their ability to read novel codebases. Larger scale studies of the current code reading module would help to verify the results found by this small-scale test of the modules positive benefits. If this module was administered to a much larger beginner computer science class, the performance of those students in future courses throughout their curriculum could be compared to previous years for any sign of tangible benefit brought by the addition of the module.

References

[1] Robert Martin. *Clean code: a handbook of agile software craftsmanship*. Upper Saddle River NJ: Prentice Hall, 2009. ISBN: 978-0-13-235088-4.

[2] Norman Peitek et al. "Program Comprehension and Code Complexity Metrics: An fMRI Study". In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). Madrid, ES: IEEE, May 2021, pp. 524–536. ISBN: 978-1-66540-296-5. DOI: 10.1109/ICSE43902.2021.00056.

[3] Janet Siegmund et al. "Measuring neural efficiency of program comprehension". In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE'17: Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. Paderborn Germany: ACM, Aug. 21, 2017, pp. 140–150. ISBN: 978-1-4503-5105-8. DOI: 10.1145/3106237.3106268.

[4] Leigh Ann Sudol-DeLyser, Mark Stehlik, and Sharon Carver. "Code comprehension problems as learning events". In: *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*. ITiCSE '12: Annual Conference on Innovation and Technology in Computer Science Education. Haifa Israel: ACM, July 3, 2012, pp. 81–86. ISBN: 978-1-4503-1246-2. DOI: 10.1145/2325296.2325319.

[5] Yida Tao et al. "How do software engineers understand code changes? an exploratory study in industry". In: *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*. FSE '12. New York, NY, USA: Association for Computing Machinery, Nov. 11, 2012, pp. 1–11. ISBN: 978-1-4503-1614-9. DOI: 10.1145/2393596.2393656.

[6] Grant Wiggins and Jay McTighe. *Understanding by Design*. Association for Supervision and Curriculum Development, Mar. 15, 2005. 383 pp. ISBN: 978-1-4166-0225-5.

[7] Matthew Woerner, David Socha, and Mark Kochanski. "Code Reading: How Students and Professionals Dier". In: *The Journal of Computing Sciences in Colleges* 39.1 (Oct. 2023), pp. 28–37. ISSN: 1937- 4763. DOI: 10.5555/3636517.3636521.

[8] Matthew Stephen Woerner. "Identifying and Addressing the Gap Between How Students and Professionals Read Code". MA thesis. University of Washington, 2024.

Improving Introductory Java Programming Education Through ChatGPT*

Jingnan Xie[1]
[1] Department of Computer Science
Millersville University of Pennsylvania
Millersville, PA
jingnan.xie@millersville.edu

Abstract

The realm of introductory computer science (CS) education is swiftly changing, as educators actively pursue inventive strategies to captivate and empower students. This manuscript introduces a fresh methodology for teaching CS1 or CS2 courses, concentrating specifically on the fundamental principles of Java programming. Harnessing the capabilities of ChatGPT, an AI language model, we delve into how integrating conversational AI into the classroom milieu can foster a more dynamic and tailored learning journey. By furnishing a platform for students to pose inquiries, seek elucidation, and promptly receive feedback, ChatGPT functions as a virtual mentor, complementing conventional teaching methodologies. We scrutinize the potential repercussions of this approach on student learning outcomes (SLOs) and juxtapose it with traditional classroom paradigms. Furthermore, we deliberate on the ramifications of employing AI in education and its contribution to molding the trajectory of introductory programming courses.

1 Introduction

In recent times, artificial intelligence (AI) has made remarkable strides, particularly in the domain of natural language processing (NLP). Among these advancements stands ChatGPT, a cutting-edge language model renowned for its ability to generate text akin to human expression based on input prompts. ChatGPT has showcased its prowess across a spectrum of tasks, from language translation to coherent conversation holding with users, ushering in new horizons for AI integration across diverse fields, including education (for example, see [7]).

The landscape of computer science education, especially at the introductory level, often grapples with the challenge of student engagement and fostering active learning [4]. Traditional teaching methods like lectures and textbooks may not always cater to the varied learning styles and requirements of students. Additionally, the abstract nature of programming concepts can present obstacles for beginners. In response to these hurdles, educators are exploring alternative teaching methodologies for introductory programming courses, aiming to enhance student involvement, understanding, and retention.

This article introduces a pioneering approach to instructing CS1 or CS2 courses by harnessing the capabilities of ChatGPT to establish an interactive and personalized learning atmosphere. Similar research can be seen in [6] and [2]. Through ChatGPT integration into the classroom environment, students gain the ability to ask questions, seek clarification, and receive immediate feedback, thereby enriching their learning journey. The objective of this paper is to delve into the potential advantages of this method, assess its impact on student learning outcomes, and juxtapose it with conventional teaching techniques.

This paper is organized as follows: Section 3 illustrates a a course design of a 14-week CS1 course with ChatGPT and emphasizes on the interaction between students and AI. Section 4 shows how we can engage the students more and get better students learning outcomes with ChatGPT. The measurement of student successs is also provided. Section 5 demonstrates the challenges we face and the strategies used to solve these challenges.

2 Background

Genetic AI, known as Generative Pre-trained Transformers (GPT), signifies a significant leap forward in the realm of artificial intelligence (AI). These models undergo training on extensive corpora of text data, mastering the art of generating coherent and contextually relevant text based on given prompts. Among the illustrious examples of genetic AI stands ChatGPT, an innovation by OpenAI that has captivated widespread attention owing to its remarkable ability to emulate human-like conversations [3].

ChatGPT finds its foundation in the Transformer architecture, renowned for its utilization of self-attention mechanisms adept at capturing long-range dependencies within text sequences. This architectural marvel empowers ChatGPT to produce fluid and contextually fitting responses across a diverse array of prompts, rendering it highly adaptable for a multitude of natural language processing (NLP) tasks [5].

In the educational sphere, the integration of ChatGPT introduces a plethora of promising applications, each poised to revolutionize the learning landscape. Personalized tutoring emerges as a prominent avenue, facilitated by ChatGPT's conversational prowess, enabling educators to tailor interactive learning experiences tailored to the unique needs of individual students. Furthermore, the multifaceted capabilities of ChatGPT extend to content generation and language translation, offering invaluable support to educators in crafting diverse learning materials and breaking down language barriers within educational settings [1].

Moreover, the transformative potential of ChatGPT in education extends beyond conventional tutoring roles. With its ability to provide instantaneous feedback and elucidation on intricate concepts, ChatGPT transcends the limitations of traditional teaching methods, serving as a virtual mentor to guide students through their learning journey. This real-time support mechanism not only enhances student comprehension but also fosters a dynamic and engaging learning environment conducive to knowledge assimilation and retention [1].

3 Teaching CS1 and CS2 with ChatGPT

3.1 Course Design with ChatGPT Integration

Designing a CS1 or CS2 course with ChatGPT integration requires careful planning to ensure that students receive comprehensive instruction in programming concepts while leveraging the capabilities of AI for enhanced learning experiences. Below is a breakdown of a 14-week course outline, highlighting the role of AI and the recommended allocation of time for student interaction with ChatGPT:

1. Week 1: Introduction to Java Programming and ChatGPT (5% AI Interaction)
    - Introduce students to the basics of Java programming language.
    - Familiarize students with the ChatGPT interface and its capabilities.
    - Discuss the potential benefits of using ChatGPT as a learning resource.
2. Week 2-3: Basic Data Types and Variables (10% AI Interaction)

- Cover fundamental data types such as int, double, boolean, and char.
- Provide examples of variable declaration and initialization.
- Encourage students to ask questions about data types and variables using ChatGPT.

3. Week 4-5: Control Structures: Conditionals and Loops (15% AI Interaction)

   - Teach if statements, switch statements, and different types of loops (for, while, do-while).
   - Illustrate the use of control structures in solving programming problems.
   - Allow students to practice writing conditional statements and loops with guidance from ChatGPT.

4. Week 6-7: Arrays and Strings (10% AI Interaction)

   - Introduce arrays and strings as composite data types.
   - Demonstrate array manipulation techniques and string operations.
   - Assist students in understanding array indexing, traversal, and string manipulation through ChatGPT interactions.

5. Week 8: Midterm Review and Project Proposal (5% AI Interaction)

   - Conduct a review session covering topics learned in the first half of the course.
   - Guide students in formulating project proposals that demonstrate their understanding of programming concepts.
   - Offer assistance and feedback on project proposals through ChatGPT consultations.

6. Week 9-10: Functions and Methods (15% AI Interaction)

   - Introduce functions/methods as reusable blocks of code.
   - Discuss function definition, invocation, parameters, and return values.
   - Provide examples of function implementation and decomposition strategies.
   - Offer personalized assistance to students in designing and implementing functions using ChatGPT.

7. Week 11-12: Object-Oriented Programming (20% AI Interaction)
   - Introduce the principles of object-oriented programming (OOP).
   - Cover topics such as classes, objects, inheritance, polymorphism, and encapsulation.
   - Guide students in applying OOP concepts to solve programming problems.
   - Facilitate discussions on OOP principles and best practices with support from ChatGPT.
8. Week 13: Project Development and Troubleshooting (10% AI Interaction)
   - Allow students dedicated time to work on their programming projects.
   - Offer troubleshooting assistance and debugging support through ChatGPT consultations.
   - Encourage collaboration and peer feedback on project development.
9. Week 14: Final Project Presentation and Reflection (5% AI Interaction)
   - Conduct final project presentations where students showcase their work.
   - Facilitate peer evaluation and feedback sessions.
   - Reflect on the learning journey and discuss the role of ChatGPT in improving comprehension and problem-solving skills.

Throughout the course, it is recommended that students spend approximately 10-15% of their class time engaging with ChatGPT for interactive learning activities, clarification of concepts, and assistance with assignments or projects. This allocation ensures that students have ample opportunities to leverage AI resources effectively while also engaging in traditional classroom instruction and collaborative activities. By integrating ChatGPT into various aspects of the course, educators can create a dynamic and supportive learning environment that enhances student engagement and promotes deeper understanding of programming fundamentals.

Incorporating ChatGPT into the teaching of CS1 and CS2 courses requires a pedagogical approach that fosters inquiry-based learning, scaffolded instruction, formative assessment, and peer collaboration. By combining these elements with the capabilities of ChatGPT, educators can create an interactive and supportive learning environment that promotes student engagement and comprehension.

3.2 Examples of Student Queries

Teaching students how to ask questions effectively is crucial in the design of this course for several reasons. Firstly, it cultivates critical thinking skills by encouraging students to articulate their uncertainties and conceptual gaps. Secondly, it promotes active engagement and ownership of learning, as students become proactive in seeking clarification and deeper understanding. Thirdly, it fosters a collaborative learning environment where students can learn from each other's inquiries and responses. Finally, it helps students develop communication skills necessary for success in both academic and professional settings.

Student: "What are the different data types in Java?"
Response: "In Java, we have primitive data types such as int, double, boolean, and char, as well as reference data types like String. Let me give you some examples of how they are used..."

```java
int     age = 25;
double  height = 1.75;
boolean isStudent = true;
char    grade = 'A';
String  name = "John";
```

Student: "What is the difference between int and double?"
Response: "An int stores whole numbers, whereas a double stores decimal numbers. Let's break it down further..."

Student: "How do I use an if statement to check if a number is positive or negative?"
Response: "You can use an if statement with a condition to check if a number is positive or negative. Here's an example..."

```java
int number = -5;
if (number > 0) {
    System.out.println("The number is positive.");
} else if (number < 0) {
    System.out.println("The number is negative.");
} else {
    System.out.println("The number is zero.");
}
```

Student: "What does the 'else' keyword do?"
Response: "The 'else' keyword is used to execute a block of code if the condition in the 'if' statement evaluates to false. Let me explain it further..."

Student: "How do I use nested loops to print a multiplication table?"
Response: "Nested loops are loops within loops. Here's how you can use nested loops to print a multiplication table..."

```
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 10; j++) {
        System.out.print(i * j + "\t");
    }
    System.out.println();
}
```

Student: "Can you explain how the inner loop works?"
Response: "The inner loop iterates over the numbers 1 to 10 and prints the product of the current row and column. Let me illustrate it further..."

Student: "Can you debug this for me? int[] A = new int[5]; for(int i = 0; i<=5; i++) A[i] = i + 5;"
Response: "The issue lies within the loop condition and the array indexing. Here's the problem breakdown and the corrected code:

- Array Index Out of Bounds:

  In Java, array indexing starts from 0. So, if you have an array of size 5, valid indices range from 0 to 4, inclusive. Accessing index 5 would result in an "ArrayIndexOutOfBoundsException".

- Loop Condition:

  The loop condition i <= 5 will cause the loop to iterate 6 times (0, 1, 2, 3, 4, 5). However, since the array A has a length of 5, accessing index 5 will result in an out of bounds error. Here's the corrected code:..."

4 Student Engagement, Learning Outcomes, and Measurement of Success

The integration of ChatGPT into CS1 and CS2 courses has the potential to significantly impact student engagement, learning outcomes, and overall success in introductory programming education. In addition to fostering active learning and providing personalized support, it is crucial to establish metrics for assessing the effectiveness of this approach and measuring student success.

ChatGPT facilitates interactive and personalized learning experiences, allowing students to ask questions, seek clarification, and receive immediate feedback. By engaging in meaningful conversations with ChatGPT, students can deepen their understanding of programming concepts, develop problem-solving skills, and gain confidence in their abilities. Moreover, the continuous feedback provided by ChatGPT promotes a growth mindset and encourages students to persist in their learning journey.

4.1 Measurement of Student Success

- Formative Assessments: Formative assessments play a crucial role in measuring student progress and identifying areas for improvement. Educators can leverage ChatGPT to administer formative assessments, such as quizzes or coding challenges, that assess students' understanding of course content. By analyzing students' interactions with ChatGPT during these assessments, instructors can evaluate students' comprehension, identify misconceptions, and tailor instruction to address individual learning needs.
- Exam Scores: Traditional summative assessments, such as exams or assignments, remain valuable measures of student achievement. Educators can compare exam scores of students who have access to ChatGPT support with those who do not, providing insights into the impact of ChatGPT on academic performance. Additionally, exam questions can be designed to assess higher-order thinking skills and problem-solving abilities, reflecting the objectives of the course.
- Programming Projects: Programming projects serve as practical demonstrations of students' mastery of programming concepts and their ability to apply knowledge to real-world problems. By incorporating ChatGPT into project-based assignments, educators can observe students' use of ChatGPT as a learning resource and evaluate the quality and complexity of their solutions. Moreover, students can reflect on their experiences working with ChatGPT and its impact on their project outcomes.
- Course Completion Rates: Course completion rates provide insights into students' persistence and engagement with course materials. Educators can analyze data on student participation, interaction frequency with ChatGPT, and completion of assignments to gauge the effectiveness of ChatGPT in supporting student learning. Higher course completion rates among students utilizing ChatGPT may indicate increased motivation and satisfaction with the learning experience.

5 Challenges and Strategies

The integration of ChatGPT into CS1 and CS2 courses introduces various challenges that educators must address to maximize its effectiveness and ensure a positive learning experience for students. These challenges encompass aspects such as student behavior, technical limitations, and instructional design considerations. Below are some key challenges and strategies for overcoming them.

1. Student Cheating and Academic Integrity: One of the primary concerns associated with the use of ChatGPT is the potential for student cheating. Students may attempt to misuse ChatGPT to obtain solutions to assignments or exams without engaging in critical thinking or understanding the underlying concepts. To mitigate this challenge, educators can implement several strategies:

   - Design assessments that require higher-order thinking skills and problem-solving abilities, making it difficult for students to rely solely on ChatGPT for answers.
   - Clearly communicate expectations regarding academic integrity and the responsible use of ChatGPT as a learning resource.
   - Foster a culture of honesty and accountability by promoting discussions on ethical behavior and the consequences of academic dishonesty.

2. Engaging Students in Active Learning: Engaging students in active learning activities is essential for promoting deep understanding and retention of course material. However, integrating ChatGPT into the classroom environment may present challenges in maintaining student engagement. To address this challenge, educators can:

   - Incorporate interactive discussions, group activities, and problem-solving exercises that leverage ChatGPT as a collaborative learning tool.
   - Provide real-world examples and practical applications of programming concepts to demonstrate their relevance and foster student interest.
   - Use ChatGPT to facilitate peer collaboration and knowledge sharing, encouraging students to actively participate in class activities.

3. Monitoring and Supervising ChatGPT Usage: Ensuring that students use ChatGPT responsibly and for educational purposes requires proactive monitoring and supervision. Educators must establish mechanisms to track student interactions with ChatGPT and intervene if necessary to prevent misuse. Strategies for monitoring and supervising ChatGPT usage include:

   - Monitoring chat logs and analyzing student interactions with ChatGPT during class sessions to identify patterns of misuse or inappropriate behavior.

- Providing guidance and support on how to effectively utilize ChatGPT as a learning resource, including best practices for asking questions and seeking assistance.
- Implementing periodic check-ins and discussions on responsible AI usage to reinforce expectations and address any concerns or misconceptions.

4. Technical Limitations and Accessibility: While ChatGPT offers powerful capabilities for generating human-like text, it may encounter technical limitations or accessibility issues that impact its effectiveness in educational settings. Educators should be mindful of the following challenges:

   - Technical constraints such as internet connectivity issues or platform compatibility may hinder students' ability to access ChatGPT during class or while completing assignments.
   - Language barriers or linguistic complexities may pose challenges for students whose first language is not English, requiring additional support or resources to ensure equitable access to ChatGPT.
   - Ethical considerations regarding data privacy and security must be addressed to protect students' personal information and ensure compliance with relevant regulations and policies.

6 Future Work and Conclusion

The next step involves conducting an assessment of this teaching method, comparing its effectiveness with traditional approaches to determine its impact on student engagement and success. However, this evaluation remains unfinished due to time constraints and insufficient data. This assessment aims to gauge whether the new method can enhance both student participation and academic achievement, providing valuable insights into its potential advantages over conventional teaching methods.

In conclusion, the integration of ChatGPT into the teaching of CS1 and CS2 courses represents a promising avenue for enhancing student learning outcomes and fostering a more engaging and interactive learning environment. By leveraging the capabilities of ChatGPT, educators can provide personalized support, promote active learning, and empower students to succeed in introductory programming courses. As AI continues to evolve, its role in education is likely to expand, offering new opportunities for innovation and improvement in teaching and learning practices.

References

[1] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[2] Chung Kwan Lo. “What is the impact of ChatGPT on education? A rapid review of the literature”. In: *Education Sciences* 13.4 (2023), p. 410.

[3] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[4] Nurazlina Md Sanusi, Massila Kamalrudin, and Syahida Mohtar. “Student engagement using learning management system in computer science education”. In: *International Journal of Recent Technology and Engineering* 8.2-11 (2019), pp. 743–747.

[5] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[6] Ramazan Yilmaz and Fatma Gizem Karaoglan Yilmaz. “Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning”. In: *Computers in Human Behavior: Artificial Humans* 1.2 (2023), p. 100005.

[7] Xuesong Zhai et al. “A Review of Artificial Intelligence (AI) in Education from 2010 to 2020”. In: *Complexity* 2021 (2021), pp. 1–18.

Studying Financial Data with Macroeconomic Factors using Machine Learning*

Sai Sravya Anem, Md Amiruzzaman, Ashik Ahmed Bhuiyan
Department of Computer Science
West Chester University, West Chester, PA 19383
{MAmiruzzaman}@wcupa.edu

Abstract

This paper focuses on the prediction of stock indices through machine learning, focusing on macroeconomic factors and market sentiment generation. It centers on major US stock index funds, notably the S&P 500, and their correlation with key economic indicators like GDP, unemployment, CPI, money supply, and retail sales. Utilizing economic data from diverse sources such as the Federal Reserve, NASDAQ, and news websites, the study cleans and transforms datasets to estimate quarterly fund returns. Employing tree-based algorithms, particularly XGBoost, enables accurate predictions. Moreover, the paper evaluates index forecast performance across various market cycles and geopolitical events. It also uses traditional NLP methods and large language models to explore market sentiment generation, offering comprehensive insights. In essence, this paper sheds light on the predictive power of macroeconomic factors on stock indices and the nuances of market sentiment analysis, leveraging both conventional and advanced techniques.

1 Introduction

The stock market, often known as the equity or share market, serves as a pivotal arena where investors trade shares or ownership stakes in publicly listed companies [1]. Its performance significantly influences economic growth and stability [11]. Macroeconomic factors, which affect the entire economy, wield profound influence over businesses and, consequently, the stock market. These factors are acknowledged for their ability to sway investor sentiment and impact stock prices. For instance, GDP growth reflects a country's overall economic health, with robust growth correlating with higher corporate earnings and rising stock prices [6].

Interest rates are pivotal in determining capital costs for businesses and investors. Lower rates often stimulate economic activity and lift stock prices [6]. Conversely, inflation erodes the value of money, with high and erratic rates undermining consumer confidence and disrupting business, thus adversely affecting stock markets [9]. Rising interest rates can elevate borrowing costs, potentially reducing corporate profitability and depressing stock prices [4]. Stock index data was sourced from the Nasdaq API, while macroeconomic indicators were extracted from the Federal Reserve Economic Data (FRED), a comprehensive repository with over 816,000 economic time series [3].

This paper aims to predict stock market trends using macroeconomic factors such as GDP, unemployment, CPI, inflation rate, and interest rates. Regression models employing the XGBoost algorithm predict stock index returns for the years 2020–2024. Performance metrics like MSE, RMSE, and MAPE validate actual versus predicted values, facilitating comparison across different economic conditions. Additionally, specific economic settings forecast performances are analyzed to comprehend the key macroeconomic drivers and market consensus [7]. In addition to forecasting, understanding current market sentiment is crucial for informed investment decisions. Major news publishers and media outlets play a vital role in portraying market sentiment. News articles are analyzed using various methods to generate numeric sentiment scores and textual summaries based on large language models to gauge sentiment.

2 Existing work

Predicting stock market trends has historically presented a challenging endeavor due to the multitude of factors influencing stock prices, including product demand, sales, production, investor sentiment, government policies, and economic conditions. Accurately forecasting stock prices holds immense potential for substantial profits. One notable approach involves leveraging eXtreme Gradient Boosting (XGBoost), renowned for its efficiency and accuracy rates surpassing 87% for both 60-day and 90-day periods. Compared to traditional non-ensemble learning techniques, XGBoost exhibits superior performance in forecasting stock market fluctuations, aiming to predict both upward and downward movements.

A study by Dey et al. explores the use of XGBoost to forecast stock market trends, employing a binary system where +1 indicates an anticipated increase in stock valuation and -1 signifies a projected decrease in prices [2]. Another paper by Nousi Christina, titled "Stock Market Prediction using Sentiment Analysis," focuses on Microsoft's stock movement by analyzing historical data and sentiment from social media platforms. The study collects approximately 90,000 tweets from Twitter and 7,440 tweets from StockTwits, alongside historical data from the Finance Yahoo website spanning a specific period. Sentiment analysis of social media data is conducted using Python libraries TextBlob and VADER (Valence Aware Dictionary and Sentiment Reasoner), while various machine learning models such as KNN, SVM, Logistic Regression, Naïve Bayes, Decision Tree, Random Forest, and MLP are implemented. Results indicate that utilizing tweets from Twitter with VADER for sentiment analysis, SVM yields the highest f-score of 75.9% and an Area Under Curve (AUC) of 65% [10].

In this paper, we adopt XGBoost to model a multivariate time series for forecasting. Beyond the model's insights into future trends, we aim to augment our analysis with real-time market sentiment to enhance decision-making. Leveraging natural language processing algorithms like Semi-Normalization and VADER, we quantify general market sentiment through various media sources such as news articles, tweets, and volatility indices. This sentiment analysis aids in understanding the market sentiment before making investment decisions.

3 Method

This section presents the data used for stock index and macroeconomic indicators data preparation and cleaning are explained and the extraction of market news is explained. Furthermore, the Machine Learning design is presented with the models that help in predicting the stock index. Market sentiment score is explained using VADAR and Semi-Normalization techniques and the normal distribution of data is considered.

3.1 Data Extraction

To elucidate the complexities and inherent dynamics of market movements through economic indicators, this investigation harnesses data from the Federal Reserve Economic Data (FRED), hosted by the Research Division of the Federal Reserve Bank of St. Louis. FRED is an extensive repository, boasting over 816,000 economic time series drawn from many sources ("FRED," 1991). Additionally, data pertaining to alternative assets such as Bitcoin, Gold, Bonds, and Treasuries are procured via Quandl[12], which serves as a premier publisher for financial and economic datasets. Financial market news and geo-political events indicate general market sentiment. News articles from leading investment research websites and news websites can be extracted using their APIs.

By leveraging these on-demand resources, we can source the most up-to-date financial news using our analytical tools. Each piece of news from the API comes with a bunch of information. We curate the output fields to use the information like when it was published, the stock tickers, and the description. Without reading the complete news article page data, we have chosen limited fields that provide a summary. The news data retrieved through the APIs is accompanied by extensive metadata.

Forecast Method To forecast the market movement, we are modeling using macro-economic indicators and news articles and leveraging a large language model like ChatGPT 4 or Hugging Face for sentiment generation and ensemble to predictions. Stock Indexes here act as labels for back-testing the model forecasts. As Figure 1 explains, input time series data is decomposed to remove any seasonality and interpolate less frequent macro features to match the daily frequency features like Indices, Volatility index, etc. And finally, they are merged into one data frame operating on the same time horizon and frequency.

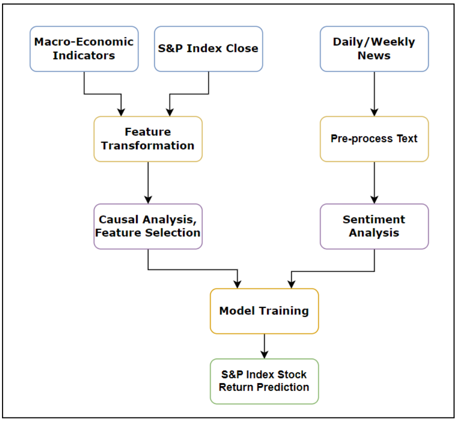


Figure 1: Forecast Processing Method

3.2 Sentiment Analysis

Sentiment analysis can be a useful tool for understanding market sentiment and identifying potential shifts, but it should not be the sole basis for making investment decisions [8].

Sentiment analysis can help identify how the public or traders feel about a particular stock or the market. Positive sentiment may indicate optimism, while negative sentiment may suggest pessimism. Sentiment analysis alone is not a reliable method for predicting stock market movements. It can provide indications of short-term market sentiment shifts, but it doesn't necessarily offer predictive accuracy for longer-term trends or specific price movements. Market sentiment can be heavily influenced by news, social media, and public opinion. It's essential to filter out irrelevant or biased sentiments to obtain meaningful insights.

3.2.1 Sentiment Analysis using VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool specifically attuned to social media sentiments. It is unique because it is sensitive to both the polarity (positive/negative) and the intensity (strength) of emotions. VADER has been found to be quite effective in handling text that contains emoticons, acronyms, and slang, which are commonly found in social media content [5].

$$C = \frac{v_s}{\sqrt{v_s^2 + \alpha}} \tag{1}$$

Where C is the compound score, v_s is the total score of each sentence word with a sentiment rating according to the VADER lexicon. Each word's score is the sum of its valence, determined by the lexicon and modified by the heuristic rules (such as punctuation, capitalization, degree modifiers, contrastive conjunctions, and negation). α is a normalization parameter to balance the distribution, ensuring that the scores are proportionately scaled between -1 and +1. This compound score is the most used metric to determine the overall sentiment of a text. A positive compound score indicates a positive sentiment, a negative compound score indicates a negative sentiment and a compound score near zero suggests a neutral sentiment [5].

3.2.2 Sentiment Analysis using Semi Normalization of Positive-Negative Words

Normalization, in the context of sentiment analysis, involves adjusting the raw counts to account for variations in text length, intensity modifiers, or other factors that can affect the sentiment score. Semi-normalization is a simpler form of normalization that may adjust these counts or scores to a standard scale or may account for text length without considering the full range of possible modifiers [13]. In this method, we calculate the sentiment score by evaluating the ratio of the count of positive words (P) and the count of negative words (N) + 1. Since there is no difference in values involved, the sentiment value will always be more than 0. Also, adding 1 in the denominator would save "zero division error" [14].

One common approach to semi-normalization might involve calculating the difference between the positive and negative word counts and then dividing by the total number of words in the text to standardize the score according to text length. The sentiment score is calculated using the counts from the positive and negative words, often adjusted or normalized in some way. A basic formula for a sentiment score with semi-normalization might look like this:

$$S = \frac{P}{N + 1} \tag{2}$$

Where S is the sentiment score, P is the positive word count, N is the negative word count.

This method is relatively simple and can be a good starting point for basic sentiment analysis tasks. However, it may not always capture the nuances of sentiment, such as sarcasm, negations, or intensity of emotions, which more sophisticated NLP techniques can address [13].

4 Results and Discussion

4.1 Overall Modeling XGBoost

We used 20 years of data from 1990-2019 for the training and later used the data from 2020-2024 for the testing. Over the 4 year of time horizon, we have had challenging market conditions starting with COVID and followed by quantitative easing leading to strong growth of the stocks. The market subsequently undergone a correction due to the FED's move to raise the interest rates for an extended time. It could be a challenging forecasting problem to evaluate against unlike a traditional sales inventory use-case. The prediction horizon is tried at different frequencies, such as 7 days, 30 days, 90 days and one year. 90 days forecast is relatively more accurate.In Figure 2 below we have overlayed the actual SPX vs Predicted price value. The features used for training are sometime non-stationary in nature. To address the issue, we also tried forecasting using another XGBoost version with the transformations to stationarize and remove any seasonality and trend factors in addition. From Table 1, we calculated the SPX predicted value from the returns value.

Table 1: Actual vs Predicted SPY price

Date	SPX	SPX_future_ret_90d	pred_label	SPX_pred
2019-12-02	3113.87	-0.079	-0.079	2865.078
2019-12-03	3093.20	-0.105	-0.219	2415.653
2019-12-04	3112.76	-0.101	-0.086	2844.269
2019-12-05	3117.43	-0.086	-0.086	2848.536
2019-12-06	3145.91	-0.099	-0.099	2832.117

We used k-folds with 50 iterations and considered the best MAE values. This tuning helped in generating output better compared to normal xgboost testing. Figure 3 below explains the prediction performance in terms of % returns. Returns can be positive or negative in percentage.

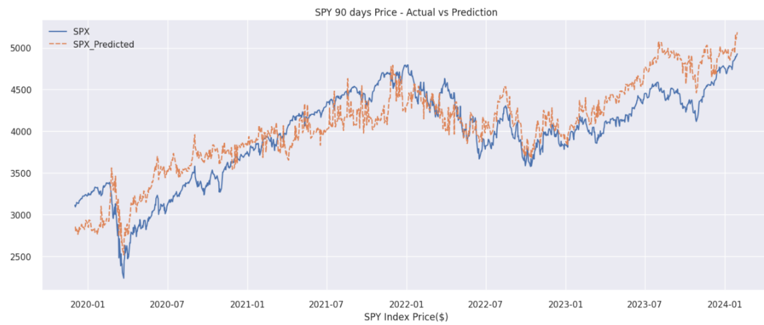


Figure 2: Predicting SPY Price after 90 days

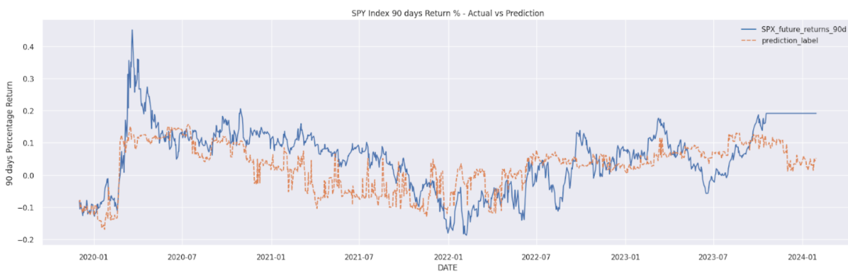


Figure 3: Predicting SPY % Returns after 90 days

4.1.1 Best Performance

After the bull-run which lasted close to 20 months from Feb 2020-Jan2022 the stock market had seen a correction for more than 8 months. The economic conditions like unemployment, interest rates, CPI are observed to recover and were stable during this time. This market scenario is observed to be the best prediction phase by the model assuming due to the market stability. From Figure 4 , we observe a bias in the model performance due to constant negative error (over estimation) during the interval. The average percentage error is observed to be less than 5 % which is impressive.From the Table 2, we observe the mean absolute percentage error (MAPE) is around 2.3% and overall, a negative bias indicating some degree of bias.

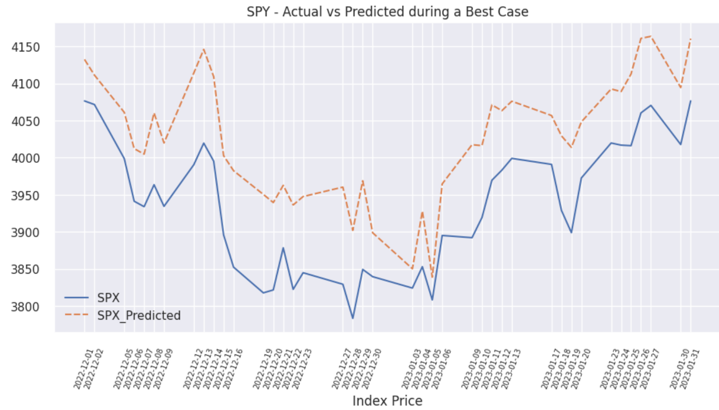


Figure 4: SPX Actual vs 90 day forecast during a best-case scenario

Table 2: Evaluation Performance in Best Performance

Bias	-90.258
MSE	8887.022
MAE	90.258
RMSE	94.271
MAPE	0.0229

4.1.2 2021-22 Market Peak (Bull Run)

Post-COVID affecting the markets, FED tried aggressive market stimulation by providing more liquidity and lowering interest rates. This resulted in a market recovery and further advance into a long bull run almost doubling the SPX index value from the COVID lows (2200). Here we have selected a few months between Dec 2021 to Feb 2022 to explain how resilient the model is about predicting the market top. Figure 5 explains the actual SPY value vs prediction using the model. Table 3 explains the forecast evaluation results for this scenario using measures like mean square error, mean absolute error, root mean square error, mean absolute percentage error. The average absolute percentage error is close to 5%.

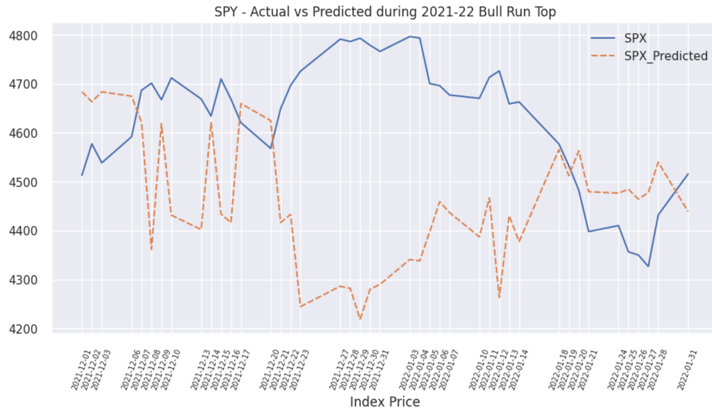


Figure 5: Market Peak

Table 3: Evaluation Performance in Market Peak

Bias	168.492
MSE	79729.445
MAE	230.859
RMSE	282.364
MAPE	0.0492

4.1.3 COVID Downturn

The world and humanity had experienced its most difficult times starting in early 2020 due to COVID outbreak. Asset classes like real estate and equity market activity volume is impacted heavily resulting in a sharp downtrend. A long-term prediction model will be uninformed to make a logical prediction in such use cases and considered as a limitation. As per the figure it is evident that we cannot predict the disaster which we never imagined.

Table 4: Evaluation Performance in Covid Breakout

Bias	-50.846
MSE	119305.585
MAE	335.905
RMSE	345.406
MAPE	0.113

The model forecast quality is evaluated using 5 metrics to indicate reliabil-

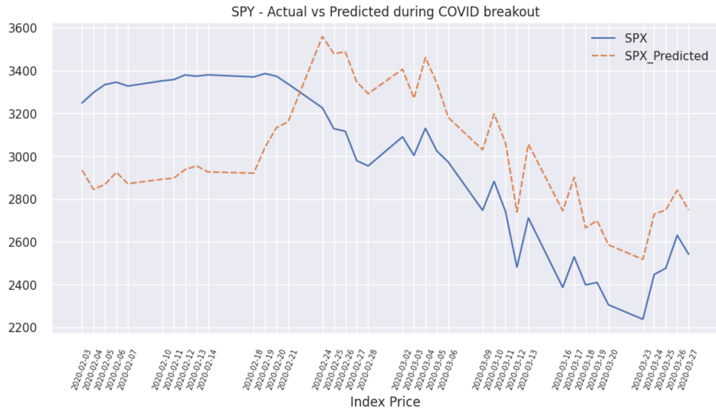


Figure 6: Covid Breakout

ity. The calculations above are the average of each statistic during the selected months to estimate the market behavior during key economic events. From the above Table 4, MAPE scores of 0.113 we can conclude the mean percentage error increased by 2 times compared to overall prediction performance. The model shows a bias characteristic for overestimating in this case for observations after COVID outbreak. In such market conditions it would be ideal to gauge the market sentiment in our investment decisions in addition to the forecast.

4.2 Market Sentiment Analysis for year 2022, 2023

Using Polygon API, we attempted to extract up to 100 news articles daily for the year 2022, 2023. This helped us to extract more relevant information about the stocks in SPY portfolio. Almost half of the news articles we resourced have stocks outside SPY 500 index which are filtered out to estimate sentiment on relevant news. The API response comes with the title and the article description in two columns and are combined into one column to provide a better context. We started with estimating the sentiment scores for each news text using Semi-Normalization and VADER methods.

From Figure 7, we are interested to understand if the news sentiment is really contributing to the stock index moves. Our initial observation suggests a positive correlation for Semi-Normalization method than VADER. Also, the below Spearman correlation suggests a statistically significant correlation from the Figure 8.

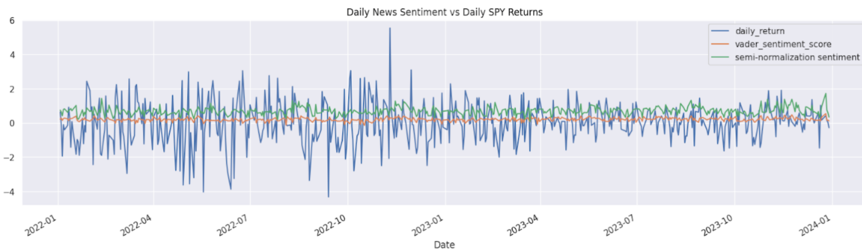


Figure 7: Daily News Sentiment vs Daily SPY Returns

```
0 days past news sample sentiment Spearmans correlation coefficient: 0.426
********************************************************************************
Sentiment are correlated (reject H0) p=0.000
********************************************************************************
1 days past news sample sentiment Spearmans correlation coefficient: 0.008
Sentiment are uncorrelated (fail to reject H0) p=0.866
2 days past news sample sentiment Spearmans correlation coefficient: -0.061
Sentiment are uncorrelated (fail to reject H0) p=0.175
3 days past news sample sentiment Spearmans correlation coefficient: 0.017
Sentiment are uncorrelated (fail to reject H0) p=0.703
4 days past news sample sentiment Spearmans correlation coefficient: -0.037
Sentiment are uncorrelated (fail to reject H0) p=0.415
5 days past news sample sentiment Spearmans correlation coefficient: -0.036
Sentiment are uncorrelated (fail to reject H0) p=0.419
6 days past news sample sentiment Spearmans correlation coefficient: -0.061
Sentiment are uncorrelated (fail to reject H0) p=0.177
7 days past news sample sentiment Spearmans correlation coefficient: -0.006
Sentiment are uncorrelated (fail to reject H0) p=0.893
8 days past news sample sentiment Spearmans correlation coefficient: 0.035
Sentiment are uncorrelated (fail to reject H0) p=0.432
9 days past news sample sentiment Spearmans correlation coefficient: -0.027
Sentiment are uncorrelated (fail to reject H0) p=0.548
```

Figure 8: Spearsman Coefficient Relation-Semi Normalization

5 Conclusion

The U.S. stock market, encompassing exchanges like the New York Stock Exchange (NYSE) and NASDAQ, is a complex ecosystem where shares of publicly held companies are traded. The S&P 500 Index, reflecting the performance of 500 large-cap U.S. equities, serves as a pivotal indicator of market health, offering insights into risk and return dynamics. Machine learning applications in finance are reshaping forecasting and risk management, providing predictive analytics for critical decision-making. Forecasting stock market trends using macroeconomic factors entails analyzing broad economic indicators to anticipate market movements and their impact on stock prices. Key factors include GDP growth, unemployment rates, inflation, interest rates, and monetary policies. These indicators influence consumer and business spending, thereby impacting corporate earnings and stock performance. However, predicting stock market behavior accurately amidst myriad variables remains challenging. In this study, we aimed to model these indicators to forecast S&P index performance over a three-month horizon using machine learning techniques such as forecasting and sentiment analysis. We utilized data from Quandl API for S&P index performance and Federal Reserve Economic Data (FRED) for critical macro indicators. After data preprocessing, including handling missing values and decomposing time series, we applied Random Forest and XGBoost models, achieving a mean absolute percentage error close to 8%. We evaluated model performance across various economic and social conditions, acknowledging limitations in adapting to future events. To augment forecasting decisions, we incorporated market sentiment analysis. Understanding sentiment aids in making informed investment choices; bullish sentiment may indicate an overvalued market, while bearish sentiment may present buying opportunities. We leveraged Polygon APIs to extract market news and applied sentiment analysis techniques such as Semi-Normalization and VADER. Our analysis revealed a positive correlation between market sentiment and S&P returns. In conclusion, integrating macroeconomic indicators and live news significantly enhances the accuracy and efficiency of index fund forecasting. By combining forecasting models with sentiment analysis, investors can make more informed decisions, navigating the complexities of the stock market with greater confidence.

References

[1] James Chen. *What Is the Stock Market, What Does It Do, and How Does It Work?* Investopedia. 2022. URL: `https://www.investopedia.com/terms/s/stockmarket.asp`.

[2] Shubharthi Dey et al. "Forecasting to Classification: Predicting the direction of stock market price using Xtreme Gradient Boosting". In: *PESIT South Campus* (2016), pp. 1–10.

[3] FRED. *FRED: Federal Reserve Economic Data.* FRED. 1991. URL: `https://fred.stlouisfed.org/`.

[4] JC Hull. "Options, Futures, And Other Derivatives.–9th ed.–Boston [etc.]" In: *Pearson Education, Inc* 2012 (2015), p. 2009.

[5] Clayton Hutto and Eric Gilbert. “Vader: A parsimonious rule-based model for sentiment analysis of social media text”. In: *Proceedings of the international AAAI conference on web and social media*. Vol. 8. 1. 2014, pp. 216–225.

[6] N Gregory Mankiw and Mark P Taylor. *Economics*. Cengage Learning EMEA, 2020.

[7] N.G. Mankiw. “Macroeconomics, 7th Edition”. In: Worth Publishers, 2010. URL: `http://www.worthpublishers.com/mankiw`.

[8] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. “Sentiment analysis algorithms and applications: A survey”. In: *Ain Shams Engineering Journal* 5.4 (2014), pp. 1093–1113. ISSN: 2090-4479. DOI: `https://doi.org/10.1016/j.asej.2014.04.011`.

[9] Frederic S Mishkin and Stanley G Eakins. *Financial markets and institutions*. Pearson Education India, 2006.

[10] Christina Nousi. “Stock market prediction using sentiment analysis”. In: (2021).

[11] Keith Pilbeam. “Finance and financial markets”. In: Bloomsbury Publishing, 2018, pp. 215–246.

[12] Quandl. *Nasdq-Quandl*. FRED. 2024. URL: `https://data.nasdaq.com/publishers/QDL`.

[13] Shipra Saxena. *Tokenization and Text Normalization*. 2021. URL: `https://www.analyticsvidhya.com/blog/2021/03/tokenization-and-text-normalization/`.

[14] Rahul Shah. *Different Methods for Calculating Sentiment of Text*. 2021. URL: `https://www.analyticsvidhya.com/blog/2021/12/different-methods-for-calculating-sentiment-score-of-text//`.

English to American Sign Language: An AI-based Approach*

James May, Kyle Brennan,
Stefanie Amiruzzaman, Md Amiruzzaman
Departments of Languages & Cultures and Computer Science
West Chester University, West Chester, PA 19383
{SAmiruzzaman, MAmiruzzaman}@wcupa.edu

Abstract

This study presents an Artificial intelligence (AI)-based approach to translating English to American Sign Language (ASL). It involves capturing audio from a speaker (i.e., a hearing person) and translating that into text, then translating the text into ASL sentences. The process involves a case study and a computer vision approach. A prototype is implemented using an audio-to-speech model, Natural language processing (NLP) for ASL, and MediaPipe a computer vision tool. Future work will include performance improvement and a complete framework to translate English to ASL and ASL to English communication.

1 Introduction

American Sign Language (ASL) is a sign language that is predominantly used by Deaf[1] and Hard of Hearing (HoH) people in the USA and Canada [13]. According to an article published by Mitchell, Young, Bachelda, & Karchmer (2006)[13], there are more than 500,000 people in the USA who use ASL as their primary language. The statistic is over a decade old, which means that

[1]Deaf people are those who knows the Deaf culture and tend to communicate in sign language as their first language, and deaf are those who have hearing loss (not aware of Deaf culture and not native sign language users)

the number has doubled by now if not tripled. A recent article claimed that approximately 3% of people in the USA whose primary language is ASL [12]. It would be unwise to assume that all Deaf people use English as their primary language, which is not the case [2, 13].

Most automated solutions that are tied to reducing the communication gap between the Deaf and hearing assume that speaking English is the way to overcome it. The reason researchers try to use AI-based approaches to provide solutions for this task is because of its performance in accuracy and swiftness to produce results. The AI-based approach is not only popular in this field but also in many fields [1, 20]. While, researchers focused on developing solutions to establish a communication medium between Deaf and hearing world; often the main focus was to improve the accuracy and speed. Perhaps, mostly because the research was led by hearing researchers, and little to no research studies involved Deaf researchers.

To the best of our knowledge, [2] included a Deaf researcher in the development process. As most previous studies involved hearing researchers, thus their work work did not fully capture the need of Deaf users. Most studies tried to provide solutions that involved ASL to English and English to text (i.e., English texts). This study focus on overcoming the gap and truly develop a framework that would allow Deaf/HoH people to see a machine vision based translation of English to ASL. This study involves a Deaf educator and researcher in this study to ensure its true use for Deaf/HoH people.

2 Related work

There have been many attempts to automate processes so that the Deaf and hearing world can communicate. However, this is not an easy task as the conversion of ASL and English is not in exact grammar order. For example, in English, we said, "I am going to the store." In ASL, we sign, "STORE, ME GO." Existing works mostly focused on recognizing ASL numbers and alphabets and tried to translate them into English [15, 4].

Over the last decade, many studies focused on automating ASL to English communication. For example, Starner et al., used Hidden Markov Models (HMM) [18] to recognize sign language. In their study, the authors focused on hand gesture movement and track them using a camera; only 40-word lexicon was considered. Gaus and Wong's [6] study focused on recognizing ASL sentences by using a camera to track the user's hands. Qutaishat et al. [14] developed an approach that mainly focused on recognizing static signs. This study was divided into two phases: (a) feature extraction and (b) classification. In the feature extraction phase: from images features were extracted using Hough Transformation. In the classification phase: these features are then passed as input to the Neural Network classification model and recognize signs.

Many other existing work such as [18], [14], and [8] concentrated on recognizing the ASL fingerspelling, alphabets and numbers from 0 to 9 only. Other studies, such as [9] focused on recognizing American Sign Language alphabets, and Bellen et al. [5] focused on ASL based gestures during video conferencing.

This current study focuses on recognizing both English and ASL sentences. This study goes beyond of recognizing just alphabets and numbers, rather complete sentences and therefore focuses on conversations. It allows English to ASL translation using NLP and machine vision. The process is guided and evaluated by a Deaf educator who verified the results as the development process continued.

3 Method

This study was done in three stages: (1) capture audio from a speaker and convert it to text, (2) use Natural language processing (NLP) to convert the text to ASL sentence, and (3) map the ASL sentence to MediaPipe animation. The section 3.1 describes how the audio to text capture was done, section 3.2 explains how the language used must be processed, and section 3.3 explains how the ASL sentence to animation was done. Fig. 1, presents the framework of this study.

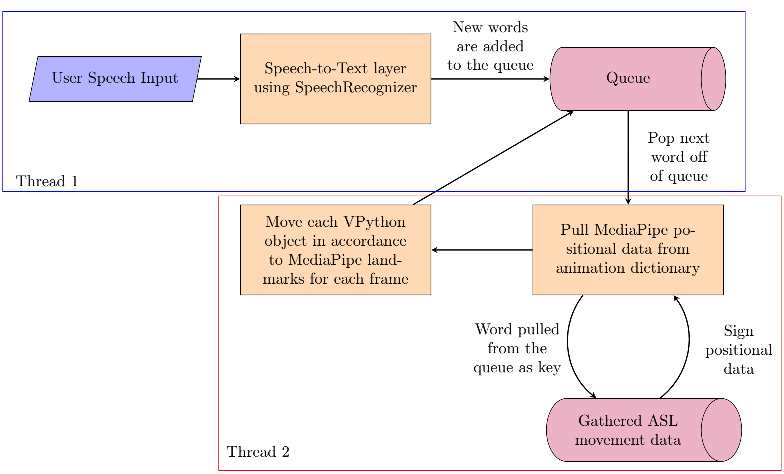


Figure 1: Framework of the entire Speech-to-ASL pipeline; processes for STT and animation are on separate threads.

3.1 Speech Collection

We utilized the "speech_recognition" and pyttsx3 Python libraries to assist with the live translation. The speaker's voice is taken as an input through the default communication device of your computer, and the corresponding text is transcribed using the Google Cloud Speech-to-Text (STT) API as the reference framework. This text is then automatically updated into a text file (.txt) to be extracted out, tokenized and parsed, and read into our animation database to retrieve a prerecorded animation of the signs for each word using a custom grammar for English to ASL translation.

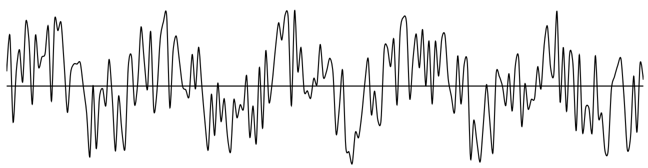


Figure 2: Example of a compressed sound wave, represented as a continuous distorted sine wave.

The way that the libraries can parse out words from speech is through a deep understanding of how sound dynamics works. Sound waves are just modified sine waves, having crests and troughs as their local maximums and minimums, respectively, called the amplitude of the wave. The range of the amplitudes in a given sound wave can be expressed by the time domain, which is a measure of how loud each sampled part of the wave is.

To convert an audio signal into something a computer can utilize, the time domain must be transformed into what is called the frequency domain, which is a measure of the pitch of the produced sound when measured at a certain sample rate. Sample rate defines the number of samples per second (in Hertz (Hz)) taken from a continuous signal to make a discrete or digital signal that computers are able to process. Because a sine wave is a continuous function, there must be a transformation from the time domain to the frequency domain, which we call the Fourier transformation.

$$F\{g(t)\} = \left(\int_{-\infty}^{+\infty} g(t)\, e^{-i\omega t}\, dt \right),$$

where the $e^{-i\omega t}$ component represents a complex exponential function which, for the sake of this paper and simplicity, is used to represent the individual sine waves of the entire recording. The discrete frequencies that the function will return allow us to parse out each unique sound from the waves; each part of the sound wave will create an individual sound which the libraries will use to recognize what sounds go where in a sentence in order to splice together an English phrase using the Google Cloud STT API. When we speak into a microphone, the microphone sends that audio signal to the model, then outputs a string together sentence as a console input for the animation model (See Section 3.3).

This model currently requires further development, as mentioned in section 5. In this study mainly focused on developing the prototype as part of our proof-of-concept (POC); enhancement and further improvement is going to be part of our future work.

3.2 Natural Language Processing

After processing the audio input and returning a string of English text, the program will undergo a couple of language processing steps in order to achieve an accurate and flexible approach to translate the texts into ASL signs. The first transformation that we will perform before adding a word to the queue is to check it against a list of articles we have established to not be valid words in actual ASL structure/grammar. Words like 'the,' 'a' and 'are' do not get added to the queue of words to be processed (see Fig. 3). After this step we would have a full body text of valid words that are grammatically correct to use in the ASL language. Later in the program runtime when searching the dictionary for a word in the text in the queue, we have also implemented safeguards to ensure all words are able to be communicated, even if not fully implemented in the sign dictionary (see Fig. 3).

The program first checks to see if a word is in the dictionary before performing the sign for the user. If a sign is not present in the dictionary it will then be broken into individual letters to be signed out to ensure that any word spoken can be communicated to the user. In the future we plan to implement two more NLP steps into the program. One step will focus on finding compound ASL words like "thank you" and "how are you?" after processing the audio input into a string. We also plan on making a final step before animating the signs in real time that takes the spoken language and translates the English grammar into the appropriate ASL grammar for one-to-one translation between the two languages (see Fig. 3).

3.3 MediaPipe Framework

Our work rests heavily of utilizing the MediaPipe's object detection framework for use in detecting the face and hands of the person in the frame of a video or camera stream [11]. *MediaPipe* (MP) is a library tool that uses Artificial Intelligence (AI), Machine Learning (ML), and computer vision techniques to develop live and streaming media. Specifically, MP contains 478 landmarks for the face, 21 for both left and right hands, and 33 for the pose of the

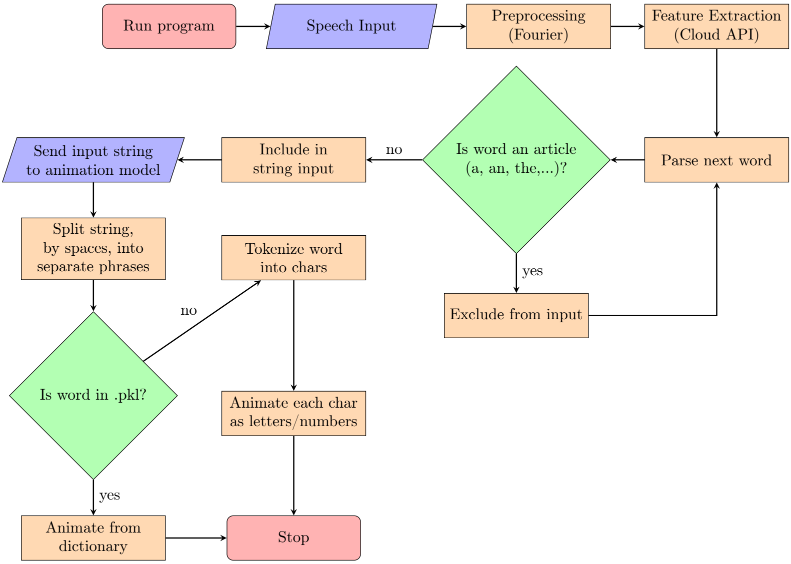


Figure 3: Flowchart diagram of the speech-to-text processing pipeline.

body [3]. These landmarks contain three dimensional positional data for each keypoint. This means MP is an excellent tool for use in real-time motion-capture animation, and keeping track of where each part of the body is in a camera frame. In this work, we have used MP for both gathering and storing our motion captured animation data.

3.3.1 ASL Animation Database Collection

In order to pull from a collection of animations during live translation, it is imperative for the program to have a collection of correctly ASL signs to be animated live for the user. In order to accomplish this, we have developed a small database of motion-captured data stored in a key-value format to be accessed during translation (see Fig. 4).

The data collected through motion-capture were recorded on a 720p 30fps webcam. Signage references were collected from the image dataset, as well as various instructive YouTube videos, to ensure the most accurate depiction of each sign was captured for clarity. We have selected a set of basic words to be recorded for sentence creation in our model, each of various lengths, where keypoint tensors are saved into a pickle (.pkl) file to be replayed at a later time. A .pkl file is a format primarily used for the serialization of vector data, in our case, where each sign recorded has the name of the sign as a key that can retrieve the collection of keypoint movements as an output value (see Fig. 4).

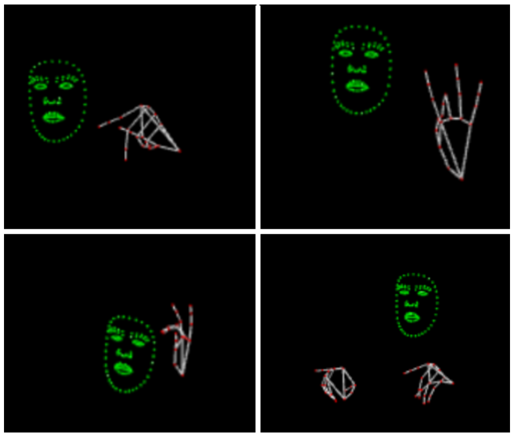


Figure 4: In program representation of four signs. Top-left: P, Top-right: 8, Bottom-left: Why, Bottom-right: Go.

One of the primary hurdles to overcome was that some signs involved placing the hands over the face, which would interfere with the keypoint recognition on MediaPipe's end; phrases such as "I'm sad," for example, involve the signer's hands fully blocking the facial landmarks and would prevent tracking for a short time after that, causing errors when saving the video's keypoints to the .pkl file. This was remedied by creating a version of the sign that did not overlap the keypoints to prevent errors, but also these signs could not be too similar to previously existing signs. For the "sad" example, the solution was to move the hands slightly out of the way of the face to register as many of the landmarks as possible. This technique was applied to several other signs. Fig. 4 exemplifies four signs that were captured and animated in our framework.

3.3.2 Utilizing VPython for Motion Captured Animation

VPython is a Python library that is made for creating scientific 3-D animations and visualization [17]. In *VPython*, a user can create different shapes and positions and add angles by utilizing several different built-in functions [16]. In this study, we used VPython to translate MediaPipe's motion data into a fully animated 3D model of the landmarks in the face and hands, in addition to adding lines between many of the landmarks in the fingers. For animation use, we decided to use all landmarks in MediaPipe's left and right-hand landmark groups. However, we only used a handful of the facial landmarks as using all

478 landmarks leaves the animation cluttered and hard to understand. In total, we used 146 landmarks with varied positions to properly show all of the major facial features. We did not use any of the pose landmarks in our study.

During the runtime, after a word in the queue is being used for translation, we will retrieve the movement data that is correlated with the sign and then animate each frame of the data by taking each keypoint's position in 3-dimensional space and assigning its position to a sphere in a VPython scene, which is the space where an animation takes place. This will repeat for each landmark from MediaPipe that was recorded and is being used in the animation program in addition to being repeated for each frame that is in a specific animation. In addition, several lines were added between many of the landmarks to give a skeletal look to the hand (see Fig. 4)

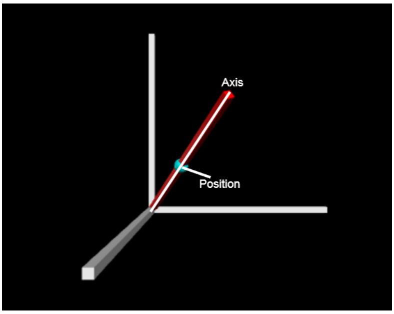


Figure 5: VPython Box Attributes

Creating the movement for these connecting 'bones' is more complicated than the keypoints. VPython does not have a direct way to calculate and direct shapes into different angles. Specifically, the box shape has many properties that make orienting it 3D space difficult. A box's position is decided by the center, meaning changing its position means changing where the center point is. This means when deciding where to place the box we need to use the midpoint to place the box between the two keypoints that we intend to connect with a bone. The position of a bone connecting landmarks at positions (x_s, y_s, z_s) and (x_d, y_d, z_d) is $(\frac{x_s+x_d}{2}, \frac{y_s+y_d}{2}, \frac{z_s+z_d}{2})$.

Determining the position of an object only places the object at the desired space, but it does not angle, or specify the size of the object. This is determined by the axis and the length, width, and height attributes.

The width and height are predetermined before runtime as these are not changed by axis and need to stay consistent to look natural and easy to understand. The axis of a box object is a line that extends outward from a point in the middle of one of the end faces through the middle and to the opposite end face. The axis can be used to determine length and direction of a box object.

Using the landmarks at positions $(x_s y_s, z_s)$, and (x_d, y_d, z_{dest}) $(x_d - x_s, y_d - y_s, z_d - z_s)$. Fig. 5 illustrates the axis and other attributes of the box in a visual diagram.

4 Results

When executed, the program displays the set of recorded words from the .pkl file, where users can string together phrases that they would like to see signed out. We recorded a total of 87 word signs, as well as signs for A-Z and 0-10, that can be played back for viewing at 30fps to match the recorded frame rate for consistency and fluidity. Currently, the input only takes the raw words and not proper sentences, so indefinite articles like "a," "an," "the," are omitted from the input phrase, and compound phrases that have a single sign require underscores in between words. Support for punctuation is also omitted for simplicity, but will be added in as we develop our natural language processing model to specify tone and help with context. An example of an input that we would use for testing and demo purposes would look something like:

Input Sentence to be translated: $\{excuse\_me \; where \; bathroom\}$

Currently, the output will play the animation in a *localhost* window. The animation skeleton would look something like the following when played back (see Fig. 6):

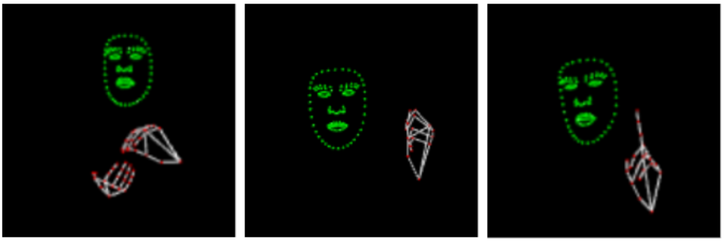


Figure 6: The signs for (from left to right) "excuse me," "bathroom," and "where" from the .pkl file.

5 Discussion

In this study, we have developed a software framework that can transform an English token into ASL modular language using MediaPipe, a object detection library that is able to capture over 543 landmarks on the face, hands and body.

We used these features to capture the position of these landmarks and use them to create animations in VPython, a 3-dimensional graphics library.

This is a groundbreaking start to creating a fully functioning two-way ASL translation software. However, there is still much work to be done. English and ASL have very different grammar and sentence structures [10]. To turn our program into something usable for a general audience, we need to translate not only the words that are being said but also the grammar in which it is being said in. We plan to approach this by utilizing Natural Language Processing (NLP) and gathering a lexical ASL dataset from scratch, which would be one of, if not the first of its kind. We are also aiming to gather data for an ASL-to-English translation program and run both simultaneously for a full translation experience.

As for the STT model that we used in our framework developing, it is very good at dealing with accents [7]; however, it is very slow when writing to the output .txt file. This makes conversation difficult because each participant involved must wait for the processing of the data prior to starting a new line of dialogue. Additionally, multiple voices during the same input period can cause errors in the API's ability to parse out some parts of speech [19]. This can be remedied with multithreading and further optimizations that we would have to make to the base model, which is currently in development.

6 Conclusion

In this study, we have created a revolutionary machine translation tool that can transform English tokens that are gathered via a Speech-to-Text model into American Sign Language. We discussed the previous works in ASL machine translation and how our work differs from what was done before. We were able to take landmarks generated by the MediaPipe Python library and save the positional data to later be used in real time animation using the VPython 3D graphics library. We were able to generate both the left and right hands in addition to the face. Picking out 146 of the 478 facial keypoints to best illustrate the emotions in the facial features while remaining clear enough to be read by the user.

We were able to record 100 signs in total that are able to be used including letters from A to Z, numbers from 1 to 10 and many of the most commonly used signs in ASL. We are also able to use the alphanumeric signs to spell any spoken words that are not currently in our dictionary, which allows even the most uncommon words the ability to be spoken.

Although our work is a massive forward for ASL machine translation, we still fall short of being able to translate between grammars as ASL and English have very different grammatical structures. We do eventually plan on gathering data for, and creating a Natural Language Processing model to be able to accomplish this task.

We also need to accomplish a way to turn phrases like "thank you" and "how are you?" into a single compound word when querying from out dictionary in order to access the correct signs for the phrases we wish to use. In addition to language processing tasks, we also need to increase the processing speed of our Speech to Text functionality as it is currently very slow during runtime. Despite these limitations, we have made a framework that has never been accomplished to the scale that we have produced. This program will give both students and educators the ability to better communicate with Deaf and Hard of Hearing (HoH) people while still retaining the emotional tone of language that textual communication cannot.

References

[1] Md Amiruzzaman et al. "Classifying crime places by neighborhood visual appearance and police geonarratives: A machine learning approach". In: *Journal of computational social science* (2021), pp. 1–25.

[2] Vijayendra D Avina et al. "An AI-Based Framework for Translating American Sign Language to English and Vice Versa". In: *Information* 14.10 (2023), p. 569.

[3] Hicham Boudlal, Mohammed Serrhini, and Ahmed Tahiri. "A novel approach for simultaneous human activity recognition and pose estimation via skeleton-based leveraging WiFi CSI with YOLOv8 and mediapipe frameworks". In: *Signal, Image and Video Processing* (2024), pp. 1–17.

[4] Joanna E Cannon, Laura D Fredrick, and Susan R Easterbrooks. "Vocabulary instruction through books read in American Sign Language for English-language learners with hearing loss". In: *Communication Disorders Quarterly* 31.2 (2010), pp. 98–112.

[5] M Bellen Enrique III et al. "Integrated Visual-Based ASL Captioning in Videoconferencing Using CNN". In: *TENCON 2022-2022 IEEE Region 10 Conference (TENCON)*. IEEE. 2022, pp. 1–6.

[6] Yona Falinie Abdul Gaus and Farrah Wong. "Hidden Markov Model-Based gesture recognition with overlapping hand-head/hand-hand estimated using Kalman Filter". In: *2012 Third International Conference on Intelligent Systems Modelling and Simulation*. IEEE. 2012, pp. 262–267.

[7] Katrina N Haug and Perry D Klein. "The effect of speech-to-text technology on learning a writing strategy". In: *Reading & Writing Quarterly* 34.1 (2018), pp. 47–62.

[8] Jie Huang et al. “Sign language recognition using 3d convolutional neural networks”. In: *2015 IEEE international conference on multimedia and expo (ICME)*. IEEE. 2015, pp. 1–6.

[9] Ahmed Kasapbaşi et al. “DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals”. In: *Computer Methods and Programs in Biomedicine Update* 2 (2022), p. 100048.

[10] Scott K Liddell. *American sign language syntax*. Vol. 52. Walter de Gruyter GmbH & Co KG, 2021.

[11] Camillo Lugaresi et al. *MediaPipe: A Framework for Building Perception Pipelines*. 2019. arXiv: 1906.08172 [cs.DC].

[12] Ross E Mitchell and Travas A Young. “How many people use sign language? A national health survey-based estimate”. In: *Journal of Deaf Studies and Deaf Education* 28.1 (2023), pp. 1–6.

[13] Ross E Mitchell et al. “How many people use ASL in the United States? Why estimates need updating”. In: *Sign Language Studies* 6.3 (2006), pp. 306–335.

[14] Qutaishat Munib et al. “American sign language (ASL) recognition based on Hough transform and neural networks”. In: *Expert systems with Applications* 32.1 (2007), pp. 24–37.

[15] Elissa L Newport and Richard P Meier. “The acquisition of American sign language”. In: *The crosslinguistic study of language acquisition*. Psychology Press, 2017, pp. 881–938.

[16] David Scherer. `https://vpython.org/`. Accessed: 5-2-2024. 2000.

[17] Steve Spicklemire. *Visualizing Quantum Mechanics with Python*. CRC Press, 2024.

[18] Thad Starner, Joshua Weaver, and Alex Pentland. “Real-time american sign language recognition using desk and wearable computer based video”. In: *IEEE Transactions on pattern analysis and machine intelligence* 20.12 (1998), pp. 1371–1375.

[19] Andreas Stolcke et al. “Recent innovations in speech-to-text transcription at SRI-ICSI-UW”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.5 (2006), pp. 1729–1744.

[20] Tsung Heng Wu et al. “Visualizing Routes With AI-Discovered Street-View Patterns”. In: *IEEE Transactions on Computational Social Systems* (2024).

Enhancing Learning of Matrix Transformations through Immersive Virtual and Augmented Reality Interfaces*

Kantida Nanon[1], Margarita Vinnikov[1],
Mathew Schwartz[2], Michael J. Lee[1]
[1]Department of Informatics, [2]School of Art and Design
New Jersey Institute of Technology
Newark, NJ 07102
{kn32, margarita.vinnikov, cadop, mjlee}@njit.edu

Abstract

This study investigated the effectiveness of 3D Augmented Reality (AR) and Virtual Reality (VR)interfaces for teaching affine matrix transformations. Matrix transformations play a critical role in various disciplines (e.g., computer science, game development, robotics, and biomechanics), and traditional teaching methods often fall short of providing an intuitive understanding of these abstract concepts. Having a tool that can generate visual representations of matrix transformations in a more intuitive manner can connect mathematical operations with visual outcomes. Participants engaged in matrix transformation activities using 2D, 3D-AR, and 3D-VR calculators. Results showed significant improvements in achievement scores for the 3D groups, indicating the potential of immersive technologies for enhancing mathematical spatial understanding. While comfort was slightly impacted, overall satisfaction with the 3D groups were high. This research contributes insights into the evolving landscape of educational technology, emphasizing the promising role of 3D interfaces in facilitating a deeper comprehension of complex mathematical spatial concepts.

1 Introduction and Related Work

Matrix transformations—which are crucial in disciplines like robotics, biomechanics, and graphics—form the foundation for understanding spatial relationships [2, 4, 27]. Traditional learning methods, relying on pen-and-paper calculations and 2D visualizations [5], have been criticized for lacking engagement and presenting challenges in comprehending transformation matrices and related concepts such as coordinate frames [7]. However, recent advancements in Virtual Reality (VR) and Augmented Reality (AR) technologies provide exciting opportunities to revolutionize the learning experience, enhance understanding [12], and offer unique possibilities to transform learning environments by providing immersive, interactive, and engaging settings for learners [20]. Numerous studies have explored the application of immersive technologies across various educational domains, highlighting their potential to enhance learning outcomes and learner engagement [13, 17, 28]. Better understanding and utilization of immersive educational technologies continue to be an important domain to explore to effectively teach and engage learners.

Previous work has explored the effectiveness of VR and AR in teaching complex concepts. For example, studies show that students demonstrate positive learning gains using VR simulations to learn binary counting [19], list sorting algorithms [18], and chemistry concepts [15]. Similarly, students using an AR biology application that facilitated interaction with virtual organisms and explore anatomical structures revealed improved spatial reasoning skills and a deeper understanding of biological concepts [29]. Immersive technologies have also been applied in various professional domains. In medicine, VR simulations train students in surgical procedures, enhancing technical skills and confidence [25]. In architecture and design, AR tools facilitate spatial visualization and prototyping, effectively enhancing students' spatial cognition and design thinking abilities [8]. Therefore, it is essential to further investigate the integration of VR and AR in education and professional training to fully understand how these technologies benefit learners, highlighting their potential to transform traditional learning and skill acquisition methods.

The potential benefits of immersive technologies in education are vast, but more research is necessary to fully understand their long-term effects on learning outcomes, learner motivation, and skill transferability. Immersive technologies like VR and AR have the potential to revolutionize education, providing engaging and interactive learning experiences successfully applied across various disciplines [3, 6]. Despite challenges, these technologies offer promising opportunities for educators to create dynamic learning environments [9]. The integration of VR and AR with matrix transformations (i.e., the topic of this study) presents a promising approach to enhance the learning experience [30]. In the context of matrix transformations, kinematics (which VR and AR can include) is essential for describing the movement of objects and understanding how they change position or orientation over time [1, 21], which can be crucial for fields such as robotics, where precise movement control is essential [22]. Therefore, incorporating kinematics into VR and AR applications can significantly enhance learning experiences [18], particularly in fields where understanding movement and transformations is critical.

In the realm of immersive technologies like VR and AR, understanding kinematics becomes even more vital. These technologies often involve the simulation of movement and interactions within a 3D space. For instance, in a VR environment, users may manipulate objects or navigate through a virtual world, and a robust understanding of kinematics is necessary to ensure realistic and accurate movement representations. The interactive and immersive nature of VR and AR allows learners to visualize and manipulate objects in 3D, enabling a deeper understanding of complex concepts.

The study described in this paper aims to investigate the effectiveness of immersive VR and AR interfaces in learning matrix transformations. By integrating mathematical concepts with interactive tools, our study utilizes spatial manipulation and transformation operations on an axis to enhance understanding of homogeneous transformations. We conducted a comparative analysis to evaluate the recall and transfer of knowledge across conventional methods and matrix-based modalities. The application allowed users to comprehend the transformation matrix, kinematics, and transition patterns between coordinate frames. New interfaces, developed and evaluated, allow users to visualize and interact with matrix transformations in 3D, with effectiveness compared to traditional methods like pen-and-paper calculations and 2D visual representations. The study seeks insights into how learners effectively recall and transfer knowledge through different modalities, emphasizing the potential to create interactive tools stimulating visual and spatial cognition. Additionally, the study explores the potential of immersive VR and AR interfaces in facilitating the learning of matrix transformations. By leveraging the benefits of visual and spatial cognition, the aim was to create an interactive and engaging learning experience empowering learners to understand the complexity of matrix transformations and apply them effectively in diverse areas. Participants engaged in matrix transformation problems, with their performance closely observed in terms of completion time, task accuracy, and overall usability. The findings of this study can inform the design and development of future learning tools for matrix transformations and related areas. By combining the power of immersive technologies with mathematical knowledge, our aim was to provide learners with an interactive and engaging learning experience that enhances their understanding and proficiency in matrix transformations. This research significantly contributes to the growing body of knowledge on the application of VR and AR in education, highlighting the potential for these technologies to revolutionize how complex concepts are taught and learned.

2 Method

The study employed a comparative design, incorporating three distinct participant groups: a control group, an experimental group using a 3D-VR calculator, and another experimental group using a 3D-AR calculator. All conditions utilized a cube as the primary instrument for visualizing position and orientation, a shape commonly used in 3D modeling software for visualizing orientation and perspective [11]. The control group utilized traditional learning resources such as class lectures and online materials. In contrast, the experimental groups interacted with an advanced VR /AR interface, which included a specialized 3D calculator for inputting and manipulating transformation matrices. The primary focus of observation was on three aspects: 1) the *time* taken by participants to complete the tasks, 2) the *accuracy* of their solutions, and 3) the *usability* of the tools they were provided with that proxy participant engagement, tool usability, and the overall impact on learning outcomes.

2.1 Research Question & Hypotheses

We explore the following research question and two research hypotheses: *Does learning using different modalities (2D traditional vs. 3D-VR vs. 3D-AR) affect learners' recall and ability to transfer matrix transformation knowledge?*

H1: The learners' average test scores across learning groups are different.
H2: The learners' average usability scores are different across learning groups.

2.2 Participants

Thirty adults were recruited from the university campus and surrounding communities using convenience sampling (see Table 1 for demographics). The sample included college students enrolled in a math class, representing the broader population of students taking similar courses, especially metric lesson.

The experimenter provided participants with a brief statement of the study and a consent form before conducting the experiment, all of which were approved by our institution's Institutional Review Boards (IRB). In summary, 16 males (53.3%) and 14 females (46.70%) participated in our study. Ages ranged from 18 to 35 years old (median 25). 40% of the participants had prior experience with VR technology before the experiment, while 23.33% of the participants had prior experience with AR technology.

Table 1: Our study participants' demographic information.

	Variable	Counts	Percent (%)
Gender	Male	16	54.55
	Female	14	45.45
Disability condition	No	28	93.33
	Yes	2	6.67
Age	18-24	17	56.67
	25-34	13	43.33
Highest education level	College/University	20	66.67
	High School	8	26.67
	Graduate School	2	6.66
Participants' experience	Gaming	13	43.33
	VR	12	40.00
	Web programming	8	26.67
	Unity/Unreal	7	23.33
	AR	7	23.33
	Game programming	6	20.00

2.3 Stimuli

We developed 3D-AR and 3D-VR applications in the form of a calculator to enable users to visualize and interact with matrix transformations in 3D. We designed these as mobile applications for tablet devices and implemented them in the Unity game engine. Figure 1 illustrates the User Interface (UI) of the 3D calculator application. The UI of the 3D calculator application consisted of the matrix calculator panel, the cube object, and its frame transformation (x, y, z axis) in 3 different colors. User could select the listed utilities in the application to transform the object based on a matrix function. The interaction method was based on a set of utilities that describe rotations, translations, matrices, and axis definitions. The utilities were arranged in left to right order: x, y, and z; there was no specified order of operations. The application workflow was top to bottom. Users could drag-and-drop and add the desired operations to the slots. Users could then enter the number (in degrees or radians) to view values in various formats for the active utility and final transformation matrix value. There are numerous ways users could interact with the utility. In the case of a rotation, the user could choose between entering degrees and radians. The single input field updates the values inside the rotation matrix. To view the matrix in terms of an expression, where cosine and sine are present with the specified radians, the user could also select the *Expression* button, whereas the *Value* button displays the matrix values. Three lines indicated the frame transformations of the cube object: the red line represented the x-axis, the green line represented the y-axis, and the blue line represented the z-axis.

In the application interface (Figure 1), the calculator panel was positioned on the right side of the screen, featuring interactive controls for numerical input

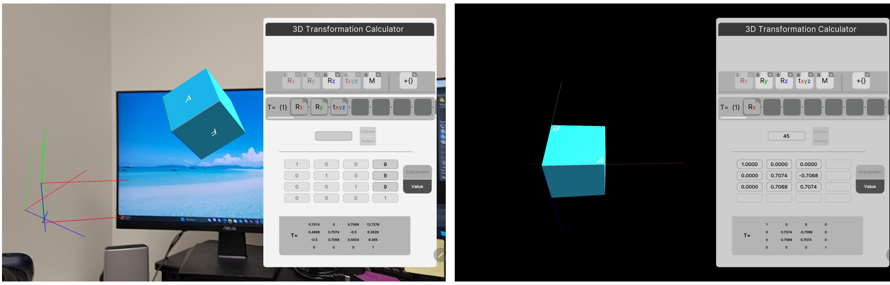


Figure 1: The 3D-AR (*left*) and 3D-VR (*right*) calculator overlay the mapping cube on the real world or virtual environment, respectively.

and calculations. The left side displayed a blue cube, with each side labeled to indicate its specific orientation. Upon launching the application, users could select a cube, and the application dynamically positions the chosen cube to correspond with the inputted numerical value. This interactive setup ensureed a visually intuitive experience, allowing users to associate the calculated number with the spatial arrangement of the labeled cube sides. There were three main areas of interest in the calculator panel (Figure 2). The upper section included utilities, operators, and an axis definition that could be dragged into the empty dark gray slots in the middle section of the interface. The middle of the display panel presented the currently active utility, allowing users to enter values and view specific transformations (local transformation matrix value). The bottom area showed the final transformation (global transformation matrix value) from the combination of active utilities. We built on our previous work to create the UI [16] and logic of the application interface [24].

2.4 Experimental Procedure

Participants started the experiment with the demographic questionnaire (see Table 1). Then, one of the researchers demonstrated the use of each tool (2D traditional, 3D-AR, or 3D-VR), which lasted approximately 3 minutes. Next, participants watched a transformation matrix lesson on a computer screen with a headset for 15 minutes. After finishing this lesson, the experimenters divided participants into three groups to solve the matrix transformation problems, completing 10 activities (see Figure 3), using the assigned tool. The control group consisted of traditional methods of learning matrix transformations, including class lecture notes, pencils, and quiz activity paper. On the other hand, the experimental groups consisted of VR /AR interfaces that provide 3D calculators for user input.

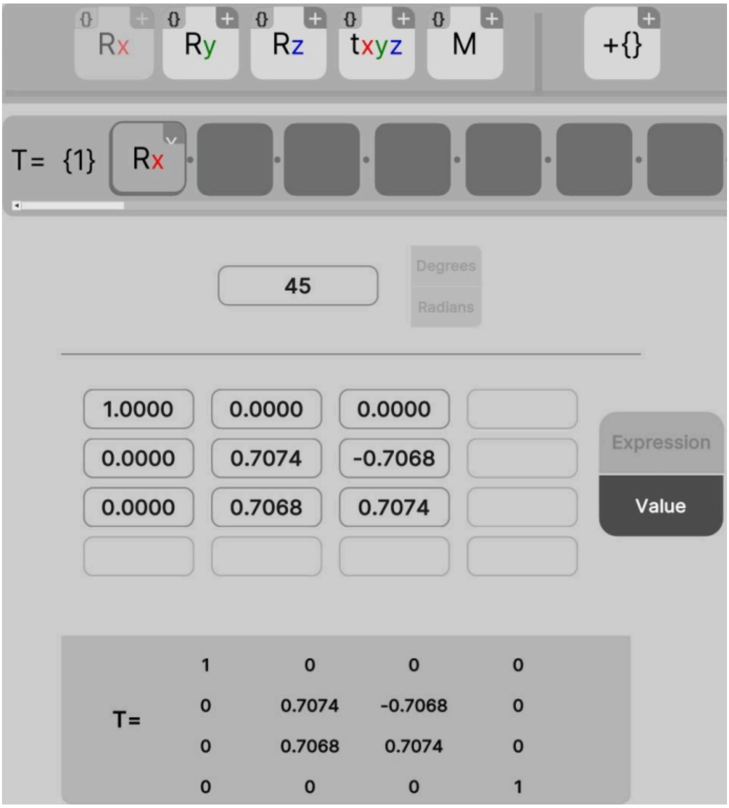


Figure 2: The 3D calculator application showing the matrix calculator panel, the cube object, and its frame transformation (x, y, z axis) in 3 different colors.

The experimental group used the 3D calculator to find the global transformation metric by manipulating the cube in space. Figure 1 shows the UI of the 3D-AR and 3D-VR calculators. The quiz for the control group (using the traditional 2D method) contained ten 2×2-matrix problems, while the quiz for the experimental group (using the 3D application) contained ten 3×3-matrix problems. We measured participants' task completion time and task accuracy. After completing the quiz activity, all participants completed a usability questionnaire to evaluate their engagement, satisfaction, ease of use, efficiency, and effectiveness. All questionnaires were completed within five minutes.

Activity Directions:

- Open the ConvApp3D application.
- Move the blue square following the transformations in the table below.
- Fill up the “Matrix” and “Transformation Matrix” of each transformation in the below table.

No	Transformation	Local Transformation Matrix	Global Transformation Matrix
1	Reflect on the x-axis	$M_{ref_x} = \begin{bmatrix} ..... & ..... & ..... \\ ..... & ..... & ..... \\ ..... & ..... & ..... \end{bmatrix}$	$M_{ref_x} = \begin{bmatrix} ..... & ..... & ..... \\ ..... & ..... & ..... \\ ..... & ..... & ..... \end{bmatrix}$
2	Reflect on the y-axis	$M_{ref_y} = \begin{bmatrix} ..... & ..... & ..... \\ ..... & ..... & ..... \\ ..... & ..... & ..... \end{bmatrix}$	$M_{(ref_x)(ref_y)} = \begin{bmatrix} ..... & ..... & ..... \\ ..... & ..... & ..... \\ ..... & ..... & ..... \end{bmatrix}$

Figure 3: The Matrix Transformation activity involves 10 questions requiring participants to fill in local and global transformation matrix numbers.

3 Results

Figure 4 presents a boxplot of participants' test scores between control (2D) and experimental groups (3D-AR and 3D-VR). The control group had the lowest average score (5), while the experimental groups with 3D-AR and 3D-VR had an average score of 9 and 8, respectively. We used Welch's test for one-way ANOVA to compare the means of the three groups, which showed that the $p-value$ was less than α=0.05 (see Table 2). This suggests that there was a significant difference between the group means between the control and experimental groups. To further examine the differences, we used the Fisher

Table 2: Participants' test scores between control and experimental groups.

Source	**DF**	**Adj SS**	**Adj MS**	**F-Value**	**p-Value**
Groups	2	83.72	41.86	14.69	< 0.05
Error	27	76.95	2.85		
Total	29	160.67			

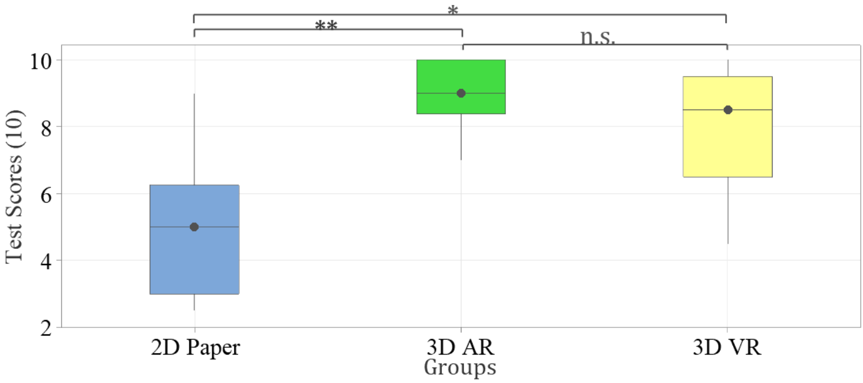


Figure 4: Participants' test scores in the control (2D) and experimental groups (3D-AR and 3D-VR). * for $p < 0.05$, ** for $p < 0.01$, n.s. for not significant.

LSD method (see Table 3) to determine which specific group means differ. The means of the 3D-VR and 3D-AR groups were not significantly different from each other, but both were significantly different from the 2D group.

According to the Fisher individual tests, the means of achievement scores for both experimental groups (3D-AR and 3D-VR) were significantly different from the control group (2D). However, there was no significant difference between the means of 3D-VR and 3D-AR groups. Our results suggest that both the 3D-AR and 3D-VR tools tend to promise higher achievement scores.

3.1 Physical Comfort Score

The boxplot of participants' physical comfort score between groups is shown in Figure 5. The control group has the highest mean score of 3.40. The 3D-AR group has a lower mean score of 1.50 compared to the control group. The 3D-VR group has the lowest mean score of 2.20 among the three groups, with a standard error of 0.68. We used Welch's test for one-way ANOVA to determine if there were significant differences among the group means. We conducted the

Table 3: The results of Fisher Individual Tests for Differences of Means.

Methods	Means	SE	95% CI	t-Value	p-Value
3DAR - 2D	3.95	0.76	$(2.07, 5.82)$	5.23	< 0.05
3DVR - 2D	2.90	0.76	$(1.03, 4.77)$	3.84	< 0.05
3DVR - 3DAR	-1.05	0.76	$(-2.92, 0.82)$	-1.39	0.36

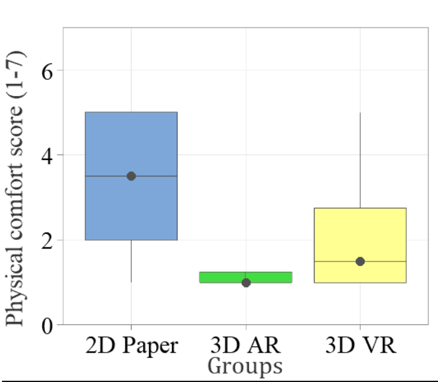


Figure 5: Participants' physical comfort score between groups, with scores of 1–7 (low–high) comfort.

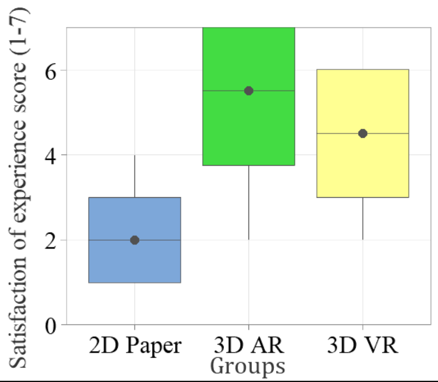


Figure 6: Participants' satisfaction of experience score between groups, with scores of 1–7 (low–high) satisfaction.

test and found a significant result ($F = 3.99, p < 0.05$), indicating differences in the means of the groups. These findings suggest that there seems to be a difference in the means of the physical comfort scores between the groups. The control group has the highest mean physical comfort score, followed by the 3D-AR group, and the 3D-VR group has the lowest mean score. However, the analysis results suggest that participants who experienced the 3D-AR and 3D-VR calculators reported higher satisfaction scores than those who used the traditional 2D paper-based approach. Figure 6 presents participants' satisfaction of experience score between groups. We conducted ANOVA analyses to examine the satisfaction of experience scores among the different groups, revealing a significant effect of the group variable on the satisfaction scores ($F = 11.64, p < 0.05$), indicating differences in the means of the groups. Furthermore, the control group's mean satisfaction score was 2.00, with a standard deviation of 1.05. The 3D-AR group had a mean score of 5.10, with a standard deviation of 1.79. The 3D-VR group had a slightly lower mean score of 4.40, with a standard deviation of 1.58. We conducted the Tukey post-hoc pairwise comparisons to examine the differences between pairs. The results indicated that the 3D-AR and 3D-VR groups had significantly higher mean satisfaction scores than the control group. However, there was no significant difference between the mean satisfaction scores of the 3D-AR and 3D-VR groups. These findings highlight the potential of immersive technologies to enhance users' overall satisfaction and engagement in educational settings.

4 Discussion

The results of this study provide valuable insights into the effectiveness of different learning tools and interfaces for understanding the relationship between matrix transformations and their spatial relevance. The findings indicate that the 3D-AR and 3D-VR calculators significantly impacted participants' achievement scores compared to the traditional 2D methods. This suggests that the immersive nature of the 3D interfaces, coupled with the calculator's interactive and visual elements, positively influenced participants' learning outcomes. The higher average test scores observed in the experimental groups using the 3D-AR and 3D-VR calculators can be attributed to several factors. It agrees with previous work by Singh, Tuli, & Mantri's work that 3D representation of matrix transformations provided a more intuitive and immersive learning experience, allowing users to visualize and manipulate the matrices more realistically and interactively [26]. Our VR and AR applications enhanced learners' spatial understanding and mental visualization of matrix transformations, which are crucial concepts in mathematics and related fields.

The interactive nature of the 3D calculators allowed participants to actively engage with the learning material. Our tool provides a hands-on approach to solving matrix transformation problems, and the calculators fostered a deeper level of understanding and application of the concepts. Unlike Kaneto & Komuro's previous work finding that participants preferred traditional implementations [10], our study shows that users preferred the VR version. This could be because our tool's design or problem being solved facilitated such outcomes. The drag-and-drop functionalities, input fields, and visual feedback provided by the calculators facilitated an iterative problem-solving process, enabling participants to explore different transformations and evaluate their results in real-time. It is important to highlight that the control group, which relied on traditional 2D methods such as lecture notes and pencil-paper activities, demonstrated the lowest average test score. This highlights the limitations of conventional learning approaches when it comes to complex mathematical concepts like matrix transformations. The static and abstract nature of 2D representations may prevent learners' ability to understand the spatial relationships and transformations inherent in matrices.

There was no significant difference in achievement scores between the 3D-AR and 3D-VR groups. This suggests that both AR and VR interfaces can be equally effective in facilitating learning outcomes for transformation metric knowledge. The similarity in performance may be attributed to the shared characteristics of both interfaces, such as their immersive nature, interactive elements, and ability to provide realistic visualizations. Future research could further explore the specific advantages and differences between AR and VR in the context of mathematics education. Furthermore, the analysis of satisfaction of experience scores revealed significant differences between the groups. The 3D-AR and 3D-VR groups reported significantly higher satisfaction scores than the control group. This indicates that the immersive nature of the AR and VR technologies positively influenced participants' overall experience and perception of the learning environment. The ability to interact with 3D objects and navigate virtual spaces likely created a more engaging and enjoyable learning experience for the participants, resulting in higher satisfaction levels.

It is worth noting the comfort scores among the different approaches. The analysis revealed that the control group had a higher mean comfort score than the 3D-AR and 3D-VR groups. This finding raises important considerations when evaluating the use of immersive technologies in educational settings. One possible explanation for the higher comfort score in the 2D paper group was the familiarity and simplicity of the traditional paper-based approach. Participants in this group may have been more accustomed to learning through printed materials and felt more at ease with the conventional learning method. The absence of additional technological equipment and the physical interaction with tangible paper materials may have contributed to a sense of comfort and ease of use. On the other hand, the 3D-AR and 3D-VR approaches introduced novel and immersive experiences that could potentially be more demanding or unfamiliar to participants. Using AR and VR technologies requires headsets, controllers, or other devices that can be cumbersome or create a sense of disorientation for some individuals. This could have influenced the comfort scores and contributed to a slightly lower perceived comfort level in the immersive technology groups. However, the comfort scores alone do not necessarily indicate the overall effectiveness or quality of the learning experience. While the 2D paper group reported higher comfort scores, the achievement and satisfaction of experience scores were significantly higher in both 3D groups. This suggests that although participants in the immersive technology groups may have perceived a slightly lower comfort level, they still benefited from enhanced learning outcomes and greater satisfaction with the learning experience.

5 Limitations and Future Work

While our participants represented our tool's target demographic, we recruited them from a university campus and surrounding communities, potentially leading to a sample that is not fully representative of the broader population. A more diverse and randomly selected sample would enhance the study's external validity. Another limitation is the relatively short duration participants were exposed to the learning materials and tools. This limited exposure may not fully capture the long-term effects or sustainability of the observed learning outcomes. Future studies should consider incorporating a longer intervention period or follow-up assessments to examine the strength of the effects and investigate the potential long-term benefits of using 3D interfaces for learning matrix transformations. In summary, while the current study provides valuable insights into the effectiveness of 3D interfaces for learning matrix transformations, future research needs to address some limitations. By expanding the sample size, extending the intervention duration, incorporating qualitative measures, and exploring new research directions (e.g., training educators to use AR/VR tools to teach course materials [14]), researchers can further enhance our understanding of the potential and limitations of immersive technologies in education. Experience with long-term use of AR/VR devices may impact the results, even with higher experience scores. With continuing improvements to AR/VR technologies that provide extremely lightweight glasses with head-tracking technologies (e.g., Nreal Light [23]), replicating this study in the future may provide insights into the importance of comfort in learning tasks.

6 Conclusion

This study investigated the effectiveness of different learning tools and interfaces on engagement, usability, and learning outcomes for matrix transformation knowledge. The analysis of the participants' test scores revealed significant differences among the group means. The control group, utilizing traditional 2D methods, exhibited the lowest average test score. In contrast, the experimental groups using the 3D-AR and 3D-VR calculators demonstrated higher average scores. Further examination shows that both the 3D-AR and 3D-VR tools significantly outperformed the traditional 2D methods in terms of achievement scores. However, there was no significant difference between the 3D-AR and 3D-VR groups, indicating that both interfaces were equally effective in facilitating learning outcomes for matrix transformation knowledge. The study supports the hypothesis that the average test and usability scores across modalities are significantly different. Using 3D-AR and 3D-VR calculators resulted in higher achievement scores than traditional 2D methods. These results highlight the potential of immersive technologies in enhancing learning outcomes and engagement in the context of matrix transformation knowledge. In conclusion, this study's findings underscore the potential of immersive technologies, specifically 3D interfaces, to enhance learning outcomes and engagement in mathematics education. The results provide empirical evidence supporting integrating such tools into educational settings to promote a deeper understanding and application of mathematical concepts. By leveraging the advantages of 3D visualization, interactivity, and real-time feedback, educators can create more engaging and effective learning environments, particularly in subjects that involve spatial reasoning and visual representations.

7 Acknowledgements

This work was supported in part by the Google Research Scholar Program. Any opinions, findings, conclusions, or recommendations are those of the authors and do not necessarily reflect the views of Google Research or other parties.

References

[1] K. Anjyo and H. Ochiai. *Mathematical basics of motion and deformation in computer graphics*. Springer Nature, 2022.

[2] C. Brown and B. Jones. Matrix transformations in biomechanics: A comprehensive review. *Journal of Biomechanics*, 45(2):89–104, 2018.

[3] M. Chen. Virtual reality in education: A review of the research. *Journal of Educational Technology & Society*, 23(4):133–148, 2020.

[4] W. Chen and L. Wang. Matrix transformations in computer graphics: Techniques and applications. *ACM Transactions on Graphics*, 36(4):67–89, 2017.

[5] R. Davidson and S. Lee. Teaching matrix transformations: Challenges and opportunities. *Mathematics Education Research Journal*, 27(2):165–182, 2015.

[6] J. Dibbern and A. Arvanitis. Augmented reality in education: Current technologies and the potential for education. In *Handbook of Research on Immersive Digital Learning Environments*, pages 170–186. IGI Global, 2019.

[7] J. Dick and M. Childrey. Enhancing understanding of transformation matrices. *The Mathematics Teacher*, 105(8):622–626, 2012.

[8] W. Hwang and S. Chang. Enhancing spatial cognition through ar tools in design education. *International Journal of Architectural Research*, 12(2):124–139, 2018.

[9] L. Johnson, R. Smith, H. Willis, A. Levine, and K. Haywood. NMC/CoSN Horizon Report, 2019.

[10] Y. Kaneto and T. Komuro. Space-sharing ar interaction on multiple mobile devices with a depth camera. In *VR*, pages 197–198, 2016.

[11] A. Khan, I. Mordatch, G. Fitzmaurice, J. Matejka, and G. Kurtenbach. Viewcube: a 3D orientation indicator and controller. In *I3D*, pages 17–25, 2008.

[12] S. Kim and J. Lee. The impact of virtual reality on learning matrix transformations. *International Journal of Educational Technology*, 23(4):123–145, 2019.

[13] M.J. Lee, A. Spryszynski, and E. Nersesian. Personalizing VR educational tools for English language learners. In *IUI Workshops*, volume 19. ACM, 2019.

[14] E. Nersesian, A. Spryszynski, T. Espiritu, and M.J. Lee. Pre-college computer science initiative for augmented and virtual reality development. In *ISEC*, pages 1–7. IEEE, 2020.

[15] E. Nersesian, A. Spryszynski, and M.J. Lee. Integration of virtual reality in secondary stem education. In *ISEC*, pages 83–90. IEEE, 2019.

[16] E. Nersesian, A. Spryszynski, J. Ross, and M.J. Lee. Usability evaluation on artifact affordances in collaborative virtual reality. In *ISEC*, pages 83–90. IEEE, 2019.

[17] E. Nersesian, A. Spryszynski, U. Thompson, and M.J. Lee. Encompassing English language learners in virtual reality. In *AIVR*, pages 200–203. IEEE, 2018.

[18] E. Nersesian, M. Vinnikov, and M.J. Lee. Travel kinematics in virtual reality increases learning efficiency. In *VL/HCC*, pages 1–5. IEEE, 2021.

[19] E. Nersesian, M. Vinnikov, J. Ross-Nersesian, A. Spryszynski, and M.J. Lee. Middle school students learn binary counting using virtual reality. In *ISEC*, pages 1–8. IEEE, 2020.

[20] R. Patel and A. Gupta. Augmented reality as a tool for enhancing understanding of matrix transformations. *International Journal of Computer Science Education*, 12(2):87–102, 2018.

[21] E.L. Pier, C. Walkington, V. Clinton, R. Boncoddo, C. Williams-Pierce, M.W. Alibali, and M.J. Nathan. Embodied truths: How dynamic gestures and speech contribute to mathematical proof practices. *Contemporary Educational Psychology*, 58:44–57, 2019.

[22] L. Pérez, E. Diez, R. Usamentiaga, and D.F. García. Industrial robot control and operator training using virtual reality interfaces. *Computers in Industry*, 109:114–120, 2019.

[23] A. Robertson. Nreal light review: almost a look at the future of AR glasses, 2022. Accessed on January 22, 2024.

[24] M. Schwartz and J. Lee. Buildable transforms for axis manipulation in 3D applications. In *HCIK*, 01 2016.

[25] M. Seymour. VR simulations for surgical training. *Medical Education*, 53(8):654–668, 2019.

[26] G. Singh, N. Tuli, and A. Mantri. Augmented reality based application for linear algebra visualization: Development of interactive learning environment. In *ICACITE*, pages 1400–1404, 2022.

[27] J. Smith and A. Johnson. Applications of matrix transformations in robotics. *Robotics Journal*, 10(3):45–57, 2019.

[28] L. Williams. Exploring the potential of virtual reality in teaching matrix transformations. *Journal of Educational Technology*, 35(3):167–182, 2020.

[29] B. Zhang. Enhancing spatial reasoning in biology education through AR. *Educational Technology Research*, 32(4):78–92, 2019.

[30] Y. Zhang and W. Huang. Virtual reality technology and its applications in education. In *International Conference on Education, Management, Information and Management Society*, pages 373–378. ACM, 2021.

An Ontology for Social Determinants of Education (SDoEd) based on Human-AI Collaborative Approach*

Navya Martin Kollapally[1], James Geller[2],
Patricia Morreale[1], Daehan Kwak[1]
[1]Kean University
Union, NJ 07083
[2]New Jersey Institute of Technology
Newark, NJ 07102
{nmartink, pmorreal, dkwak}@kean.edu, james.geller@njit.edu

Abstract

The use of computational ontologies is well-established in the field of Medical Informatics. The topic of Social Determinants of Health (SDoH) has also received extensive attention. Work at the intersection of ontologies and SDoH has been published. However, a standardized framework for Social Determinants of Education (SDoEd) is lacking. In this paper, we are closing the gap by introducing an SDoEd ontology for creating a precise conceptualization of the interplay between life circumstances of students and their possible educational achievements. The ontology was developed utilizing suggestions from ChatGPT-3.5-010422 and validated using peer-reviewed research articles. The first version of developed ontology was evaluated by human experts in the field of education and validated using standard ontology evaluation software. This version of the SDoEd ontology contains 231 domain concepts, 10 object properties, and 24 data properties.

1 Introduction

According to the US Department of Health and Human Services (HHS), Social Determinants of Health (SDoH) [18] are the conditions in the environment where people are born, live, learn, work, play, and age that affect the quality-of-life outcomes and risks. Education, health, and well-being are intrinsically interconnected. Education profoundly impacts individuals' lives, playing a crucial role in alleviating poverty and diminishing socioeconomic and political disparities. According to a study by the Centers for Disease Control and Prevention [7], high school students who demonstrated higher academic performance showed a greater tendency towards better health-related behaviors and a notably lower prevalence of health-related risk behaviors when compared to students who exhibited poor academic performance. In analogy to SDoH, these factors have variously been referred to as Social Determinants of Education (SDoEd) or simply SDE [3], e.g., lack of access to a high-speed internet connection.

Given the significant influence of education on individuals' lives and its role in addressing poverty and reducing inequalities, it is important to establish an ontology in this domain. It can serve as a comprehensive framework for organizing and representing knowledge related to education and its impact on society. By capturing the relationships, concepts, and interdependencies within the educational landscape, an ontology can facilitate better understanding, analysis, and decision-making.

Computationally, an ontology is a hierarchical structure of concepts, where pairs of concepts are connected by IS-A (generalization) links and semantic links. Concepts may also have their own local attributes. In a diagram, an ontology appears as a nodes-and-links graph. Refer to Figure 2 in the Results Section for an intuition of such a diagram. Bubbles represent concepts, and arrows are IS-A links. General concepts are at the left, and specific concepts are at the right. Notably, some of these determining factors might be circular and mutually reinforcing. For example, bad health will lead to poor school attendance, which could in turn lead to not learning about a healthy lifestyle or not being able to get into college, perpetuating social issues associated with low income and low living standards, closing the cycle by not being able to afford good healthcare.

The risk factors of SDoEd are not restricted to racial and ethnic minorities as they are often income-based, but these populations are at a higher risk compared to their white peers. The research goal of this work is to present an ontology for Social Determinants of Education (SDoEd). A human-AI collaborative approach to concept collection using ChatGPT-3.5-010422 was utilized. Along with the design and development of the ontology, the human expert and software-based evaluation criterion for ensuring consistency and coherence of the ontology are presented in this work.

2 Background

An existing framework for Determinants of Education/Learning involves asking nine inter-dependent questions related to education; the answers to these questions will produce the concepts for School Health Education (SHE) [7]. To create a comprehensive ontology, it is crucial to compile an extensive list of terms and concepts that cover the domain under consideration. When enriching a domain ontology, developers may rely on research articles to gather concepts that expand the ontology's scope and coverage. However, despite a thorough search, the tasks of gathering all relevant concepts and ensuring the ontology's comprehensiveness were challenging. To address this issue, we utilized a Generative Pretrained Transformer (GPT), a language model trained on extensive text datasets. OpenAI's ChatGPT [19], built on the GPT-3.5/4/4o, is a chatbot that utilizes supervised and reinforcement learning techniques to generate human-like responses to natural language prompts and was trained on licensed and publicly available data through 2023.

Rather than relying solely on the concept choices suggested by ChatGPT-3.5-010422, we ensured the validity of the concepts and their relationships by cross-referencing them with published articles from reputable sources such as PubMed Central (PMC) [16], International Journal of Education Research (IJER) [21], and American Educational Research Association (AERA) [6].

3 Methods

We utilized the ontology principles stated by Noy [17] for developing the ontology for SDoEd. Furthermore, we have used the design and evaluation criteria as used in [8, 10, 11].

3.1 Domain and scope of ontology

The scope of Social Determinants of Education (SDoEd) encompasses a wide range of factors that impact student engagement in education and substantiate the existence of an achievement gap. As defined by the American Board of Education, the achievement gap occurs when there is a statistically significant disparity in average scores between different groups of students [5], typically categorized by race/ethnicity or gender. These factors can be influenced by a variety of elements, including political, economic, cultural, and societal factors.

3.2 Enumerate important concepts for developing SDoEd

We retrieved articles and reports available from trustworthy sources by performing a keyword-based search on the web. These sources utilized terms such as "social determinants of education," "role of education in SDoH," "educational disparities," "reasons for achievement gap in education," and "determinants of learning." By analyzing the results, the main categories of SDoEd were identified to achieve comprehensive coverage of relevant domain concepts.

To confirm coverage in terms of concepts and to address any potential gaps, we then utilized ChatGPT-3.5-010422. We used specific prompts such as "Main categories of Social Determinants of Education," "Sub-concepts related to Economic stability that contribute to the achievement gap," "Child concepts associated with Parental factors influencing educational determinants," "Is there an IS-A relationship between factors affecting health and well-being and the parent concept of Social Determinants of Education," and "Does cyberbullying fall under the child concept of technology integration?" These prompts helped us to find more candidate concepts and to clarify the relationships and classifications within the broader context of SDoEd.

Before adding each of the concepts under a main category of the ontology, we validated the IS-A relationships by searching for articles in PMC within the range of 2018-2023, in the IJER, and on the websites of the AERA and the Department of Education. We utilized the advanced query feature of PMC to validate the concept pairs suggested by ChatGPT-3.5-010422.

During the validation of concept pairs from ChatGPT-3.5-010422 in the relevant sources, we encountered new concepts that were not in the output lists from ChatGPT-3.5-010422. Hence in addition to *forward validation*, i.e., validating concept pairs extracted from ChatGPT-3.5-010422 utilizing the target sources, we also performed *backward validation*, i.e., extracting concept pairs from target sources and validating them using ChatGPT-3.5-010422. Figure 1 represents the forward validation in which ChatGPT-3.5-010422 states that "availability of after-school programs" is a child concept of "availability of educational resources."

To validate this concept pair, we used the prompt "how availability of after-school program and educational resources affect social determinants of education" in PMC, IJER, the AERA, and the Department of Education websites. After identifying relevant articles, a concept pair (parent-child concept pair) is either accepted into the SDoEd ontology or rejected. For backward validation while performing a manual review of relevant articles from target sources, new concept pairs may be identified. These concept pairs will be framed as two concepts connected by an IS-A relationship as shown in Figure 1. We prompted ChatGPT with text corresponding to "Does this RDF triple share a valid IS-A relationship?" (RDF is the Semantic Web Resource Description Framework.)

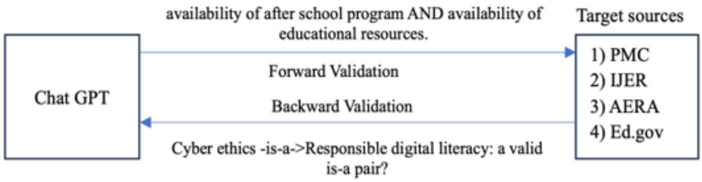


Figure 1: Visualization of forward and backward validation.

3.3 Concept Categorization

After the initial step of concept extraction from ChatGPT-3.5-010422, scholarly articles, and government educational websites, we placed the concepts under six main categories. They are:

a) Cultural factors: This parent concept includes child concepts that significantly shape the educational environment and practices within a particular community or society [12]. They can influence how education is valued, the expectations placed on students, the teaching and learning methods employed, and the overall educational goals and priorities.

b) Economic factors affecting education: This category includes concepts referring to the financial resources and socio-economic conditions that play a significant role in shaping educational opportunities and outcomes [20]. These factors encompass aspects such as funding and resource allocation, socioeconomic disparities, and access to educational resources.

c) Factors influencing health and well-being: The sub-concepts under this main concept encompass a range of elements that impact the physical, mental, and emotional well-being of individuals, which in turn can affect their educational experiences and outcomes.

d) Institutional factors influencing education: Institutional factors encompass the policies, structures, and organizations within the education system that directly or indirectly impact educational outcomes [1].

e) Neighborhood factors influencing education: Neighborhood factors comprise the characteristics and conditions of the local community that surrounds a school, which can significantly influence educational opportunities [15].

f) Parental factors: Parental factors refer to the influences, actions, and characteristics of parents or guardians that significantly impact educational opportunities and outcomes for children [2]. Parental factors play a crucial role in shaping children's educational experiences, motivation, and academic achievements, as parents serve as primary caregivers and key influencers in their children's educational journey.

3.4 Developing an SDoEd Ontology

To implement the SDoEd ontology, we utilized Protégé 5.5, an open-source ontology editor by Stanford University [14]. The SDoEd ontology was developed as a Web Ontology Language (OWL) file. Protégé refers to "concepts" as "classes," and allows adding properties (attributes) and relationships between the classes. The class "Thing" is predefined in Protégé and is used as the root of every ontology created with it. Protégé enables users to edit ontologies in OWL and use a HermiT reasoner to validate the consistency and coherence of the developed ontologies.

3.5 Software-based SDoEd Ontology Evaluation

We performed consistency checking in Protégé by utilizing HermiT [4] Version 1.4.3.456. The HermiT reasoner is based on hyper tableau calculus, which allows it to avoid nondeterministic behaviour exhibited by the tableau calculus that is utilized in FaCT++ [23] and Pellet [22]. Nondeterministic behaviour arises when tableau calculus may have to make arbitrary choices that can lead to inefficiency, and this is avoided by structuring the reasoner process as in hyper tableau calculus.

3.6 Human Expert SDoEd Ontology Evaluation

The main goal of the evaluation of an ontology is to make sure that it is consistent, accurate, and maintains a high level of adaptability and clarity. After evaluating the SDoEd ontology with HermiT for consistency and coherence, we involved two human expert evaluators with extensive experience in the field of education. To understand the percentage agreement between the two evaluators, we utilized Cohen's kappa coefficient (κ). κ is an alternative when the overall accuracy is biased to understand the level of agreement between two evaluators. Both human evaluators (P1 and P2) were provided with the same spreadsheet of 100 randomly selected concept pairs. Among the 100 concept pairs, we provided 10 concept pairs as training samples to present the flavor of the ontology and 90 concept pairs that needed to be evaluated. The spreadsheet contained three kinds of concept pairs: pairs related by IS-A, pairs related as ancestor/grandparent-child, and pairs that were not hierarchically related. P1 and P2 were aware of the fact that the spreadsheet contained these three kinds of concept pairs. Table 1 provides examples of the concept pairs included in the spreadsheet.

Parent	Relationship	Child	Related	Farther away	Reason if unrelated
Availability of educational resources	<--is--a	Availability of tutoring centers	Yes		
Neighborhood safety	<--is--a	Emotional and Social Development in early years	Yes		
Sleep quality of children	<--is--a	Discipline polices	Yes		
Parental factors	<--is--a	Parental educational level	Yes		
Sleep quality of children	<--is--a	Technology integration policies		Yes	Sleep quality of children is unrelated to technology integration

Table 1: A snippet of the concept pairs provided to the human expert.

The 10 samples provided to the evaluators included five of the 'Child' fields filled with 'No' and corresponding reasons were provided in 'Reason if unrelated,' three of the 'Child' fields filled with 'Yes,' and two of the 'Farther away' fields with Yes. For each pair, the fourth column ('Child?' in Table 1) had to be filled in with 'Yes,' if the evaluator felt that the concepts were connected by a parent-child (IS-A) relationship, and 'No,' otherwise. If the answer was 'No,' they were asked to fill in the reason in the column 'Reason if unrelated.' These reasons provided us with directions on how to make improvements to the design of the ontology. The evaluators were asked to fill in the 'Farther away' column with 'Yes,' whenever they felt that the concepts were related by a grandparent or ancestor relationship, i.e., a chain of IS-As. The evaluators were also asked to give reasons in this case. P1 and P2 independently reviewed the pairs, and we used an online κ calculator [13] to identify the level of agreement.

A $\kappa > 0.4$ is considered as moderate agreement, $\kappa > 0.6$ indicates substantial agreement, and $\kappa = 1$ means perfect agreement.

4 Results

We could not locate any preexisting domain ontology specific to SDoEd. This supports the need for our research work. We also used Protégé for evaluation (HermiT). The class metrics returned by Protégé/HermiT are in Table 2.

Metrics	Count
Axioms	498
Logical axiom count	267
Declaration axiom count	231
Class count (not counting Thing)	231

Table 2: Class metrics from Protégé.

Our SDoEd ontology, developed in Protégé, is available as an OWL file in GitHub [9]. In Figure 2 is a snippet of the ontology, visualized using the OWLViz plugin of Protégé.

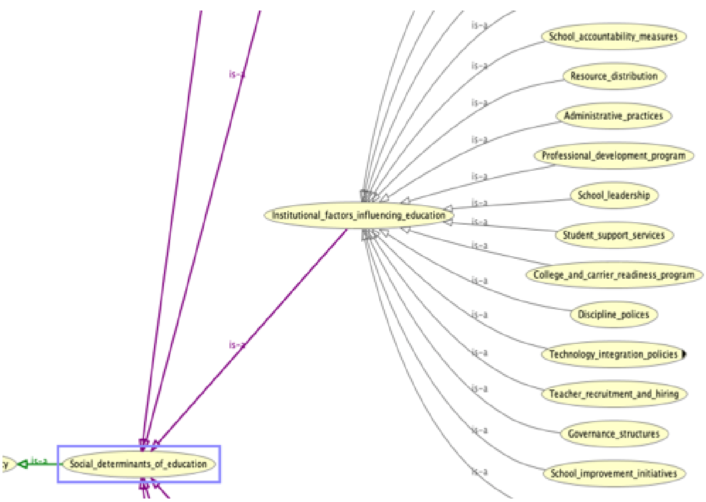


Figure 2: Snippet from OWLViz visualization of SDoEd ontology.

The SDoEd ontology is coherent and consistent as per the HermiT reasoner, also available as a plug-in in Protégé. The confusion matrix for evaluators is given in Table 3, and a κ value of 0.6345 was obtained in the first round of evaluation. This indicates a substantial agreement (83.389%), hence no mitigation plan and no second round were necessary. The κ value represented that the experts were in "substantial agreement" about the domain coverage of the designed ontology. For concept extraction and backward validation, a total of 72 prompts were posed to ChatGPT-3.5-010422. These prompts encompassed a wide range of topics and concepts to ensure comprehensive coverage. The extracted concepts were then validated to ensure accuracy and relevance in the given context.

Confusion matrix	Hierarchical related concept pairs	Unrelated concept pairs
Evaluated as hierarchical related concept pairs by evaluator 1	46	9
Evaluated as hierarchical un-related concept pairs evaluator 1	19	16
Evaluated as hierarchical related concept pairs by evaluator 2	52	0
Evaluated as hierarchical un-related concept pairs by evaluator 2	21	17

Table 3: Confusion matrix of evaluator 1 and evaluator 2.

Figure 3 shows the six main categories and few of direct subcategories of the SDoEd ontology, full view in the GitHub repository [9]. Even though details are hard to see in the figure, it provides an overall "Gestalt" of the ontology.

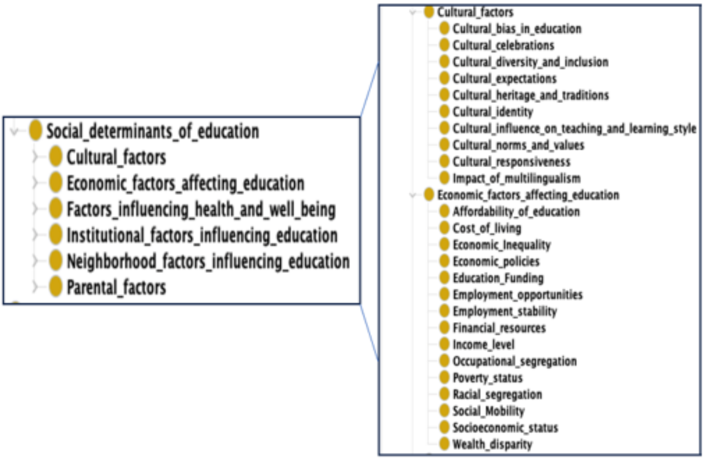


Figure 3: Six main categories and few of direct subcategories of SDoEd ontology.

5 Conclusions

The ontology for Social Determinants of Education holds significant potential in enhancing our understanding of the complex interplay between education and various socio-environmental factors, including health. By creating a comprehensive framework that captures the concepts, relationships, and dependencies within this domain, the ontology can serve as a tool for organizing and representing knowledge related to educational disparities, poverty alleviation, and reducing inequalities. This research contributes to the broader goal of leveraging data-driven and intelligent systems to enhance educational outcomes and promote equity considering pressing challenges, such as the recognition of structural racism. This prototype of the SDoEd ontology contains 231 concepts, 10 object properties, and 24 data properties. It is available in the GitHub repository [9].

6 Limitations and Future Work

To facilitate the utilization of the Social Determinants of Education ontology in natural language processing (NLP) tasks, we plan to annotate the ontology using CURIES IDs, which are shortened, standardized references that simplify concept identification. This will enhance the accessibility and interoperability of the ontology. Additionally, to enhance the richness of relationships within the SDoEd ontology, more contextual information will be incorporated. The authors also plan to explore various prompt engineering techniques and train an LLM for supporting ontology development.

Acknowledgments

During the final editing of several sections of this paper, ChatGPT was used to check for grammar and expression errors. No changes of the content were performed by it.

References

[1] C. Akenbor. "Institutional Factors Influencing the Academic Performance of Students". In: *International Journal of Higher Education Management (IJHEM)* 1 (Aug. 2014), p. 15.

[2] M. Đurišić and M. Bunijevac. "Parental Involvement as a Important Factor for Successful Education". In: *Center for Educational Policy Studies Journal* 7 (Sept. 2017), pp. 137–153. DOI: 10.26529/cepsj.291.

[3] K. Fortes et al. "The influence of social determinants of education on nursing student persistence and professional values". In: *J Prof Nurs* 39 (Mar. 2022), pp. 41–53. DOI: 10.1016/j.profnurs.2021.11.011.

[4] B. Glimm et al. "Hermit: An Owl 2 Reasoner". In: *Journal of Automated Reasoning* 53 (Oct. 2014). DOI: 10.1007/s10817-014-9305-1.

[5] N. A. Harris and L. E. Eberman. "Board of Certification Examination Achievement Gaps as a Barrier to Diversifying the Athletic Training Profession". In: *J Athl Train* 58.1 (Jan. 2023), pp. 1–8. DOI: 10.4085/1062-6050-0483.21.

[6] M. S. Knapp. *American Educational Research Association*. Online. 2003. URL: https://www.aera.net.

[7] L. J. Kolbe. "The Future of School Health Education in the United States: An Ontology". In: *J Sch Health* 94.7 (July 2024), pp. 661–673. DOI: 10.1111/josh.13436.

[8] N. Kollapally, Y. Chen, and J. Geller. "Health Ontology for Minority Equity (HOME)". In: *13th International Conference on Knowledge Engineering and Ontology*. 2021.

[9] N. M. Kollapally. *GitHub Repo*. Online. 2024. URL: https://github.com/navya777/Social-Determinants-of-Education-Ontology/tree/main.

[10] N. M. Kollapally et al. “An Ontology for the Social Determinants of Health Domain”. In: *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2022, pp. 2403–2410.

[11] N. M. Kollapally et al. “Integrating Commercial and Social Determinants of Health: A Unified Ontology for Non-Clinical Determinants of Health”. In: *AMIA Annu Symp Proc* 2023 (2023), pp. 446–455.

[12] A. S. Latif. “The Importance of Understanding Social and Cultural Norms in Delivering Quality Health Care-A Personal Experience Commentary”. In: *Trop Med Infect Dis* 5.1 (Feb. 2020). DOI: 10.3390/tropicalmed5010022.

[13] M. L. McHugh. “Interrater reliability: the kappa statistic”. In: *Biochem Med (Zagreb)* 22.3 (2012), pp. 276–282.

[14] M. A. Musen. “The protégé project: a look back and a look forward”. In: *AI Matters* 1.4 (2015), pp. 4–12. DOI: 10.1145/2757001.2757003.

[15] J. Nieuwenhuis and P. Hooimeijer. “The association between neighbourhoods and educational achievement, a systematic review and meta-analysis”. In: *OTB Working Papers* 2014 (Nov. 2014). DOI: 10.1007/s10901-015-9460-7.

[16] NIH. *PubMED Central*. Online. URL: https://www.ncbi.nlm.nih.gov/pmc/.

[17] N. Noy. *Ontology Development 101: A Guide to Creating Your First Ontology*. 2001.

[18] OASH. *Social Determinants of Health*. Accessed 2024. URL: https://health.gov/healthypeople/priority-areas/social-determinants-health.

[19] OpenAI. *ChatGPT*. Accessed July, 2024. URL: https://openai.com/chatgpt/.

[20] Y. Qiu and P. Ye. “The influence of family socio-economic status on learning engagement of college students majoring in preschool education: The mediating role of parental autonomy support and the moderating effect of psychological capital”. In: *Front Psychol* 13 (2022), p. 1081608. DOI: 10.3389/fpsyg.2022.1081608.

[21] ScienceDirect. *International Journal of Educational Research*. Online. URL: https://www.sciencedirect.com/journal/international-journal-of-educational-research.

[22] E. Sirin et al. “Pellet: A practical OWL-DL reasoner”. In: *Journal of Web Semantics* 5.2 (June 2007), pp. 51–53. DOI: 10.1016/j.websem.2007.03.004.

[23] D. Tsarkov. *FaCT++ reasoner*. Accessed 2024. URL: http://owl.cs.manchester.ac.uk/tools/fact/.

Enabling Blind and Low-Vision (BLV) Developers with LLM-driven Code Debugging*

Clark Saben, Jessica Zeitz and Prashant Chandrasekar
University of Mary Washington
Fredericksburg, VA, 22401
csaben@mail.umw.edu, jzeitz@umw.edu, pchandra@umw.edu

Abstract

BLVRUN is a command line shell script designed to offer developers within the blind and low-vision (BLV) community a succinct and insightful overview of traceback errors. Its primary function involves parsing errors and utilizing a refined large language model to generate informative error summaries. In terms of performance, our model rivals that of well-known models like ChatGPT or AI-chatbot plug-ins tailored for specific Integrated Development Environments (IDEs). Importantly, BLV users can seamlessly integrate this tool into their existing development workflows, eliminating the need for any modifications or adaptations to facilitate debugging tasks.

1 Introduction

Code debugging is one of the five main programming tasks or challenges that developers undertake while building programs or softwares [6]. Whether one is new to coding, or an experienced coder, one comes across the situation where their code, or the code they're working with, causes an error. We are specifically describing a situation where executing the code produces an explicit error. Our research does not address the scenario of unintended behavior where no error is produced on the screen. These "traceback" errors are unstructured text that is often verbose and difficult to comprehend. While the intention is to communicate as much information as possible, the source of the error, and the understanding of it, can only be discovered by putting together information from interpreting one or two key sentences. Developers of the blind and low vision (BLV) community are forced to process the lengthy error trace sequentially.

This process can be time-consuming as it extends the time and effort towards debugging and, thereby, writing correct code. BLVRUN is a shell script that runs in the background of any command prompt on any operating system. Once installed, it takes in error trace and summarizes it using a machine learning model that has been fine-tuned on the widely used Python bug dataset, PyTraceBugs [1]. The BLV programmer (user) is then presented only with a concise and insightful summary of the error trace. While code summarization or code description through machine learning is not new, our solution focuses on maintaining the current workflow of BLV programmers who use text buffers, printf-styled debugging methods, and command line. By using BLVRUN, BLV programmers do not need to make any change to their current workflow. Neither would they have to worry about committing to a "tool" and keep up with the tool or plug-in's documentation and development lifecycle. From their point-of-view, when a buggy program is executed, a summarized error is produced. This will greatly reduce the time it takes for BLV programmers to assess an error, thereby reducing, if not removing, the frustrations that BLV programmers would otherwise experience. For BLV programmers in the early stages of learning the art, this would be hugely impactful.

2 Related Work

This research effort is informed by two specific (sub-)areas of literature. The first area focuses on code debugging challenges. Code Debugging is listed as one of the primary programming tasks and/or challenges that programmers undergo in their development process [6]. Debuggers are primarily used for either: (a) analyzing runtime behavior, or (b) finding logical errors in code. Our research specifically addresses BLV developers who deal with (b). Debugging or, more specifically, the process of finding errors in code is heavily supported in IDEs. There are numerous reasons why current debugging tools or plugins are challenging to use for BLV developers. Firstly, use of debugging tools or plugins within IDEs require understanding of the spatial layout of IDEs. The functionality of the debugging tool is built without consideration of screen reader interpretation, leading to incompatibility between the two [2, 3, 9, 13]. Visual aids such as Syntax Highlighting or Syntax Error cues like squiggles, that assist in code debugging and possibly navigation, is challenging for programmers with visual impairments, especially when compared against sighted programmers [7, 9]. This results in BLV developers leaning on text editors instead of IDEs, employing "printf" debugging instead, and editing code using text buffers [2, 5]. These "workarounds" imply the requirement or use of the command-line interface in conjunction [8, 11]. This brings us to our second context or area of our research. Harini et. al highlight the lack of understanding of accessibility of CLIs [11]. One of the main findings of their work was the inaccessibility of scrolling a terminal with screen readers. Scrolling is extremely pertinent in the context of debugging. Not even considering the outcome of "printf debugging", scrolling through traceback errors is a challenge. We calculated the some metrics on a large Python software defect dataset. The median number of sentences per error, among 3864 errors, was around 26. The median number of words per error was 76. Which is why they recommended that (a) long output of unstructured text is converted into an accessible format, and (b) error messages be easier to comprehend.

The primary intent of this research is to aid BLV developers in their debugging efforts by improving the accessibility of CLI output. In the following sections we describe the design of our solution and report on its performance.

3 Solution

3.1 Approach

BLVRUN, our innovative CLI application, is designed to simplify the debugging process by providing concise summaries of traceback errors. It is built on a two-pronged approach:

1. A Rust-written shell application that operates in the background, monitoring the output of Python code executions. This script is adept at capturing the often unstructured and verbose text generated from traceback errors.

2. A fine-tuned 7 billion parameter CodeLlama model [10]. This model is specifically trained on traceback data, ensuring that even with reduced precision, it maintains robust performance on traceback summarization, even if its efficiency in other tasks diminishes.

LLMs are typically slow in generating text on standard consumer-grade hardware. To counter this, we have optimized BLVRUN by reducing the precision of the model's parameters. While this makes text generation faster compared to using full precision, it's still not rapid enough for an optimal user experience. To further address this, BLVRUN is configured to load the precision-reduced model upon startup. Consequently, every time an error is

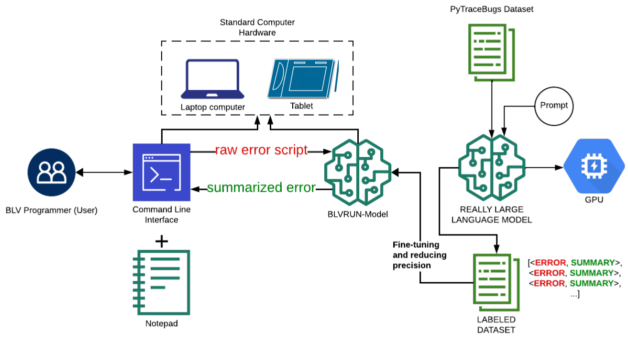


Figure 1: Architecture and Development Components of BLVRUN. Starting from the left, a BLV programmer, who using CLI and text buffers executes their Python code. When an error is produced, BLVRUN's script captures the verbose and unstructured text and only presents the user with a concise and accurate description of the error. This is possible because BLVRUN's model is fine-tuned using a dataset we created from PyTraceBugs. Finally, BLVRUN is optimized to run on any machine, thereby not requiring BLV programmers to depends on IDEs and/or switch contexts with ChatGPT-like solutions.

emitted in the terminal, BLVRUN swiftly captures the traceback information and consults the model for a summary.

This setup ensures that users do not need to adopt any new practices, gain additional knowledge, switch contexts, or acquire new hardware to better understand their code errors. In the subsequent (sub-)sections we delve into the details of BLVRUN's development. This is also shown in Figure 1.

3.2 Dataset

The dataset enhancing BLVRUN's model performance is PyTraceBugs [1]. This dataset includes training and evaluation data with 14,118 and 56 errors, respectively. The PyTraceBugs dataset offers a broad spectrum of traceback types. This is beneficial for model generalization and, therefore, user support. There are 555 unique error types across the dataset. However, the occurrence per error type is extremely sparse. Additionaly, we filter both training and test sets for the keyword "Traceback," focusing on this aspect in our model. We wanted to ensure that our evaluation of our model was rigorous. Therefore, we only fine-tuned and tested our model on certain categories of errors that were most commonly occurring in both the training set and the test set. These traceback errors include; TypeError, ValueError, AttributeError, IndexError, NameError, RuntimeError, and KeyError. Specifics about these errors in the dataset can be found at [1].

3.3 Fine-Tuning

Fine-tuning is a technique where we take a model that has already been trained on a general task and then continue training it on data specific to a particular domain. This process often results in the model performing better on the new, specific task. In the case of BLVRUN, the base model is provided by Meta and is known as CodeLlama [10]. Originally trained on Python code, CodeLlama serves as an advanced development assistant, generating Python code based on the context provided.

CodeLlama operates by taking the given code and predicting the next sequence of tokens, which are essentially bits of Python code. Meta has released several versions of the CodeLlama model, each differing in the amount of computational power and memory required. This variance is due to the number of parameters in each model. Parameters, the values in each layer of the neural network, are crucial in the process of output generation. The CodeLlama models come in different sizes, namely 7 billion, 13 billion, and 34 billion parameters, with larger models typically showing better performance.

The fine-tuning process itself is done using QLoRA [4]. QLoRA is an advanced technique designed to make training large AI models, like the one in BLVRUN, more manageable on regular computers. Training such extensive models typically requires substantial computing power, but QLoRA reduces this need by cleverly minimizing memory usage. At its core, QLoRA specializes in fine-tuning large models (with billions of parameters) while still maintaining high-quality performance. Think of it like precisely adjusting a complex machine to improve its efficiency. In the context of AI, it involves refining certain model components to enhance its task-specific effectiveness. A standout feature of QLoRA is its use of a novel data type called 4-bit NormalFloat (NF4). This data type is theoretically optimal for handling weights in the model that follow a normal distribution, a common scenario in AI model data. Essentially, it's akin to finding an incredibly efficient way of packing data into a smaller space without losing the essence of the information. These high-performance models offer a progressive approach to handling complex tasks. For instance, we use the larger parameter model to process a subset of the errors in the training set to generated our training data of "<error, summary>" pairs.

To build BLVRUN, we use the 7 billion parameter CodeLlama as the base model. We fine-tune the base model using the dataset generated from PyTraceBugs, in combination with the 13 billion parameter CodeLlama model (as mentioned previously). This approach helps BLVRUN to better understand and handle Python traceback errors, which are critical in debugging and development processes

3.4 Reducing precision

A common practice in deploying Large Language Models (LLMs) involves reducing the model's precision beforehand. This process is known as quantization. Reducing precision refers to storing fewer digits after the decimal point for each parameter in the model. The lower the precision, the smaller the model, thereby increasing the speed at which it can generate summaries. However, this approach entails a performance tradeoff. As the precision decreases, so does the model's representation quality, which can affect its effectiveness.

By concentrating on a specific task, the model is expected to maintain acceptable performance levels in that area, despite a potential drop in its overall capabilities.

The process begins by applying QLoRA. After the QLoRA fine-tuning, a specialized component called a LoRA adapter is produced. This adapter contains the adjustments made to the original model, enabling it to perform well on specific tasks despite the reduced precision. The LoRA adapter is then converted to match the format of the original model, ensuring compatibility. Subsequently, the base model and the LoRA adapter are merged. This combined model is further quantized to what is termed Q2K, where on average, each parameter is represented by approximately 2 bits. This extreme level of quantization greatly reduces the model's size from 12.55GB to just 2.83GB, making it more manageable for deployment in applications like BLVRUN.

As described further in the Evaluation section, the Q2K quantization achieves significant performance gains, especially when compared to a model that is simply quantized to 2 bits without the benefit of our QLoRA fine-tuning.

3.5 Addressing LLM Hallucinations

As noted in the literature [14], use of large language models introduces the risk of generating text unrelated to the prompt. Whether the information is simply unrelated or incorrect, the longer the generation the more likely a hallucination is to present itself. In order to mitigate this issue, BLVRUN enforces a strict 130 token generation limit. Tokens themselves are subword(s) determined by the tokenizer used by the underlying base model of BLVRUN[10]. The original tokenizer implementation used in [10] is used in [12]. The choice of 130 was determined empirically and doesn't correspond to an industry standard the authors are aware of.

3.6 User interface

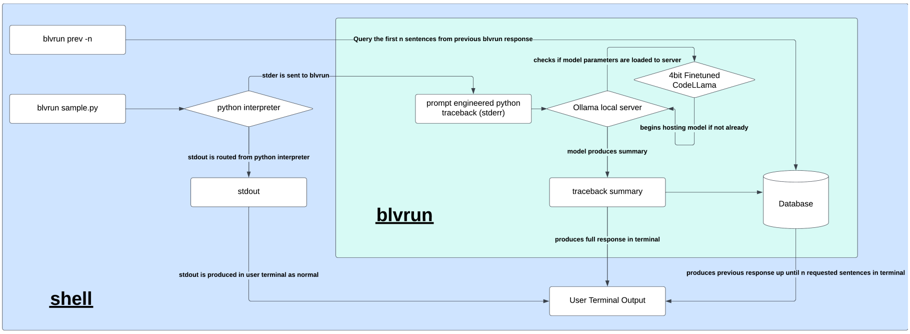


Figure 2: Information Flow within BLVRUN. When blvrun sample.py is executed in the shell, the prompt is sent to our model that is hosted on a Ollama server. Our model produces a traceback summary that is sent back to the terminal and saved in a database. BLV programmers can see previously generated summaries using the blvrun prev -n command.

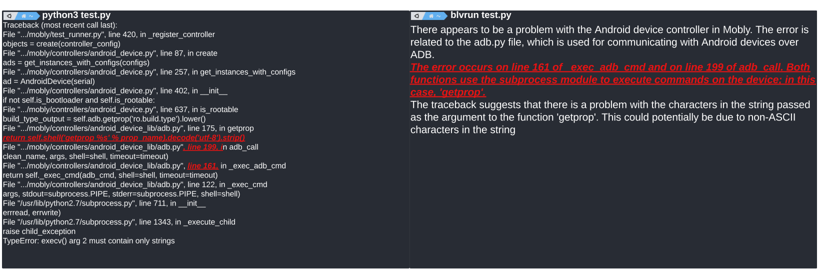


Figure 3: Example of the usefulness of BLVRUN. On the left one can see the unstructured, and verbose, output printed to BLV programmers (without the assistance of BLVRUN). On the right, we see the summary produced by BLVRUN. Within the summary, we have highlighted the key takeaway of the error, which BLVRUN presents it in a concise and, therefore, consumable manner.

The user interface of BLVRUN is command-line driven, designed to assist users in quickly understanding and acting on Python traceback errors. When a Python script encounters an error, as seen in the right of Figure 3 executing `blvrun sample.py` directs the traceback to BLVRUN, which then provides a clear summary of the error in the terminal. Where BLVRUN really improves quality of life while coding for BLV users can be seen when compared to the error shown on the left of Figure 3. A large traceback will often verbosely list line numbers from dependency files where each error occurs while not necessarily making it clear where in the user's code the issue is on a glance. BLVRUN provides an excellent workaround with allowing the model to read and summarize where the error likely occurs.

BLVRUN allows users to revisit the last generated summary with a command such as `blvrun prev -n`, enabling them to retrieve the 'n' most recent sentences from the summary for further review. This feature is particularly useful for users who need to recall or further examine the details of the last error without re-running the script. The database supporting this feature is optimized to hold just the previous response, maintaining system efficiency by avoiding storage of extensive historical data.

The focus of BLVRUN's UI is on functionality and ease of use, offering essential commands without overwhelming the user with unnecessary options or configurations. This ensures that the user can remain focused on their primary task—coding, while BLVRUN handles error summarization efficiently in the background.

4 Evaluation

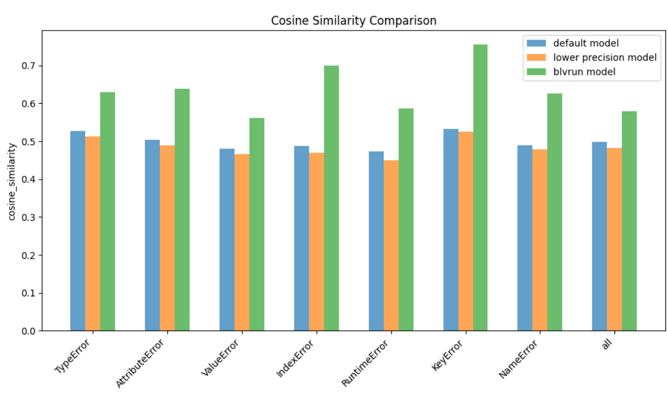


Figure 4: Cosine similarity scores of summaries generated by (1) base model (with no fine-tuning or lowered precision), (2) base model (with lowered precision), (3) BLVRUN's fine-tuned and optimized model compared against "gold standard"

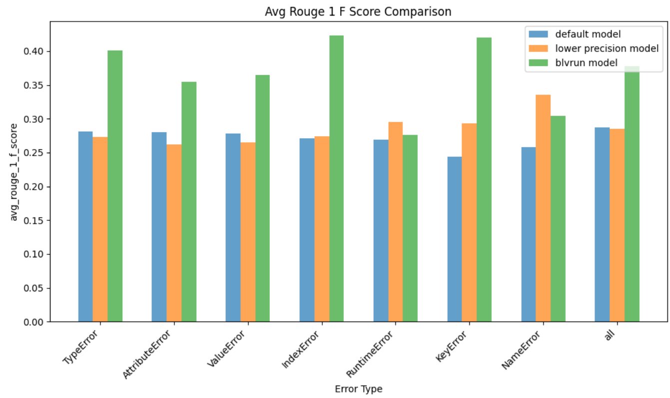


Figure 5: ROUGE-1 f-scores of summaries generated by (1) base model (with no fine-tuning or lowered precision), (2) base model (with lowered precision), (3) BLVRUN's fine-tuned and optimized model compared against "gold standard"

We designed BLVRUN to run on any computer hardware in the background of the terminal without disrupting a BLV user's workflow. BLVRUN is only helpful if it produces accurate summaries of long traceback errors. To evaluate the accuracy, we compared the output produced from BLVRUN against Meta's largest model (13 billion parameter model). PyTrackback's open dataset also provides a test set to be used for evaluation. Similar to the fine-tuning process, we employed Meta's model to generate a labeled dataset of "<error, summary>" pairs from the test set of PyTraceback. As a result, we compared BLVRUN's summaries against this "gold-standard" for 23 number of errors. We picked a subset of the errors based on the number of data points available for each error. We used cosine similarity and ROUGE-1 to compare the two summaries. From Figures 4 and 5, we see that our model achieved a fairly high similarity, and high ROUGE-1 scores. Most importantly, we can see the improvement achieved in the model performance when compared against the baseline non-fine-tuned model and non-optimized model. We find that BLVRUN does the best job in producing a shorter version of a lengthy traceback error by excluding the "noisy" text and highlighting the key insights. In the next section we describe our plans to expand on our model capabilities and our planned user study.

5 Planned Future Work

Our current and near-future efforts are two-pronged: (1) Widen the support for prompt message understanding: Currently, our model is fine-tuned to support error messages that were frequently represented in the error database. Our immediate next step is to widen the support for messages, error or otherwise, by generating a synthetic dataset to augment the low-incidence report of these messages. (2) Gain insight and feedback to inform future design of BLVRUN: We are in the process of getting IRB approved for our user study. We plan to recruit BLV participants who primarily use text editors and command line terminals in their current development workflow. Participants in our control group will be asked to perform tasks that involve interaction with the terminal without support of BLVRUN. Participants in our test group will perform the same tasks using the aid of BLVRUN. We will measure task performance, usability of the BLVRUN interface, and self-reported impact on participants (such as stress, fatigue, frustration, etc.). Insights from this study will further inform the development of future AI-models and design of the interface.

6 Contributions

AI- or LLMs-centric models are reliable enough that they are increasingly being included as part of the interface for programming IDEs. Our effort has demonstrated the feasibility of harnessing that potential and reliability into a command line interface that is considered, through support from many studies, as being more accessible as compared to IDEs. More importantly, our solution works on a local CPU, thereby not requiring BLV users to switch contexts or change their current programming "workflow." BLVRUN is currently fine-tuned and engineered to support more frequently occurring errors. This is done to support debugging-related efforts through the command line. We are refining our models to increase the scope of different types of program outputs that BLVRUN can support. In our user study, we plan to examine the the long-term impact of BLVRUN on the development process of the BLV programmers. Any insights will naturally inform the design of our model and future work on its interface.

References

[1] Elena N. Akimova et al. "PyTraceBugs: A Large Python Code Dataset for Supervised Machine Learning in Software Defect Prediction". In: *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*. 2021, pp. 141–151. DOI: 10.1109/APSEC53868.2021.00022.

[2] Khaled Albusays and Stephanie Ludi. “Eliciting Programming Challenges Faced by Developers with Visual Impairments: Exploratory Study”. In: *2016 IEEE/ACM Cooperative and Human Aspects of Software Engineering (CHASE)*. 2016, pp. 82–85. DOI: 10.1145/2897586.2897616.

[3] Catherine M. Baker, Cynthia L. Bennett, and Richard E. Ladner. “Educational Experiences of Blind Programmers”. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. SIGCSE ’19. Minneapolis, MN, USA: Association for Computing Machinery, 2019, pp. 759–765. ISBN: 9781450358903. DOI: 10.1145/3287324.3287410. URL: https://doi.org/10.1145/3287324.3287410.

[4] Tim Dettmers et al. “QLoRA: Efficient Finetuning of Quantized LLMs”. In: *arXiv preprint arXiv:2305.14314* (2023).

[5] Sean Mealin and Emerson Murphy-Hill. “An Exploratory Study of Blind Software Developers”. In: *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2012, pp. 71–74. DOI: 10.1109/VLHCC.2012.6344485.

[6] Aboubakar Mountapmbeme, Obianuju Okafor, and Stephanie Ludi. “Addressing Accessibility Barriers in Programming for People with Visual Impairments: A Literature Review”. In: *ACM Trans. Access. Comput.* 15.1 (Mar. 2022). ISSN: 1936-7228. DOI: 10.1145/3507469. URL: https://doi.org/10.1145/3507469.

[7] Maulishree Pandey et al. “Accessibility of UI Frameworks and Libraries for Programmers with Visual Impairments”. In: *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 2022, pp. 1–10. DOI: 10.1109/VL/HCC53370.2022.9833098.

[8] Maulishree Pandey et al. “Understanding Accessibility and Collaboration in Programming for People with Visual Impairments”. In: *Proc. ACM Hum.-Comput. Interact.* 5.CSCW1 (Apr. 2021). DOI: 10.1145/3449203. URL: https://doi.org/10.1145/3449203.

[9] Venkatesh Potluri et al. “CodeTalk: Improving Programming Environment Accessibility for Visually Impaired Developers”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI ’18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–11. ISBN: 9781450356206. DOI: 10.1145/3173574.3174192. URL: https://doi.org/10.1145/3173574.3174192.

[10] Baptiste Rozière et al. *Code Llama: Open Foundation Models for Code*. 2024. arXiv: 2308.12950 [cs.CL].

[11] Harini Sampath, Alice Merrick, and Andrew Macvean. "Accessibility of Command Line Interfaces". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: 10.1145/3411764.3445544. URL: https://doi.org/10.1145/3411764.3445544.

[12] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. 2016. arXiv: 1508.07909 [cs.CL].

[13] Andreas Stefik et al. "SODBeans". In: *2009 IEEE 17th International Conference on Program Comprehension*. 2009, pp. 293–294. DOI: 10.1109/ICPC.2009.5090064.

[14] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. *Hallucination is Inevitable: An Innate Limitation of Large Language Models*. 2024. arXiv: 2401.11817 [cs.CL].

Decoding SPAM: A Comprehensive Exploration of Unsolicited Messages*

Nooh Bany Muhammad[1], Tamer Alghamaz[2], Luma Alghamaz[2]
[1] Frostburg State University
Frostburg, Maryland 21532
[2] Al-Albayt University
Mafraq, Jordan
{nooh.bany, tamer.magableh, luma.naim94}@gmail.com

Abstract

In an era dominated by digital communication, spam remains a pervasive threat to privacy and security, causing significant disruption across internet platforms. This paper delves into the dynamic landscape of spam, tracing its evolution from simple unsolicited messages to today's sophisticated cyber threats that leverage advanced machine learning techniques to evade detection. By integrating historical insights with contemporary research, this study highlights the critical challenges in spam detection and underscores the necessity of innovative technological solutions paired with stringent legislative frameworks. We examine the effectiveness of current machine learning strategies in identifying and mitigating spam and propose a multi-faceted approach to enhance future spam detection systems. Our findings advocate for a balanced integration of technology and policy to robustly defend against spam, ensuring safer digital environments for users worldwide.

Keywords: Spam, machine learning in spam detection, emails, Social Media Spam, Spam Detection, trackback, spiders, bots, DDOS, spear phishing, deceptive phishing, vishing, pharming.

1 Introduction

Spam is the unsolicited messages delivered over the internet to a massive number of internet users. The use of spam originated back in 1864. This was when irrelevant and unsolicited messages were sent over the wires when telegraph lines were used to deliver information on investment opportunities to the affluent Americans. In this period, it was not labeled as Spam up to until the early 1990s.

The name spam originated from a Monty Python Sketch with a setting in a café, where Vikings hid out the information by being louder than others by singing out Spam repeatedly. The initial modern Spam was delivered on ARPANET, the militarized computer network that foreshadowed the internet.

In 1978, Gary Turk sent email spam to 400 people in order to advertise his contemporary line of computers. However, Turk's actions proved to be infamous that it took more than a decade for someone to attempt it. In 1993, the term spam had its initial application when irrelevant messages were sent to 200 users of Usenet newsgroup network accidentally. The first deliberate use of Spam was on 18 January 1994 when USENET sent letters to every newsgroup. The message was controversial and sparked a mass outcry. The first commercialized use of Spam was in April, 1994 was when two lawyers from Phoenix hired a programmer to send "Green-Card Lottery Final One" message to available newsgroups.

In 1996, the Mail Abuse Prevention System was found by software engineers, Dave Rand and Paul Vixie, to keep lists of IP addresses that sent mail Spam [7]. Numerous managers of networks wanted to keep the list to block spam emails.

In the 2000s, Spam became a lethal problem; hence the fight to control it began in early 2000. Therefore, regulations were set to counter Spam. On 20 April 2001, Justin Mason created a Spam-filtering system known as Spam Assassin and uploaded it on SourceForge.net. In May 2000, a computer worm called *ILOVEYOU* was sent to millions of users with windows personal computer. The resulting impact showed how Spam had become pervasive. In 2003, the CAN-SPAM Act was signed into law by G. Bush. The law was enforced to control spam. The fight to prevent spam has therefore picked up globally as it has become a nuisance.

2 New Common Spam

2.1 Comment spam

Comment spam is the unwanted comments posted to a blog, with the primary objective being to drop a link to the website of the spammer. Comment spam is also known as spomment, customarily characterized by the presence of replies that are useless to the blog entry, along with a link that directs into the commenter's website. The comments posted on sites where they are not required is Spam. Spammers use comment spam to post irrelevant advertisements as comments on forums, blogs, wikis and online guestbooks [2]. The main reason for spam comments is to link back to their own website so as to rank higher and acquire more traffic. Spammers don't usually choose a target. They use special programs that target websites based on various factors like the rankings of a particular site for multiple keywords and what the site is all about. Spammers want to manipulate the success of everyone for their own gain. There are several ways to recognize spam comments. Firstly, the site owner should prioritize real names over mysterious nicknames. Legit readers and corporations have no reason to hide their names [22].

2.2 Trackback Spam

The trackback spam is developed by spammers who use automated scripts to send millions of trackbacks globally. Trackbacks were formed to act as a notification to the webmaster of a new backlink by creating a link to the source of the backlink. The sole purpose of sending trackbacks is to leave their links on unmonitored websites [17]. Therefore, if a website is unmonitored, links are created by spammers from their website to your website. Spam links mostly direct one to suspicious activities such as selling drugs and distribution of malicious codes and programs. Your website will then communicate with a trackback. After the trackback is live, the link to your website is removed by spammers. This makes it appear like you are the one linking to them. Trackback spam can be mitigated by the use of plugins such as Akismet. Trackback spam can also be disabled by disabling trackback from the Discussion settings in WordPress.

2.3 Negative SEO Attack

Then there is the Negative SEO attack and it is a typical type of SPAM that mainly destroys businesses. The sole purpose of a Negative SEO attack is to fool google that you are the one doing Blackhat SEO tactics. It is used primarily by competitors who want to destroy your business instead of uplifting you. Spammers use Blackhat and unethical ways to eliminate and lower a competitor's ranking in search engines [17]. Spammers use it to make thousands of spammy links to your website. To prevent negative SEO attacks, there are several ways as once destroyed, and it cannot be fixed completely. Firstly, Google Webmaster Tools Email Alerts should be set. Secondly, one should monitor the backlinks profile. Most spammers perform negative SEO by making inferior ranks or redirects. Therefore, it is very important to know when a spammer is forming links and redirects to another website.

2.4 Spiders, Bots and DDoS Attacks

Finally, another common type of Spam is the Spiders, Bots, and DDoS Attacks. A spider is a bot of the internet that crawls through the internet and stores the data for search engine indexing. A bot is a program used by Google to gather information by scouring the web or performing tasks that are automated. Search engines often use crawlers like spiders and bots for internet browsing and create an index. This information may include the site the website links to. The spiders and bots are not necessarily destructive. However, spammers can use them to overload the firewall, Central Processing Unit, or the server's bandwidth. This is what is referred to as a Distributed Denial of Service (DDoS) attack. It occurs when the bandwidth is flooded by many users, usually a single or more web servers. The main goal is to disturb the normal operations of the targeted websites; when the website receives a very large number of users, it will crash if it is not ready for it. DDoS are used in blackmailing and extorting the site owners [17]. Spammers use different methods to perform DDoS, such as UDP fragmentation, CharGen attacks, and TCP anomalies. There are several remedies to DDoS attacks. In Cloudflare, the Cloudflare network is distributed across 102 data centers and manages more than 10 TB/s and handles all attacks. Another solution, Incapsula, consists of 32 datacenters and gives 32TBps. Incapsula can tackle attacks in 10 seconds or less. Other solutions are Akamai, AWS Shield, BeeThink Anti-DDoS Guardian, Sucuri, Cloudbric, Alibaba, F5, and Radware DefensePro.

2.5 Spam in media

Spams are used in various types of media. Firstly, it is through email. This has been discussed above. Also, Spam is spread through instant messaging. Systems for instant messaging are utilized by spammers to send messages. This is somehow less frequent than the email. Instant Messaging is mainly used by spammers as it can escape anti-virus software and malware. The solution against Instant messaging spam is by blocking any message from those not in your contact list. Another media is the mobile phone. Spam is sent through the text messages of a mobile phone. However, the number of spam sent through the mobile phones as text messages is significantly low. This is due to the high charges incurred when sending spam texts. The social media is also another media for sending spam. Facebook and X are vulnerable to spam [17]. Another media is the Voice over Internet Protocol (VoIP). When the user accepts a spam call, a pre-recorded spam message or advert is played back. Spammers use VoIP as they are affordable and provide anonymity. Moreover, mobile apps are also media for spam transmission. These are apps that utilize unrelated keywords excessively to attract users through indeliberate searches.

2.6 Virus spamming

At times, the virus sent as spam mails are sent automatically to different computers using scripts in the virus that causes the recipient's computer to deliver it to other address books. This is, however, without the recipient's knowledge. The sole purpose of virus Spam is to provide viruses across networks through the email. In addition to advertisement and virus delivery, spam mails are also used for phishing [3]. Phishing spam emails are fake emails that tend to cheat the recipient that the message came from a trusted source, for instance, banks. Phishing spam email is forwarded to targeted recipients into visiting a particular website and provide their financial data or any other relevant and sensitive information like login details. The spam mail is false at the first look and appears to come from the masked organization initially.

3 Spam Detection Technologies and Approaches

To counter spam most social networks have a spam reporting system in place. This allows users to tag specific posts as spam, so they can be later removed. Some platforms detect spam by finding spamming patterns through searching their data and activity logs, which are designed to monitor events on the platform [1]. The problem with such spam detection methods is that they take a relatively long time to return any results. Consequently, the spam will have a higher chance of being detected as it gets more attention from users ("likes" or "up-votes"), or if it follows the behavior of conventional spam. Furthermore, the current anti-spam technologies focus on classifying users as "spammers" by looking at their histories and behavior [1]. Which also means that it is harder to detect spammers with minimal "history" (like new users for example.) And considering that there will be an enormous number of requests to perform these "background checks", it will no doubt put a lot of strain on the social platform's API/computational limits.

In order to solve some of the aforementioned problems some researchers have suggested that social platforms evaluate individual posts and messages for spam without taking into consideration the user's history [1]. This works by analyzing a large dataset of spam messages and identifying certain patterns associated with spam. The larger the dataset, the more accurate the system will be in detecting future spam. This can potentially significantly reduce the amount of time it takes to take down spam posts. In addition, this approach can be applied to most social media platforms as it simply relies on text-based analysis. Another approach focuses on identifying spam patterns collectively across multiple accounts and multiple social media platforms [4]. In this method, posts from many different accounts are evaluated using a clustering algorithm [4]. This algorithm scans different posts by different users and detects if there are any common/duplicate spam URLs that are being collectively posted by these accounts. Then, these accounts are gathered and evaluated in bulk to confirm the existence of spam. The algorithm then puts the confirmed cases in one group and suspected cases in another. The suspected accounts are then evaluated manually (by the spam regulation team) to confirm their violation of the given platform's rules and policies.

3.1 Machine learning and SPAM

Recent advancements in machine learning have significantly enhanced spam detection capabilities. Techniques such as Deep Learning and Natural Language Processing (NLP) have emerged, enabling more sophisticated analysis of content for potential spam. These methods leverage large datasets to learn and predict spam patterns with greater accuracy, addressing the limitations of traditional machine learning approaches [1]. Each one of these methods are applied to their specific anti-spam duty, of course, they can fill more than one anti-spam types. Machine Learning generally works with a set of training samples, where it will train itself on samples with answers. These samples are already classified and then the process of learning is done for differentiating between what options are available [1].

Classifier Methods:

- **Naïve Bayes**. Bayesian classifier depends on events that happened and attempts to predict what possible future events that may happen based on the previous events that it dealt with. It basically never forgets its enemies. The Bayesian mainly predicts the occurrence of words based on its "spamminess" calculation tokens [1]. Where it would compare whether certain words appear more in spam or ham content messages or emails.

- **K-Nearest Neighbor (K-NN)**. K-NN is based to work with examples, where the provided data for its Machine Learning process, is used for comparison. It will assign the document it has to a category that other documents are like. Basically, it will put a document with the group of documents it belongs to or most similar to, and that group of documents belong to a certain category. The process of K-NN classifier can also be enhanced "using traditional indexing methods" [1]. The idea of K-NN is to do a real time process. It compares the document to the k nearest neighbor using its own algorithm, and the quickest match up of either spam or ham, it will classify accordingly.

- **Artificial Neural Networks (ANN)**. In short words also called "Neural Network" (NN). As its name suggests, this classifier is based on the biological neural networks, based on that its model is computational and interconnected (perceptron / multilayer perceptron) [1]. The ANN, like other machine learning classifiers, also learn by feeding it examples. It will keep generating function until it finds a function that correctly classifies all its examples, only the algorithm stops.

- **Support Vector Machines (SVM)**. The vector Machines follow the concept of decision making based on defined boundaries according to decision planes [1]. A decision plane is "one that separates between a set of objects having different class membership" [1], the SVM tried to separate two planes to place the set of objects into an easier scope of classification dimensions.

- **Artificial Immune System**. This classifier is based on the fascinating immune system in the human body, which is as we all know very successful in protecting us. Also, as we know, the role of our immune system is to protect us from viruses, bad bacteria and so on. The way our immune system functions give the Artificial Immune System classifier a mapped plan to build itself upon. This classifier follows the exact same steps of a human's Immune system in its algorithm, which has been very successful with the technology of Machine Learning [1].

- **Rough Sets (RS)**. Developed by Pawlak, this classifier computes reduction of information systems. It has a great ability to reduce redundant attributes, and this classifier method is designed to the need of generating simple useful knowledge. For example, if it was taking attributes of a human profile, it would check for the attributes it needs with their required values. Headache, muscle-pain, and temperature, if all of them have a value of yes then the profile of that human is sick, and so on the algorithm classifies on reduction of information systems.

4 Law and Regulations

The Computer Fraud and Abuse Act (CFAA) and the common law of contract and trespassing have been used to control Spam. This has been done indirectly by watching the unlawful acquisition of email. However, these solutions offer little help. Putting the focus on the acquiring emails has no support whatsoever for people whose email addresses are used by spammers. Mailing lists can be purchased, sold, or posted for free by those who have them. As a result, federal acts that address spamming directly to fill the gaping hole and provide an option after an email address has been obtained. To limit spamming efficiently, a new federal law that replaces the CAN-SPAM Act needs to be activated. Different federal laws that target spamming activities directly, preference to opting into opting out, have enough evidence to file complaints, and provides reliable resources for spam investigation would reduce spamming effectively. Federal laws against Spam can enhance the need for private backlists, blocking Spam without the generation of anticompetitive forces [6]. Unwanted commercialized emails have also been prohibited. Another legislative solution for spamming is the increase of funding allocated to control spamming in a federal way. Software that gets addresses must also be illegal, and the header of the message should always be able to tell if the content is an advertisement or not. Lastly, as spamming is a global problem, there has to be an international co-operation that fights against spammers as a single entity.

Legislative measures have also been laid to curb spamming. The CAN-SPAM Act of 2003 was passed to stop the high growth of unwanted commercial emails. Therefore, reducing the cost to recipients and Internet Service Providers of forwarding and discarding unwanted email. The Act also obligated the senders of advertising emails to use correct header information and clearly label their emails as advertisements or solicitation [14]. Also, the sender is required to notify the recipient of the option to restrict receiving incoming emails in the future. The sender is therefore illegalized from sending future emails to the recipient once the recipient has opted out. However, the CAN-SPAM Act has failed because of several reasons. Firstly, senders are not required to be granted permission before they send. Secondly, the Act has been under-enforced. Enforcing the act to limit Spam has been minimized hence considered a failure. Spammers still send spam emails despite users opting out.

Comparatively, the European Union's General Data Protection Regulation (GDPR) introduces stringent penalties for spamming, offering a different model of enforcement. An analysis of spam trends post-GDPR reveals a noticeable decline in spam originating from EU countries, suggesting the efficacy of harsh penalties combined with clear consent mechanisms. This paper advocates for international cooperation modeled on the GDPR's success, proposing a global spam treaty that harmonizes anti-spam laws, sharing of best practices, and joint efforts in spammer prosecution

4.1 Current state of regulation

In the current state of the state of regulations for Spam, there is several common state laws. Most of these laws are targeted towards the fake or commercial mails. Most states illegalize faking of the address origin of the emails or messages. The use of a third-party internet address without the owner's consent is also forbidden. Several states have even illegalized the selling and distribution of programs designed for the sole purpose of faking the source of the information or the routing information on email messages.

4.2 Spamming as a freedom of speech

Is Spam freedom of speech? Several questions concerning the protection of spam under the laws continue to prevail. When spammers send emails, this is their freedom of speech. Many states have made laws to monitor commercial spam and not political spam. Political Spam is a form of freedom of speech. The First Amendment Act allows the freedom of speech, but it does not guarantee forceful hearing of messages [20]. Regulations on spam, therefore, limit the freedom of speech despite being irrelevant. The Commerce clause prohibits states from discrimination against discrimination of interstate commerce. Spam is a form of interstate commerce. As not all spam are dangerous, therefore the freedom of speech cannot be blocked. The freedom of speech in spam has bad effects as malicious emails are sent to the recipient's accounts.

5 Conclusion

In conclusion, the fight against spam represents a critical front in maintaining the integrity and usability of digital communication platforms. As spamming techniques grow in sophistication, so too must our strategies to detect and neutralize them. This paper underscores the importance of ongoing research, ethical considerations, and international collaboration in the quest to secure our digital landscape against spam The discussion digs deeply to define Spam, types of spam, the utility of Spam, state regulations against Spam, and the technical and legislative solutions undertaken to reduce spam. On the other hand, social spam has had a profound impact on political discussions on social networks in recent years. Many social spam campaigns were launched with the goal of manipulating public opinions while the culprits remained largely anonymous. Spammers are also taking advantage of users in times of natural disasters to spread spam and malicious content. There are many ways to counter spam but none of the solutions work one hundred percent. All different strategies have advantages and disadvantages. Different strategies may be used in different scenarios.

6 Future Directions: Emerging Threats

As artificial intelligence (AI) technologies become more accessible, we anticipate an uptick in AI-generated spam that mimics human writing styles, making detection increasingly challenging. To combat this, future research should focus on developing AI models capable of understanding the nuances of human communication to distinguish between genuine messages and sophisticated AI-generated spam. For example, incorporating natural language understanding (NLU) techniques could enhance the model's ability to detect subtle cues of authenticity in messages.

References

[1] W. A. and S. ELseuofi. "Machine Learning Methods For Spam E-Mail Classification". In: *International Journal of Computer Science and Information Technology (IJCSIT)* 3.1 (2011).

[2] R. Amin, J. Ryan, and J. Van Dorp. "Detecting Targeted Malicious Email". In: *IEEE Security & Privacy Magazine* 10.3 (2011), pp. 64–71. DOI: 10.1109/msp.2011.154.

[3] A. Bhattarai and D. Dasgupta. "A Self-Supervised Approach to Comment Spam Detection Based on Content Analysis". In: *International Journal of Information Security and Privacy* 5.1 (2011), pp. 14–32. DOI: 10.4018/jisp.2011010102.

[4] Valve Corporation. *Steam*. Retrieved February 2024. URL: https://store.steampowered.com/about/.

[5] Valve Corporation. *Steam*. Retrieved February 2024. URL: https://store.steampowered.com/about/.

[6] B. Gao et al. *Forum mining for suspicious link spam sites detection*. U.S. Patent No. 8,219,549. 2012. URL: https://patents.google.com/patent/US8219549B2/en.

[7] A. George. *Google AI blocks 100 million spam messages daily*. Retrieved April 8, 2023. Feb. 2019. URL: https://www.digitaltrends.com/computing/google-ai-blocks-100-million-spam-messages-daily/.

[8] Google. *Ridding Gmail of 100 million more spam messages with TensorFlow*. Retrieved August 2023. 2019. URL: https://cloud.google.com/blog/products/g-suite/ridding-gmail-of-100-million-more-spam-messages-with-tensorflow.

[9] D. Gudkova. *Spam and the law*. Retrieved May 14, 2020, Accessed March 2024. 2010. URL: https://securelist.com/spam-and-the-law/36301.

[10] G. Hatchimonji. *Social media spam on the rise, says study*. Retrieved November 2023. 2013. URL: https://www.csoonline.com/article/2133983/social-media-spam-on-the-rise--says-study.html.

[11] J. Hong. “The state of phishing attacks”. In: *Communications of the ACM* 55.1 (2012), pp. 74–81. DOI: 10.1145/2063176.2063197.

[12] H. A. Jalab, T. Subramaniam, and A. Y. Taqa. “Overview of textual anti-spam filtering techniques”. In: *International Journal of Physical Sciences* 5 (2010), pp. 1869–1882. URL: http://www.cs.rug.nl/~tanguyen/pubs/article-Subramaniam.pdf.

[13] J. Ju, J. K. Lee, and J.-H. Ahn. “Competing with Spams More Fiercely: An Empirical Study on the Effectiveness of Anti-Spam Legislation”. In: *SSRN Electronic Journal* (2017). DOI: 10.2139/ssrn.2933278.

[14] M. Khonji, Y. Iraqi, and A. Jones. “Phishing Detection: A Literature Survey”. In: *IEEE Communications Surveys & Tutorials* 15.4 (2013), pp. 2091–2121. DOI: 10.1109/surv.2013.032213.00009.

[15] K. Lee, J. Caverlee, and S. Webb. “Uncovering social spammers”. In: *Proceedings of the 33rd International ACM SIGIR Conference*. 2010. DOI: 10.1145/1835449.1835522.

[16] Mc. *How Spam Works on Social Media*. Retrieved February 25. 2016. URL: https://www.mailcleaner.net/blog/spam-world-news/how-spam-works-on-social-media/.

[17] S. Raja and K. Venkatesan. “Electronic Mail spam zombies purify in email connection”. In: *International Journal of Advanced Research in Computer Science Engineering & Information Technology* 1.1 (2013), pp. 26–36.

[18] M. Rajdev and K. Lee. “Fake and Spam Messages: Detecting Misinformation During Natural Disasters on Social Media”. In: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. 2015, pp. 17–20. DOI: 10.1109/WI-IAT.2015.102.

[19] Y. Roth and D. Harvey. *How Twitter is fighting spam and malicious automation*. Retrieved May 13, 2020. 2018. URL: https://blog.twitter.com/en_us/topics/company/2018/how-twitter-is-fighting-spam-and-malicious-automation.html.

[20] K. M. Sullivan. *Two Concepts of Freedom of Speech*. Retrieved December 2023. 2010. URL: https://law.stanford.edu/publications/two-concepts-of-freedom-of-speech/.

[21] K. Thomas, C. Grier, and V. Paxson. Retrieved May 13, 2020. 2012. URL: https://www.usenix.org/conference/leet12/workshop-program/presentation/thomas.

[22] S. K. Trivedi and S. Dey. “Effect of feature selection methods on machine learning classifiers for detecting email spam”. In: *Proceedings of the 2013 Research in Adaptive and Convergent Systems on - RACS 13*. 2013, pp. 35–40. DOI: 10.1145/2513228.2513313.

[23] Twitter. *How Twitter is fighting spam and malicious automation*. Retrieved March 20, 2024. 2018. URL: https://blog.twitter.com/en_us/topics/company/2018/how-twitter-is-fighting-spam-and-malicious-automation.html.

[24] J.-P. Verkamp and M. Gupta. *Five Incidents, One Theme: X Spam as a Weapon to Drown Voices of Protest*. Retrieved February 14, 2024. 2013. URL: https://www.usenix.org/conference/foci13/workshop-program/presentation/verkamp.

The Impact of Changing a Course to Follow Equitable Grading Practices: A Case Study of Incremental Changes to Grading in Computer Science III*

David G. Cooper[1]
[1]Computer Science Department
West Chester University
West Chester, PA 19383
dcooper@wcupa.edu

Abstract

Alternative Grading systems have recently been framed as a way to grade for equity. These systems have been framed in many ways, mastery grading, specifications grading, un-grading, etc. In addition there are some steps in assessment that are different than those in standard grading systems. These include provide multiple chances to succeed, encourage revisions of work, change the scale of grading in the classroom to match the scale of grading of the school, and include flexibility with deadlines. In order to gain adoption of alternative grading, practitioners claim that a course can be changed incrementally using equitable grading practices. With that in mind, this paper explores the question of how robust final examination scores are to incremental changes to the grading of a course? Specifically, this paper looks at 1) the modification of homework assignments to have automatic test suites for grading, 2) the modification of multiple choice quizzes to allow for unlimited attempts by the deadline, and 3) the use of self-assessment worksheets and a self assessment narrative as the main grading criteria. This study looks at six different semesters of the third course in a sequence of computer science and programming fundamentals courses. Across semesters, incremental changes to the grading system were made, with only minor changes to the assignments given in order to deter cheating by using prior answers to assignments.

1 Introduction

Computer Science III in our program is the third programming course in a sequence of computer science courses for computer science majors. This is often the first programming course in our curriculum that transfer students take since they have transfer credit for the earlier courses. Computer Science III is taught in java, and it focusses on solidifying beginner concepts of programming and introducing intermediate java concepts such as Inheritance, Polymorphism, Interfaces and Abstract Classes, Generics, Exceptions, etc. For traditional students this is the 3rd semester of programming and either the 2nd or third class that is in the Java language. For transfer students some of the students haven't had any java programming experience. The goal of the course is to solidify student competency in programming while introducing students to object oriented concepts and the ability to program custom basic data structures such as linked lists and binary trees. The starting version of the course had 10 weekly programing assignments, two leetcode style assignments on text processing and linked list topics, 5 quizzes about 5 of the topics of the course, and a final comprehensive exam at the end of the semester. The course started with strict deadlines and grading and feedback based on the instructor compiling and running each assignment and checking the assignment for correctness and good style. Prior to this study this course was graded in a traditional matter. In the first and second semester of offering the class, auto-graders were created for each homework assignment in order to loosen the deadlines so that revisions were allowed after deadlines and students could get feedback from the auto-grader to know which parts of there project were functionally working. For assignments that came in on time, additional feedback from the instructor was provided. In addition all assignments were graded with a 50 as the minimum score. In the 3rd semester, no changes to the grading were made, but the quizzes were changed from being one time quizzes to having unlimited attempts with questions that change each time the quiz is taken. In the 5th semester, grading was changed away from weighted assessments to a set of narrative self-assessments that culminated in a final self-assessment that included a proposed grade. The instructor would use the evidence presented in the self-assessment referencing the completed work in the class and the final exam to determine the final grade. The goal of this paper is to examine if there are any noticeable changes with performance on the final comprehensive exam, the non-passing or withdraw rate, or a change in the overall grades of the students who passed the course.

2 Motivation

One of the reasons that alternative grading is done is the concept of equitable grading or grading for equity [3]. The idea is that some students, who don't fit into the system, lose the opportunity to learn because they may have other responsibilities that were not expected when standard grading practices were created. This means that in standard grading practices, the ability to make strict arbitrary deadlines can have more of an impact on a student's grade than the student's understanding or ability with the material of the course. There are two things that equitable grading is meant to address. 1. Is the rule around measuring the performance of the student fair. 2. Does the assessment of students measure expected learning outcomes, or is it a proxy. If it is a proxy, what information is lost, and could it be recovered with a different assessment tool?

Standard grading practices come with a colonial understanding that students should fit into the mold in order to be assessed. Those who don't make all the deadlines or turn in all of the assignments tend to be punished for non-conformity regardless of how much they've learned. A positive side effect of these standard practices is that they ensure the ability to get things done on schedule. There is a premium on submitting all of the assigned work and submitting it on time because there are real consequences when work is not completed to a predefined schedule.

An argument against standard grading practices is that the need to fit into a mold goes against the goal that the classroom is a place for *all* participants to learn *all* of the material. It has been argued that requiring students to fit into such a mold places inequitable expectations on the student and that lifting these expectations can lead to more equitable learning spaces.

With the goal of creating a more equitable learning space, this paper explores the incremental changes to a Computer Science III class in order to see if there are measurable changes for the students.

3 The Alternative Grading Menu

[5] discuss stakeholders in grades, alternative grading strategies, and modifications to tests to consider students with diverse needs in order to approach the goal of equitable, accountable, and effective grading. The stakeholders and primary concerns are Administrator/counselor concerned with the performance of the school, teacher concerned with instructional effectiveness, fam-

ily concerned with student's success in the courses, student concerned with showing others level of achievement and ability, and future employers are concerned with the student's competency on the job. The paper suggests grading strategies that could be broadly categorized as stakeholder related strategies and grading scheme strategies. The stakeholder related strategies are IEP-Based Grading, Individual Contracts, Shared Grading, Narrative Reports (from the teacher and optionally the student), Parent/Family conferences, and Self-Comparison Grades (amount of learning over time). The grading scheme strategies include three traditional methods Point Systems, Pass-Fail grading, and Weighted Grading. In addition, the alternative grading scheme strategies mentioned are Multiple Grades (content competency, effort, and progress), Checklist Evaluations (specifications grading), and Portfolio Systems. The top four modifications to tests suggested are extended time, simplifying wording of test questions, practice questions for study guides, and extra help preparing for tests. More recently, [2] suggested that courses should be modified to have equitable grading practices including minimum grading, learning outcomes based grading scales, smaller grading scales, multiple submissions, the removal of late penalties, and support to make it easier to make these modifications.

Recent explorations of alternative grading in computer science include binary grading [1], specifications and contract grading [7, 8, 4], and ungrading [6]. [1] take on binary grading is to create a final grade based on the number of successfully completed activities. Activities included 4 quizzes, 1 final exam (counting as 2 activities), 4 self assessments of an eight week group project, and 5 individual explorations. Completion of 12 of the 15 activities constituted an A in the class, 9 constituted a B, 6 for a C, and 3 for a D. Initial quizzes were at set times, but retakes could be done at any time before the end of the semester. The final was the only exam that couldn't be retaken and either counted as one activity if more than 50% was correct or 2 if more the 85% was correct. All other activities could be completed and revised any time before the end of the semester. They suggested that they would have a recommended schedule for completion, and set retake times for quizzes in future versions of binary grading. Students were asked about their perceptions of the grading process, but there was no comparison or discussion of outcomes as far as success rates compared to when traditional grading was used.

[7, 8] describe a discrete math and programming class using specifications grading. In the discrete math class, the grade was based completely on the number/percentage of topics that were passed/mastered. A quiz could be retaken for each topic after it was introduced until all questions were answered correctly for that topic. The final grade is the percentage of the ten topics mastered. In the programming class, the mastery quizzes were worth 40% of the grade, while participation was worth 10%, team assignments were worth

10%, and individual programming assignments demonstrating mastery in specific skills were worth 40%. Essentially 80% of the grade was based on mastery of skills from either quizzes or programming projects, while 10% assessed participation and 10% of the assessment came from a team effort. These courses did not compare their assessment to a traditional assessment.

[6] discusses a course based on the idea of ungrading. In this case assignments are given levels of completion, either binary for lab activities, 4 levels for Homework assignments and the final project, or a binary threshold for quizzes. Then each assignment type had expectations of the number of assignments completed at each level of completion for a grade of an A, B, or C, with anything under the C level being considered a C. There is also a maximum number of redos per assignment type for particular grades. In addition to completion of the work, students were given a survey about their expected grade, and sometimes students would request an audience to discuss grades. Most of the time students were given their requested grade, and most differences were within a letter grade of the request. This grading scheme was compared to prior times the course was taught, and there were no significant differences in the grade distributions, however there was significantly more effort using the alternative grading method.

[4] surveyed 11 papers that did some form of specifications or contract grading, including [7, 8, 6], and though the papers discussed subjective measures of improvement due to using the alternative grading practice, there was no objective assessment of the learning outcomes between a traditional grading approach and the

The dimensions that have been considered are deadlines of assignments, number of revisions or redos, schedule of retakes, type of assessment used for grading, grading scale, and whether behavior can have a side effect on the grade.

4 Methods

The goal of this study is to make incremental changes away from standard grading practices more towards equitable grading practices while keeping the same final self assessment as a way to monitor changes of learning. Changes started with having a minimum grade for any assignment of 50 and having resubmissions allowed for programming assignments only in the first two semesters. In the second two semesters quizzes were changed to be dynamic so that they could be redone as many times as the student wanted. In the third two semesters, students were required to do a self assessment narrative that discussed what they learned with regard to the learning outcomes instead of using a weighted grading system.

4.1 Fall 2021 & Spring 2022

The initial grade breakdown is shown in Table 1. Program projects are assigned weekly after a review of java and an introduction to GitHub classroom where all of the assignments must be submitted. Students were expected to submit an initial version of each project with the expected files stubbed out as a quick way for everyone to get started. Students were told to commit and push their code often and reminded that only their final version will be evaluated. A test suite was available for each assignment to give feedback about compilation and some functionality, but it was expected that students also test the their projects themselves. After the deadlines additional feedback would be given from the instructor. Some feedback was simply to turn the program in, while other feedback was more substantive explaining why some of the tests failed and giving suggestions for fixes.

Table 1: Meeting & Assessing Student Learning Outcomes version 1

Category	Percentage	Description
Quizzes	25%	5 quizzes total 5 points each
Program Projects	50%	10 homework assignments
Linked List and Text Processing Programs	10%	1% per problem
Final Exam	15%	Comprehensive final exam

Students were allowed to resubmit their projects, but needed to inform the instructor about any resubmissions after the deadline so that additional feedback could be given and new grades could be recorded.

4.2 Fall 2022 & Spring 2023

The next changes that were made was to make quizzes more dynamic so that they could be retaken until the concepts were understood, also, in order to reduce procrastination, students were given two formal times to submit revisions. The first was one week after the due date, and the second time was two weeks after the due date.

The PrarieLearn quiz system was used to transform quizzes that were multiple choice quizzes on the LMS into multi-select quizzes that were programmed to randomly change questions with different values, names, correct answers, and distractors. The topics of the quizzes were the same, but if a student took multiple attempts on a quiz, then they would get more coverage of the topic. Since the quizzes could be taken multiple times, most students took each quiz more than once, and many took each quiz more than twice. The instructors thought that taking the quiz multiple times could impact overall test scores, but there was no evidence that having the multiple quiz option improved the final exam scores.

4.3 Fall 2023 & Spring 2024

The most recent change to the course was to change the Linked List and Text Processing projects to be half-semester projects instead of sets of leetcode problems, to add 10 minute in class practice worksheets, and also to change the grading to be based on a narrative self assessment instead of a weighting of each of the categories of assignments. Table 2 shows the categories of assignments. The expectation was to complete each of the assignments, and to use the submitted work as evidence for the narrative self-assessment. A single final self assessment was used in Fall 2023, and for Spring of 2024 students were asked to do a midterm self-assessment to practice and get feedback before the final self-assessment. Students were required to do all of the assignments.

Table 2: Meeting & Assessing Student Learning Outcomes version 2

Category	Description
In Class Practice	topic based worksheets
Quizzes	5 quizzes
Program Projects	9 homework assignments
Linked List Exploration	Pair/Trio Project
Text Processing Project	Solo Project
Self-Assesments	Narratives based on outcome/topic grid
Final Exam	Comprehensive final exam

They were encouraged to submit multiple revisions until all tests passed, and for each project, they were required to write a self-assessment of what parts of the project improved learning outcomes for which topics of the course.

5 Results

The course can be measured in a number of ways. The first looks at the average final grade for each semester in Figure 1. If there is big change then that could indicate further investigation.

Next, Figure 2 looks at the raw final exam score across semesters and again look to see if there is a big change.

The third score to look at in Figure 3 is the scaled score and both look at the difference across semesters and also the difference between the average scaled final exam score and the average final grade to see that they move together.

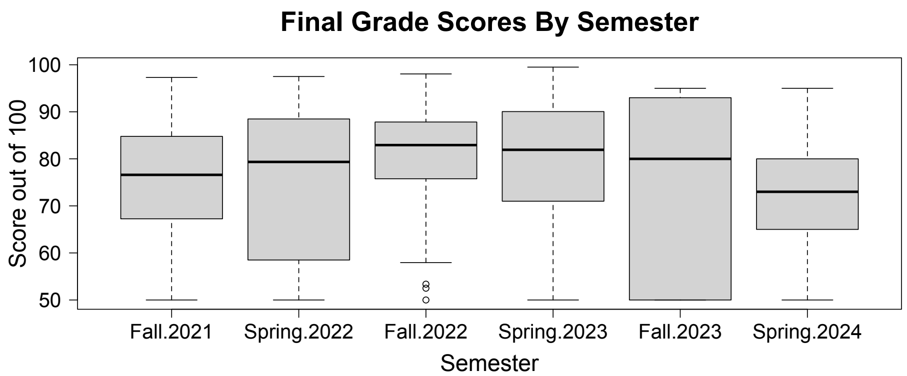


Figure 1: The median final grades rise a little bit from year to year until the final semester when the median score jumps down again.

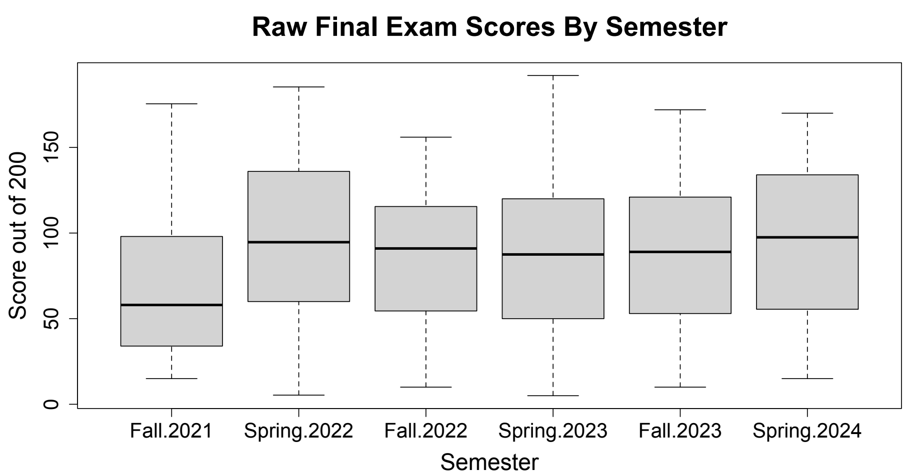


Figure 2: The median raw final score was pretty steady at around 50% of the answers were correct. The fall 2021 class had the fewest number of students (24), so that could be why it is further from the other medians.

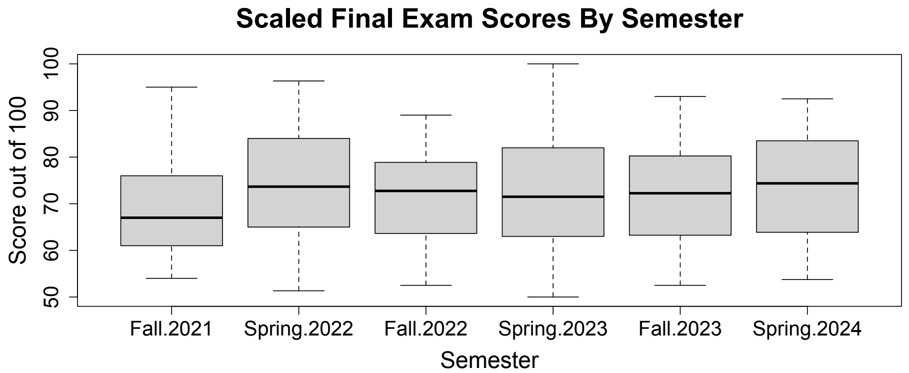


Figure 3: The median scaled final score changed between 67 and 74, which is only a seven point spread over the six semesters. The scaled score is just a compressed representation of the 200 point score into a range between 50 and 100.

When looking at passing rates in Table 3, it appears that there may be a higher chance of passing in the Fall than in the Spring. It looks like there may have been a raise in passing rates in year two due to changes in the class, but it could just be in the variability of classes. There doesn't appear to be any trend in the numbers.

Table 3: Passing vs Non-passing Rates by Semester

	Fall 21	Spr 22	Fall 22	Spr 23	Fall 23	Spr 24
%-age Passing	55.88%	50.00%	69.23%	60.32%	66.67%	46.55%
%-age D's	8.82%	6.67%	8.97%	9.52%	3.70%	12.07%
%-age F's	14.71%	25.00%	7.69%	9.52%	18.52%	22.41%
%-age W's	20.59%	18.33%	14.10%	20.63%	11.11%	18.97%
# of Students	34	60	78	63	54	58

6 Discussion

The biggest take-away from this study is that incremental changes of adding grading practices geared towards equity appears to have little or no significant negative impacts on the outcome of the students. This suggests that changing a course one piece at a time is a viable option for moving towards equitable grading practices.

In the latest two semesters, the self-assessment process was difficult for most students, and the instructor plans to add additional scaffolding for the self-assessment. In addition, in the Spring 2024 semester, a number of students tried to use only one or two homework assignments as evidence for a majority of the topics and learning outcomes. A more explicit expectation that a weekly homework assignment can work to justify at most one learning outcome of at most one topic, while different parts of the half-semester projects can be used for multiple learning outcomes of multiple topics as long as the justification is pointing to different parts of the project.

References

[1] Andrew Berns. "Scored out of 10: Experiences with binary grading across the curriculum". In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 2020, pp. 1152–1157.

[2] Adrienne Decker et al. "Transforming Grading Practices in the Computing Education Community". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, 2024, pp. 276–282.

[3] Joe Feldman. *Grading for equity: What it is, why it matters, and how it can transform schools and classrooms*. Corwin Press, 2023.

[4] Brian Harrington et al. "Specifications and Contract Grading in Computer Science Education". In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, 2024, pp. 477–483.

[5] Jo Hendrickson and Robert A Gable. "Collaborative assessment of students with diverse needs: Equitable, accountable, and effective grading". In: *Preventing School Failure: Alternative Education for Children and Youth* 41.4 (1997), pp. 159–163.

[6] Scott Spurlock. "Improving student motivation by ungrading". In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 2023, pp. 631–637.

[7] Ella Tuson and Tim Hickey. "Mastery learning and specs grading in discrete math". In: *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*. 2022, pp. 19–25.

[8] Ella Tuson and Timothy Hickey. "Mastery Learning with Specs Grading for Programming Courses". In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 2023, pp. 1049–1054.

Ad-hoc Ensemble Approach for Detecting Adverse Drug Events in Electronic Health Records*

Saurav K. Aryal and Howard Prioleau
Department of Electrical Engineering and Computer Science
Howard University
Washington, DC 20059, USA
saurav.aryal@howard.edu, howard.prioleau@bison.howard.edu

Abstract

Adverse Drug Events (ADEs) are a significant concern in healthcare due to their prevalence and impact. However, implementing such personalized systems will not be without its own challenges, as ADEs are frequently documented in electronic health records (EHRs), and the inherent complexities of these records, particularly unstructured clinical notes, pose significant obstacles for their identification. In this study, we seek to address these challenges by using large language models (LLMs) to detect ADE occurrences in clinical text. We will employ the zero-shot classification capabilities of various pre-trained LLMs and then integrate them into an ensemble approach to consolidate their predictions. We conducted a series of experiments utilizing various ensemble strategies and model comparison techniques to gauge the effectiveness of these models in detecting and classifying an ADE in medical notes. Our findings highlight the potential and limitations of current state-of-the-art LLM and their maximal zero-shot potential when leveraging ensembling, emphasizing the importance of continuous fine-tuning to enhance model reliability and applicability in real-world scenarios.

1 Introduction

Adverse Drug Events (ADEs) is a significant concern in healthcare due to their prevalence and impact. It annually accounts for 3.5 million outpatient visits and 1 million emergency department visits in the United States, also ranking as the fourth leading cause of death in the nation. The financial toll is also substantial, costing the U.S. healthcare system between $30 and $130 billion each year. Despite rigorous three-stage clinical testing requirements by the Food and Drug Administration (FDA) and randomized controlled trials to detect potential ADEs, many ADEs are only identified post-market. This gap underscores the critical need for effective detection methods that can improve patient safety and optimize treatment outcomes.

Furthermore, ADEs not only exacerbate the financial burden on healthcare systems but also influence the general public trust in medical institutions [7]. This erosion of trust can lead to hesitancy in seeking treatment or following medical advice, complicating health outcomes and driving patients towards alternative, sometimes unproven, treatments not aligned with evidence-based practices [6]. In addition, ADEs are often individual-specific, with elderly patients experiencing increased risks, highlighting the variability in how these events affect different populations [7]. This is further realized by the nocebo effect, where the expectation of side effects from treatment is consistently related to the symptoms being realized [9].

With this noted, it is evident that systems should be developed with a focus on being patient-specific, as this approach would theoretically lead to improved performance and better outcomes tailored to each individual's unique needs and circumstances. However, implementing such personalized systems will not be without its own challenges, as ADEs are frequently documented in electronic health records (EHRs), and the inherent complexities of these records, particularly unstructured clinical notes, pose significant obstacles for their identification. The medical notes within the EHRs are often embedded in intricate unstructured text that traditional data processing tools struggle to decipher. The subtlety of clinical language, diverse terminology, and variations in narrative style across different healthcare providers further complicate the automatic extraction and interpretation of relevant data. Furthermore, the voluminous nature of EHRs means that manually sifting through them to locate ADE instances is inefficient and prone to errors, underscoring the need for sophisticated computational techniques capable of handling large datasets with high precision and efficiency. Our study seeks to address these challenges by using large language models (LLMs) to identify ADEs.

In this paper, we plan to explore the potential of LLM to detect ADE occurrences in clinical text. We will employ the zero-shot classification capabilities of various pre-trained large language models (LLMs) and then integrate them into an ensemble approach to consolidate their predictions. This strategy will evaluate their collective proficiency in interpreting and processing clinical language related to ADEs, without the need for prior training on ADE-specific datasets. By leveraging the combined strengths of multiple models, we seek to understand current state-of-the-art LLMs on ADE detection.

2 Relevant Works

2.1 ADE Detection Task

The earliest proposal for ADE Detection can be found in ADE Corpus [12]. The authors developed the ADE corpus, which consists of 3000 MEDLINE case reports manually annotated for mentions of drugs, adverse effects, dosages, and their relationships. The annotation process involved three annotators and multiple rounds with intermediate harmonization steps to ensure high-quality, consistent annotations. The final harmonized ADE corpus contains 2972 documents with 4272 sentences annotated with drug-related adverse effect information.

Building upon this work, TAC 2017 [21] presented a task that challenged participants to extract mentions of adverse reactions and identify related modifiers such as negation, severity, and drug class. The task also required the determination of unique sets of positive adverse reaction mentions across all sections of a drug label and their normalization to the MedDRA terminology. However, the data provided in a simplified XML format may have resulted in the loss of important contextual information critical for accurate ADE extraction. Some of the limitations in [12, 21] were addressed in the 2018 National NLP Clinical Challenges Track 2 (n2c2) task [14], which serves as the primary data source and guide for this study.

The n2c2 task expanded the concept types to include medications, ADEs, and signature information like strength, form, dosage, frequency, route, duration, and reason. Furthermore, N2C2 required participants to identify relations between these concepts, linking them to their corresponding medications. This more comprehensive approach to ADE extraction aimed to provide a more complete understanding of the medication information and potential adverse events described in clinical narratives. The data retained its original format from the electronic health records, facilitating a smoother transition to real-world applications compared to previous tasks.

2.2 Modeling Techniques for ADE Detection

Several advanced machine learning approaches were employed in exploring the modeling techniques used for ADE detection across the three studies [12, 21, 14]. These studies leveraged a mix of traditional machine learning and deep learning techniques to tackle the complex nature of ADE detection. ADE Corpus [12] utilized a maximum entropy classifier trained on the ADE corpus sentences and achieved an F1 score of 0.70. Both [21, 14] modeling techniques were improved using Conditional Random Fields that were commonly used due to their effectiveness in sequence tagging, which is crucial to identify mentions of ADEs and their attributes, such as severity and negation. In addition, deep learning models, particularly those incorporating bidirectional Long Short-Term Memory networks combined with CRFs, were prominent. These models are adept at capturing the dependencies in sequential data, typical of narrative clinical texts where ADEs and related medical concepts are discussed.

As an example use case, LLMs, which only came into broader recognition and use after these studies were completed, have yet to be fully explored for their effectiveness in identifying ADEs in clinical texts. A study by [11] utilizes LLMs on the ADE Corpus, and they were able to achieve an F1 score of 91.99 by leveraging knowledge distillation to scale biomedical knowledge curation and focusing on ADE extraction. Despite their potential, the specific focus on applying LLMs to the n2c2 task has not been extensively pursued, although they were able to increase the baseline from the original study by 0.2 F1; this is what we seek to address in this study.

2.3 Ensembling

Ensembling is a technique that combines the predictions of multiple individual models to enhance overall predictive performance. This approach takes advantage of the strengths and mitigates the weaknesses of various models. It has demonstrated impressive general performance on a wide array of machine learning classification tasks, as indicated in [20, 10]. In addition, it has shown remarkable performance in the medical domain. As seen in [17], they deployed a diverse ensemble of SVMs enhanced by a weighted voting strategy using a simulated annealing genetic algorithm to optimize weight vectors, significantly improving performance across multiple unbalanced medical data sets. Through our research review and previous works, it was evident that ensembling has the potential for high performance in ADE detection [3, 5, 22, 18, 2, 4].

3 Methodology

3.1 Dataset

This research used the n2c2 dataset [14], which consists of approximately 500 discharge summaries from the MIMIC-III (Medical Information Mart for Intensive Care III) database. These medical documents were annotated for drug and

ADE information by seven domain experts, including four physician assistant students and three nurses, using specific entity tags and attributes. We used that annotation to categorize medical notes as containing adverse drug events (ADE) or not, based on the presence or absence of entity tags indicative of ADE in the documents. This classification schema, which labeled documents as containing ADE if they were tagged with ADE and as non-ADE otherwise, was consistently applied across both training and testing datasets. Specifically, the training dataset included 237 instances identified as containing ADE and 67 instances as non-ADE, while the test dataset contained 156 ADE and 47 non-ADE instances.

3.2 Pre-Processing

The pre-processing involves converting the structured format of medical records into a more natural language representation. This included extracting and reformating critical pieces of information such as admission and discharge dates, patient demographics (e.g., date of birth, sex), and the attending service details, which are originally presented in a coded format. This was done since the initial header format would have the models treat the data as a coding problem. So, for example, "Sex: F" is replaced with a more conversational format such as "Sex is F", and the content is restructured to flow like a standard written paragraph. Also, since the medical notes were large in token size, ranging from 8,000 - 15,000 tokens per medical note, we had to truncate the data to fit within each model's context window, which will be expanded more in the modeling section.

3.3 Modeling

We utilized seven models for zero-shot classification of ADEs to evaluate their performance as-is, to directly gauge the effectiveness of these models in recognizing and classifying ADEs based on their trained capabilities without any task-specific modifications. The selected models include BERT [8], Alpaca [23], GPT2 [19], and BART [16] for their robust general understanding of natural languages, and MedAlpaca [13], BioLinkBERT [24], and BioClinicalBERT [1] for their domain-specific specialized pre-training.

To achieve optimal zero-shot classification performance, the selection of precise labels was crucial. We experimented with various prompt designs for optimal zero-shot classification performance, including defining an ADE within the label, rephrasing the classification label, and much more. After experimenting, we selected labels "Does Not Contain Adverse Drug Event" and "Does Contain Adverse Drug Event" as these enabled the models to understand better and distinguish the classes.

Model	Param #	Data	Year	CWS
Alpaca	7B	Q/A	2023	2048
GPT2 XL	1.5B	General	2019	1024
BioClinicalBERT	108m	MED	2019	512
BERT	108m	General	2021	512
MedAlpaca	7B	MED	2023	2048
BART	406m	General	2021	2048
BioLinkBERT	333m	MED	2022	512

Table 1: Overview of Models Utilized: Parameters, Application Domains, Publication Years, and Context Window Size (CWS).

3.4 Experiments

To fully assess the performance capabilities of current models in classifying medical notes as either containing or not containing Adverse Drug Events (ADEs), we conducted a series of experiments utilizing a variety of ensemble strategies and model comparison techniques. Each model was used to predict the presence of an ADE in medical notes, and we aggregated the results through different methods to determine the most effective and reliable classification approach.

- **Majority Voting:** This approach involved taking the most common output (ADE or no ADE) from all models as the final decision for each document.
- **Minority Voting:** We also considered the least common output as the final decision, to explore the impact of outlier predictions on overall performance.
- **Most Confident by Probability:** The model output with the highest confidence (probability) was chosen as the final prediction for each instance, highlighting the influence of individual model certainty.
- **Sum of Positive vs. Negative Probabilities:** We aggregated the probabilities assigned to the positive (contains ADE) and negative (does not contain ADE) classes across all models, and the category with the higher total probability was selected.
- **Meta-Model (Decision Tree/Random Forest/SVM) of Votes:** A meta-model was trained on the votes (as features) of each classifier to make a final prediction, utilizing traditional classification algorithms to interpret the ensemble outputs.

- **Meta-Models of Class Probabilities:**Similarly, meta-models were trained on the class probabilities provided by each model, rather than their binary outputs, to refine the decision-making process.
- **Meta-Model for Both Votes and Probabilities:** This approach combines meta-models of votes and class probabilities, leveraging binary decisions and confidence levels to enhance predictive performance and robustness.

These experiments were designed to identify the most effective individual models and ensemble techniques and explore different strategies for integrating diverse model outputs into a coherent classification system. The meta-models used, including Random Forest (RF), LightGBM (LGBM), SVC, Gaussian Naive Bayes, Decision Tree, Logistic Regression, and Multinomial Naive Bayes. Were selected for their diverse algorithmic approaches enhancing the ensemble's predictive performance and robustnes. From these meta-classifications, we chose the top five performers based on weighted f1 and then parameter-tuned them for the final results. This fine-tuning process ensured the optimal performance of each of the selected models. The results of these experiments will help determine the optimal approach for future implementations in similar medical NLP tasks.

3.5 Evaluation

To assess and compare the performance of the ensemble models, we mainly use the weighted F1 score as the leading indicator of model performance due to class imbalance. We will also report accuracy, precision, and recall on both train and test set. Moreover, distinguishing between train and test sets for model performance in zero-shot classification is unnecessary; however, we make this distinction to facilitate direct comparisons in ensembling and provide a baseline performance foundation for further research.

4 Results

4.1 Zero Shot Classification Results

Our findings highlight the impressive capability of both general-purpose and domain-specific LLMs in detecting ADEs. BioLinkBERT and Alpaca achieved the best performance on the dataset, despite differences in model size and pre-training corpus. Surprisingly, domain-specific models like MedAlpaca and BioClinicalBERT did not perform satisfactorily, indicating the complexity of ADE detection and the importance of model pre-training quality. Overall, the performance reported shows the potential of LLMs trained in general language

	Train				**Test**			
model	f1	precision	recall	accuracy	f1	precision	recall	accuracy
BERT	0.706	0.697	0.766	0.766	0.665	0.63	0.734	0.734
BioLinkBERT	0.717	0.703	0.743	0.743	**0.702**	0.688	0.734	0.734
BioClinicalBert	0.46	0.654	0.424	0.424	0.458	0.63	0.424	0.424
BART	0.686	0.663	0.773	0.773	0.669	0.636	0.754	0.754
gpt2	0.709	0.693	0.753	0.753	0.671	0.646	0.714	0.714
MedAlpaca	0.100	0.358	0.214	0.214	0.131	0.533	0.241	0.241
Alpaca	**0.725**	0.760	0.789	0.789	0.683	0.671	0.754	0.754

Table 2: Comparison of Train and Test Weighted Metrics on Zero Shot Classification.

understanding to adapt to specialized domains like healthcare. However, the erratic performance across models underscores the need for ensembling to improve reliability and consistency and better understand zero-shot classification limits.

4.2 Ensembling Results

	Train				Test			
Ensemble Type	f1	precision	recall	accuracy	f1	precision	recall	accuracy
Majority Vote	**0.729**	0.792	0.796	0.796	0.669	0.636	0.754	0.754
Minority Vote	**0.086**	0.218	0.204	0.204	0.125	0.669	0.246	0.246
Most Confident Proba	0.702	0.684	0.753	0.753	**0.707**	0.699	0.754	0.754
Sum of Proba	**0.729**	0.792	0.796	0.796	0.669	0.636	0.754	0.754
Meta Vote RF	0.745	0.739	0.753	0.753	0.702	0.692	0.714	0.714
Meta Proba GNB	0.75	0.755	0.789	0.789	**0.707**	0.699	0.754	0.754
Meta Vote and Proba LGBM	1.000	1.000	1.000	1.000	**0.724**	0.743	0.778	0.778

Table 3: Comparison of Train and Test Weighted Metrics on Ensembling Types

The top-performing ensembling techniques included Most Confident Probabilities, Meta Vote and Probabilities with LGBM, and Meta Probabilities with Random Forest. We were able to improve performance over the individual best performing model by 2+%. This demonstrates the predictive capabilities of confidence levels derived from zero-shot classification in detecting ADEs.

However, the meta classifiers for Probabilities and both Vote and Probabilities tended to overfit the training set, yet they achieved the highest performance levels. Furthermore, the suboptimal performance of the minority voting ensembling indicates that there is minimal residual performance left during ensembling. This implies that the individual models in the ensemble are already performing well, and the ensembling process is effectively leveraging their collective knowledge. Also of note, for the Meta Classifier, we chose to present only the highest performing one to provide a clear benchmark of the best possible performance achievable through this method. Ensembling consistently outperformed individual models by leveraging the set of models, showcasing the benefit of ensembling in improving overall prediction accuracy and robustness. These findings underscore the ability of the current state-of-the-art LLMs and their maximal zero-shot potential when leveraging ensembling, understanding, and detecting ADEs, highlighting the extent to which these approaches can be applied in real-world scenarios.

5 Conclusion

Our study highlights the potential of ensembling techniques to enhance the detection of ADEs using LLMs. Our findings demonstrate that both general-purpose and domain-specific LLMs can effectively identify ADEs, with BioLinkBERT and Alpaca achieving the best individual performance. This highlights the impressive adaptability of LLMs trained on general language to specialized domains like healthcare. The best performing ensembling techniques were Meta Vote and Probabilities with LGBM, Most Confident Probabilities, and Meta Probabilities with Random Forest. These techniques showcase the predictive capabilities of confidence levels derived from zero-shot classification in ADE detection, emphasizing how far zero-shot classification can go in this domain. In conclusion, our findings highlight the potential and limitations of current state-of-the-art LLMs and their maximal zero-shot potential when leveraging ensembling, emphasizing the importance of continuous fine-tuning to enhance model reliability and applicability in real-world scenarios.

6 Limitations & Future Work

The major limitation of this work is the limited data availability and the small sample size, which restricts the robustness and generalizability of our findings. Additionally, the need for more compute resources posed a significant challenge. This constraint prevented us from testing the largest pre-existing models or extensively tuning the approaches we evaluated, limiting our analysis's depth. Furthermore, the context window size of the models used needed to be more significant to incorporate the full medical notes, which likely impacted the models' ability to understand and process the complex information in the notes fully. These limitations highlight the need for more comprehensive resources for a thorough and accurate analysis.

In the future, our research will focus on further finetuning these LLMs to detect ADEs. As we improve the performance of our models, we plan to apply them to extract additional instances of ADE occurrences from existing datasets, such as MIMIC-IV [15]. To ensure the accuracy of these mined results, we seek to establish collaborations with domain experts who can provide access to additional data and validate the generated labels. Upon successful validation, we aim to extract drug-ADE entity linkages to advance drug safety research and identify known ADEs associated with specific medications. Moreover, we envision developing a custom architecture that can process clinical notes and incorporate past EHR data to predict personalized probabilities of ADE occurrence at the point of prescription, further tailoring our approach to individual patient needs.

References

[1] Emily Alsentzer et al. "Publicly available clinical BERT embeddings". In: *arXiv preprint arXiv:1904.03323* (2019).

[2] Saurav Aryal and Howard Prioleau. "Howard university computer science at semeval-2023 task 12: A 2-step system design for multilingual sentiment classification with language identification". In: *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*. 2023, pp. 2153–2159.

[3] Saurav K Aryal et al. "Evaluating Ensembled Transformers for Multilingual Code-Switched Sentiment Analysis". In: *2023 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2023, pp. 165–173.

[4] Saurav K Aryal et al. "From Predicting MMSE Scores to Classifying Alzheimer's Disease Detection Severity". In: *Journal of Computing Sciences in Colleges* 39.3 (2023), pp. 317–326.

[5] Saurav K Aryal, Howard Prioleau, and Surakshya Aryal. "Sentiment analysis across multiple African languages: A current benchmark". In: *arXiv preprint arXiv:2310.14120* (2023).

[6] Dominic Balog-Way et al. "Effects of public trust on behavioural intentions in the pharmaceutical sector: data from six European countries". In: *Journal of Risk Research* 24.6 (2021), pp. 645–672.

[7] Kathrin M Cresswell et al. "Adverse drug events in the elderly". In: *British medical bulletin* 83.1 (2007), pp. 259–274.

[8] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR abs/1810.04805* (2018). URL: http://arxiv.org/abs/1810.04805.

[9] Kate Faasse and Keith J Petrie. "The nocebo effect: patient expectations and medication side effects". In: *Postgraduate medical journal* 89.1055 (2013), pp. 540–546.

[10] Heitor Murilo Gomes et al. "A survey on ensemble learning for data stream classification". In: *ACM Computing Surveys (CSUR)* 50.2 (2017), pp. 1–36.

[11] Yu Gu et al. "Distilling large language models for biomedical knowledge extraction: A case study on adverse drug events". In: *arXiv preprint arXiv:2307.06439* (2023).

[12] Harsha Gurulingappa et al. "Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports". In: *Journal of Biomedical Informatics* 45.5 (2012), pp. 885–892. DOI: 10.1016/j.jbi.2012.04.008. URL: https://www.sciencedirect.com/science/article/pii/S1532046412000615.

[13] Tianyu Han et al. "MedAlpaca–An Open-Source Collection of Medical Conversational AI Models and Training Data". In: *arXiv preprint arXiv:2304.08247* (2023).

[14] Sam Henry et al. "2018 n2c2 shared task on adverse drug events and medication extraction in electronic health records". In: *Journal of the American Medical Informatics Association* 27.1 (2020), pp. 3–12.

[15] Alistair EW Johnson et al. "MIMIC-IV, a freely accessible electronic health record dataset". In: *Scientific data* 10.1 (2023), p. 1.

[16] Mike Lewis et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension". In: *arXiv preprint arXiv:1910.13461* (2019).

[17] Na Liu et al. "A novel ensemble learning paradigm for medical diagnosis with imbalanced data". In: *IEEE Access* 8 (2020), pp. 171263–171280.

[18] Howard Prioleau and Saurav K Aryal. "Benchmarking Current State-of-the-Art Transformer Models on Token Level Language Identification and Language Pair Identification". In: *2023 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2023, pp. 193–199.

[19] Alec Radford et al. *Language Models are Unsupervised Multitask Learners*. 2019.

[20] Ye Ren, Le Zhang, and P.N. Suganthan. "Ensemble Classification and Regression-Recent Developments, Applications and Future Directions [Review Article]". In: *IEEE Computational Intelligence Magazine* 11.1 (2016), pp. 41–53. DOI: 10.1109/MCI.2015.2471235.

[21] Kirk Roberts, Dina Demner-Fushman, and Joseph M Tonning. “Overview of the TAC 2017 Adverse Reaction Extraction from Drug Labels Track”. In: *TAC*. 2017.

[22] Hrishav Sapkota, Saurav Keshari Aryal, and Howard Prioleau. “Zero-Shot Classification Reveals Potential Positive Sentiment Bias in African Languages Translations”. In: *ICLR Tiny* (2023).

[23] Rohan Taori et al. *Stanford Alpaca: An Instruction-following LLaMA model*. 2023. URL: `https://github.com/tatsu-lab/stanford_alpaca`.

[24] Michihiro Yasunaga, Jure Leskovec, and Percy Liang. “LinkBERT: Pre-training Language Models with Document Links”. In: *Association for Computational Linguistics (ACL)*. 2022.

Finiteness Considerations in Machine Learning*

Jeffrey C. Jackson
Mathematics and Computer Science Department
Duquesne University
Pittsburgh, PA 15282
jacksonj@duq.edu

Abstract

Many machine learning textbooks include at least some coverage of one or both of the No Free Lunch theorems for learning and probably-approximately correct generalization error bounds. However, it is not a simple matter to reconcile the implications of these two topics and provide advice to students (and practitioners) regarding when learning claims such as "this learned hypothesis will be at least 95% accurate on previously-unseen data" can reasonably be made. This paper shows how finiteness considerations can potentially provide such a reconciliation. It also suggests that finiteness considerations can be used to simplify certain generalization error bounds by eliminating their reliance on the VC-dimension of hypothesis classes, which might be of independent pedagogical interest.

1 Introduction

A key question in machine learning is, when can it reasonably be claimed that an algorithm has successfully learned? For instance, consider the problem of learning to recognize whether a photograph (of fixed dimensions) contains an image of a bicycle. Under what circumstances, if any, can it reasonably be claimed that a learning algorithm has produced a hypothesis function that with 95% accuracy classifies photographs as containing or not containing bicycles?

One possible answer to this question is provided by a Bayesian application of the No Free Lunch (NFL) theorems for learning [13]. Specifically, consider a rational agent who begins by assuming that all possible functions mapping photographs to bicycle/no-bicycle classifications are equally likely to be perfectly accurate, that is, who has a uniform prior over possible target classification functions. One of the implications of the NFL theorems is that, regardless of the learning algorithm employed by this agent, after applying the algorithm to a given training set to produce a hypothesis function, by Bayesian reasoning the agent should conclude that the average accuracy of the hypothesis—averaged according to the posterior distribution over target functions and over all unseen photographs—is exactly 50%, the same as random guessing. The NFL theorems might therefore seem to imply that the only way to make a positive claim for a learning algorithm's accuracy is to begin by assuming a non-uniform prior distribution over possible target functions.

Another possible answer to the accuracy question is provided by so-called probably-approximately correct (PAC) [11] generalization bounds. Such bounds can, under the right circumstances, be used to make statements such as "With probability at least 99% (the 'probably' part of PAC), this learned classifier is 95% accurate (the 'approximately' part)." This might seem to be a more promising answer than the one provided by the NFL theorems. However, the "probably" part of such claims does not mean what one might like it to mean. In fact, when we attempt to reconcile PAC bounds with NFL implications, we find that PAC bounds by themselves do not free us from those negative implications.

This raises another question: Rather than reconciling the NFL and PAC viewpoints, can we simply choose one viewpoint and ignore the other? It does not seem so. For many machine learning/artificial intelligence textbooks—e.g., [1, 3, 7, 8, 9, 10]—contain at least some coverage of the NFL theorems and/or PAC generalization bounds. That is, both viewpoints would appear to be well-established elements of machine learning pedagogy.

This paper will begin by expanding on the discussion above to clearly show that there is a fundamental conflict between the Bayesian and PAC views of learning when it comes to the necessity of making prior assumptions. Next, it will be shown that some PAC bounds can be simplified by making a certain finiteness assumption. This leads to employing a different finiteness assumption in order to reconcile the negative implications of the NFL theorems with the positive claims of PAC bounds. It will also be suggested that this finiteness assumption is reasonable. In the end, for those willing to embrace this assumption, it will be justifiable under certain circumstances to make learning claims without the need for prior assumptions.

2 The PAC/Bayes Conflict

This section will illustrate the fact that, from a Bayesian perspective, PAC generalization bounds are not meaningful without certain assumptions. The section begins by defining some learning models and terminology, then describes a Bayesian application of the NFL theorems for learning, and finally considers this Bayesian view of PAC bounds.

2.1 Machine Learning Models and Terms

Consider a very simple learning problem. There are 100,000 small disks numbered 1 through 100,000 in a room. The underside of each disk is either green or red. We will be given the opportunity to choose a small subset of the disks and to view their colors. Based on this, we would like to learn a function that, given a disk's number, will predict whether the disk is green or red. (We will consider a more general form of prediction function below.) This example and the photographs example of the Introduction will be used to illustrate a number of terms commonly used in machine learning. See any machine learning textbook, *e.g.*, [1], for additional details and terms.

The disk color learning problem is an example of a **supervised learning** problem, the type of problem studied in this paper (and a very common form of learning problem). Such problems have an **input space** $\mathcal{X}$, the domain of the function that we are attempting to learn, and an **output space** $\mathcal{Y}$, the codomain of the function. For the photographs problem of the Introduction, the input space consists of the set of all of the photographs that the learning algorithm might be given and the output space is {bicycle, no-bicycle}. For the disks problem, the input space is $\{1, 2, \ldots, 100{,}000\}$ and the output space is {red, green}. All of the learning problems considered in this paper will for simplicity be so-called two-class problems, problems for which $|\mathcal{Y}| = 2$.

We will throughout assume, again for simplicity, that there is a deterministic **target function** $f : \mathcal{X} \to \mathcal{Y}$ that assigns the correct output class to every input. We will further assume that there is no restriction on what the target function might be. So, if $\mathcal{X}$ is finite and that we are considering a two-class problem, the target could be any of the $2^{|\mathcal{X}|}$ possible functions $f : \mathcal{X} \to \mathcal{Y}$. For instance, in the disk problem, any of the $2^{100{,}000}$ colorings of the disks is considered possible initially.

A learning algorithm is given a **training set** d of input/output pairs, or **examples**. We will make the typical assumption that there is a fixed probability distribution D over $\mathcal{X}$ and that the training set is formed by successively drawing $x \in \mathcal{X}$ according to D and pairing each selected x with the value $f(x)$ assigned to it by the target function f. A training set for the disks problem might be formed by uniformly at random selecting disks 100 times and flipping them to reveal their colors. We would then have a training set of 100 examples.[1]

The function output by a learning algorithm, after it has processed the training set, is referred to as the **hypothesis** of the algorithm. In its simplest form, a hypothesis is deterministic, that is, is of the form $h : \mathcal{X} \to \mathcal{Y}$, just as the target is assumed to be. The PAC bounds considered later will assume deterministic hypotheses. However, the **hypothesis space** of all possible hypotheses output by a learning algorithm might be a proper subset of the set of all possible targets. For instance, a very simple learning algorithm for the disk problem might be designed to output only one of two possible hypotheses, one that ignores its input and always predicts "green" and the other that always predicts "red."

Bayesian learning allows for a more general, probabilistic, form of hypothesis. This and other aspects of Bayesian learning are covered as part of the next section.

2.2 Bayesian Learning and the NFL Theorems

Let us begin by considering a simplified version of the disk color learning problem in which there are only four disks numbered 1 through 4, and consider a Bayesian approach to this problem (see, *e.g.*, [9] for details on Bayesian learning). Given that all that we know is that each disk is red or green, it seems reasonable to begin with the assumption that all $2^4 = 16$ possible red/green colorings of the disks are equally likely. That is, we might begin by applying the principle of indifference [5], or in the terminology of Bayesian learning, begin with a uniform **prior distribution** over the set of 16 possible target classification functions that map disks to colors.

Once a prior has been fixed, Bayesian learning proceeds by drawing a training set d as described in the previous section. Let us begin with a training set of size 1. So, one of the disks is chosen at random—say, disk 3—and flipped. Imagine that its underside is green. Then, in Bayesian learning, we use this information to update the probability of each of the possible target colorings h_i by employing Bayes' Rule:

$$P(h_i \mid d) = \alpha P(d \mid h_i) P(h_i)$$

where d is the observed data, α is a normalization factor that guarantees that $\sum_i P(h_i \mid d) = 1$, and $P(h_i)$ is the prior probability of possible target h_i. For our problem, $P(h_i) = 1/16$ for every h_i. Also, notice that the conditional

[1] It is possible that this random process will select some input x more than once. In such a case, the resulting training set will have fewer than 100 distinct elements. Thus, although this paper follows the standard convention of referring to a training "set," training "collection" would be more accurate.

probability $P(d \mid h_i)$ is 0 for any h_i that assigns red as the color of disk 3 and is 1 for the remaining h_i. Since there are eight of each type of possible target, it is easy to see that after this Bayesian update the **posterior probability distribution** $P(h_i \mid d)$ will be 0 over the h_i that are inconsistent with the data we have seen (assign red to disk 3) and uniformly positive (1/8) over the eight possible targets consistent with this data.

Next, imagine that we instead form training set d by randomly selecting three disks and that those disks are 3, 1, and 4. Extending the reasoning of the previous paragraph, it is not hard to see that if all three disks are green then Bayes' Rule will produce a final posterior probability distribution over the h_i that places 50% probability on the target that colors disks 1, 3, and 4 green and disk 2 red, 50% probability on the h_i that colors all four disks green, and 0% probability on the remaining 14 possible target functions.

The hypothesis function produced by Bayesian learning is not deterministic but instead combines all of the targets according to the posterior probability distribution and therefore is itself a probability distribution over the output space values (classes). In this case, under the posterior distribution one target has 50% weight and says that disk 3 is red, another target has 50% weight and says that disk 3 is green, and no other target has any weight. Thus, the learned hypothesis is one that would predict that disk 3 is red with probability 0.5 and also predict that it is green with the same probability.

This is exactly the same hypothesis regarding disk 3 as the one that would be obtained by using the prior distribution to combine hypotheses, since the prior distribution assigned equal weights to the eight targets labeling disk 3 red and the eight labeling it green. In short, after observing the training data, the Bayesian learning algorithm has not changed its prediction for disk 3 and in this sense has learned nothing about disk 3.

What's more, it turns out that similarly negative results will hold for larger problems. For instance, if we return to the original problem having 100,000 disks and consider a training set of size 100, and if all 100 examples are green, Bayesian learning—if we begin with a uniform prior—tells us that the learned hypothesis should assign equal probability of red or green to any disk that was not part of the training set.

This analysis is taken a step further by applying the No Free Lunch theorems for learning [13]. One of these, Theorem 3, implies that no matter what learning algorithm—Bayesian or otherwise—is applied to the training data, from a Bayesian perspective and given a uniform prior all over possible targets, the prediction made by the learned hypothesis on disks outside the training set is just as likely right as wrong. Intuitively, the rationale for the theorem is that according to a Bayesian perspective that begins with a uniform prior, all disks outside the training set are just as likely red as green.

The following quote from [13] sums up nicely a key challenge posed by this Bayesian viewpoint for anyone wishing to make claims about hypothesis accuracy without invoking assumptions about prior distributions. **Generalization error** in this quote refers to the probability that a hypothesis incorrectly predicts the classification of an input that has not previously been observed, either as part of a training set or as part of a separate test set. (We will have more to say about generalization error soon.)

> In addition, the NFL theorems have strong implications for the common use of a "test set" or "validation set" T to compare the efficacy of different learning algorithms. The conventional view is that the error measured on such a set is a sample of the full generalization error. As such, the only problem with using error on T to estimate "full error" is that error on T is subject to statistical fluctuations, fluctuations that are small if T is large enough. However if we are interested in the error for [data not in the training or validation sets], the NFL theorems tell us that (in the absence of prior assumptions) error on T is meaningless, no matter how many elements there are in T.

So, returning to the disk learning problem, if based on a training set we hypothesize that all of the disks are green, and if we further observe that this hypothesis is completely accurate for the elements in a second, relatively large, set of randomly chosen disks (the validation set), this quote is claiming that the zero error observed on the validation set tells us nothing whatsoever about the performance of the all-green hypothesis on unseen disks in the absence of a non-uniform assumption about the prior over targets. This is consistent with the earlier comment that observing 100 consecutive green disks did not have any impact on a uniform-prior Bayesian learner's predictions regarding unseen disks.

With this challenge to the assumption-free use of generalization bounds in mind, we next consider PAC generalization error bounds and see how they do indeed run afoul of a Bayesian view based on the No Free Lunch theorems.

2.3 Probably Approximately Correct Generalization Error

We now consider a learning setting in which there is no assumption regarding a prior distribution over possible target functions. Hoeffding's bound on the tails of probability distributions over sums of a random variable [4] along with the union bound of probability theory readily give rise to the following observation (see, e.g., [1] for a derivation): If the hypothesis space of a learning algorithm has finite cardinality M and the algorithm is given N randomly drawn training examples of a target deterministic two-class function f, then the probability over the random choice of the training set that the overall error rate of the hypothesis h output by the algorithm differs by more than ϵ from the error rate of h on the training set is at most

$$2Me^{-2N\epsilon^2} \tag{1}$$

Here the **overall error rate** of h, $e(h)$, is the probability that h is correct on examples drawn according to the same distribution D used to generate the training set. That is, $e(h) = P_{x \sim D}(h(x) \neq f(x))$. The **training error rate** of h on training data d, $e_d(h)$, is the percentage of errors h makes on the N examples in d. That is, $e_d(h) = (1/N)\sum_{(x,y)\in d}(h(x) \neq y)$.

Technically, $e(h)$ is the error over the entire input space; in the case of disk learning, it is over all disks, those that were seen in the training set and those that are unseen. Thus, $e(h)$ is not exactly the same as the generalization error referred to in the previous section, which was error measured only over unseen examples. However, in typical learning problems, the training data set represents a tiny fraction of the overall input space. In such settings, the difference between overall error and generalization error as described in the previous section is, for practical purposes, nil. Such settings will be assumed in this paper, and $e(h)$ will therefore also be referred to as "generalization error."

Using δ to represent an upper bound on (1) and algebraically rearranging the resulting inequality, we can obtain the following probably approximately correct (PAC) bound on generalization error $e(h)$: With probability at least $1 - \delta$ over the random choice of training set d of size N,

$$e(h) \leq e_d(h) + \sqrt{\frac{1}{2N}\ln\left(\frac{2M}{\delta}\right)} \tag{2}$$

However, we are making no prior assumption here, and per the Wolpert quote of the previous section, a Bayesian view of the NFL theorems therefore suggests that $e_d(h)$ is meaningless.[2] How can we reconcile the seemingly positive PAC claims with the negative NFL-based claims?

Notice that the "probably" part of the PAC bound is a probability over the random choice of training data d. Imagine, then, that the target f is an essentially random, "white noise," function. Intuitively, given that the training data covers a small fraction of the input space, essentially any learning algorithm we might imagine will produce a hypothesis h having $e(h) \approx \frac{1}{2}$. However, if we fix a learning algorithm and repeatedly draw random training sets d for

[2]Technically, Wolpert's quote says that validation error, not training error $e_d(h)$, is meaningless. But clearly he has in mind that error measured on seen data is meaningless, in the absence of prior assumptions, when it comes to estimating error on unseen data.

this target f, there is some chance that eventually we will draw a d such that $e_d(h)$ is noticeably less than $\frac{1}{2}$. In this case, a PAC analysis based on (2) could lead us to erroneously claim that $e(h)$ is also noticeably less than $\frac{1}{2}$. The PAC bound tells us that we can expect that this will happen no more frequently than roughly a δ fraction of the experiments. But, as this example shows, it might be that *every* time this low-probability event occurs it leads us to conclude that h predicts the target better than random guessing does, which is false. On the other hand, every time this low-probability event does not occur, the generalization bound is uninteresting because it does not allow us to make a claim that h predicts better than random guessing. Put differently, if our algorithm is executed repeatedly on a "white noise" function, it will rarely (probability at most δ) have anything to say, and when it does say something, it is wrong! What's more, Kolmogorov complexity analysis [6] tells us that not only do such essentially-unlearnable functions exist but they are the norm within the set of all possible functions.

We see, then, why PAC generalization bounds by themselves do not allow us to escape the NFL-based negative conclusions: The "probably" part of the bound is not what we might like it to be, not "With probability 99% over the executions of my learning algorithm, the algorithm produces a 95% approximator." Instead, it is the probability that our learning algorithm is being misled by the random training data. This means that, depending on the target functions the algorithm is run on, the PAC bounds might *never* lead to a good approximator, despite the algorithm making occasional claims to the contrary. Furthermore, the NFL theorem analysis suggests that, even for a learning algorithm informed by PAC generalization bounds, we should (roughly speaking) expect the algorithm's hypotheses to be wrong as often as they are right if the target functions are chosen uniformly at random. That is, despite the use of PAC bounds, we should expect $e(h) = \frac{1}{2}$ on average (the exact equality holding only when $e(h)$ represents error on unseen inputs rather than overall error).

Hopefully, at this point the problem is clear: We would like to make learning claims without making prior assumptions, but the NFL theorems seem to imply that it would be unreasonable to do so, even if we use learning algorithms providing PAC guarantees. In the next section we employ finiteness considerations to begin developing a possible solution to this problem.

3 Replacing Infinite Hypothesis Spaces with Finite

The PAC bound given in the previous section assumed a learning algorithm that would output one of a finite number M of possible hypotheses. But many learning algorithms, such as those for learning neural networks, can—at least in theory—output one of an infinite number of possible hypotheses. In the case of neural network learning, this is because the neural network output by the algorithm is described by a vector of real-valued numbers (weights).

A standard approach to producing generalization bounds when hypothesis spaces are infinite is to employ Vapnik-Chervonenkis (VC) analysis of the hypothesis space [12, 2]. This produces a generalization bound (see [1], for instance, for details) with a form similar (up to constants) to that of (2) but substituting for M a term involving the so-called VC-dimension $VC(\mathcal{H})$ of the hypothesis space $\mathcal{H}$:

$$e(h) \leq e_d(h) + \sqrt{\frac{8}{N} \ln \left(\frac{4(2N)^{VC(\mathcal{H})}}{\delta} \right)}$$

One potential disadvantage of this approach is that, depending on $\mathcal{H}$, finding the correct value for $VC(\mathcal{H})$ can be a somewhat daunting theoretical undertaking. And even if an appropriate term is known, an instructor presenting bounds that use a VC term will likely find it prudent to spend course time providing at least a surface treatment of VC theory to provide context for the term. Although the theory has a certain amount of beauty in its own right, there would seem to be some value in having a simpler, perhaps more intuitive, alternative way to generalize error bounds to infinite hypothesis spaces.

And, at least for hypothesis spaces such as neural networks that are described by vectors of real values, there does seem to be such a way.[3] The approach depends on the fact that digital computers do not store real numbers; they store finite-bit-length approximations to real numbers. So, if a neural net is defined by the value of v weights in a computer using 64 bits to approximate real values, the effective size of the hypothesis space is no more than 2^{64v}. More generally, for a hypothesis space consisting of hypotheses defined by v parameters each occupying b bits, we obtain the **bit-based** PAC generalization bound

$$e(h) \quad \leq \quad e_d(h) + \sqrt{\frac{1}{2N} \ln \left(\frac{2^{bv+1}}{\delta} \right)}$$

If $VC(\mathcal{H})$ is taken to be v—v is the VC dimension for the simple hypothesis space of linear separators (perceptrons), and is typically an underestimate of the dimension for more complex spaces—then we can compare the VC and bit-based bounds numerically. For instance, under this assumption regarding $VC(\mathcal{H})$ and taking $b = 64$, it is not hard to show that for $N \geq 8$, the bit-based bound is tighter than the VC bound for any values of v and δ. For $b = 128$, $N \geq 128$ guarantees an advantage to the bit-based bound. Thus, for reasonable

[3] Avrim Blum first suggested something like this idea to the author.

values of b and N and a hypothesis space defined by v b-bit values, nothing is lost by using bit-based bounds in place of VC bounds.

This bit-based generalization bound would seem to have pedagogical value in its own right. But it also illustrates a concept that is so familiar that it is easily overlooked: We routinely use finite approximations in computing. The bit-based generalization error bound is one example of a broad array of applications of finiteness in rational decision making. In the next section, we will leverage a different form of finiteness to potentially allow PAC bounds to provide meaningful guarantees without prior assumptions.

4 Finiteness and Small Probabilities

The author would be more than happy to make the following bet with the reader. The reader pays the author \$1. We then begin simulating the flipping of a fair coin using some agreed-upon website, such as random.org, that generates random bits. We continue flipping coins/generating bits once per second until we observe 50 consecutive heads/1s, at which time I will pay the reader \$2 (inflation adjusted, plus interest earned on the \$1 wagered by the reader). Is the reader willing to play?

In theory, the reader *should* be willing to play. After all, if we flip the simulated coin enough times, we will almost certainly observe 50 consecutive heads at some point in time and the reader will be money ahead.

In practice, I feel confident that the reader *will not* be willing to play. The problem, of course, is that the two of us will almost certainly both shuffle off this mortal coil long before 50 consecutive heads is observed. (If the reader doubts this, replace 50 with 500 or some other sufficiently large number; at some point, even if one imagines that human lives can be extended indefinitely, heat death of the universe becomes a limiting finiteness factor.) So, the reader will almost certainly lose this bet.

We see, then, that finiteness considerations—in this case, finiteness of human life—can lead us to reasonably ignore small probability events. Mathematically, we must take all events into account, even those that have minute probabilities. Physically, when finiteness constraints are taken into account, it can be reasonable to behave differently. That is, our existence in a physical universe that is less than ideal mathematically can at times allow us to escape the strictures imposed by purely mathematical analyses.

Consider next a second application of finiteness considerations that is more directly related to machine learning. A typical learning algorithm, after it has completed executing on a training set, will produce not only a hypothesis function h but also some relevant information computed during training, such as the final error rate $e_d(h)$ of h on the training set d. Can we be absolutely certain that the $e_d(h)$ value output by the computer is correct? Of course not. Even if we were to assume that the learning algorithm software and the operating system were bug free—a big assumption—computers are physical devices and are therefore subject to errors due various physical phenomena such as radiation associated with cosmic rays [14]. Thus, although the chance of a computation producing an erroneous output that is not obviously in error is presumably extremely small, the chance is nonzero.

Does this mean that, if we give our students a homework assignment that asks them to report the training set error for a learned hypothesis, we should require that their submissions be of the form, "With probability such-and-such the computer has not erred and therefore the training error is so-and-so"? No, standard practice is of course to treat computer systems as if they run error-free and, modulo possible concerns about application software bugs, to accept their computations as correct. That is, we ignore the positive-but-tiny probability that a computing system has erred without our noticing. We are aware at some level that such an error might occur, so we would not entirely rule out evidence that such an error had occurred if such evidence were to be presented to us. But, barring such evidence, we simply trust that the computer itself has not misled us.

What we have been considering is how we treat the possibility of a computer misleading us regarding its calculation of $e_d(h)$ from a given training data set d. But, as noted earlier, another way that we can be misled is if randomly selected d is such that $e_d(h)$ (correctly computed) is not as close to $e(h)$ as suggested by the PAC bound of inequality (2). In such a case, we would be misled into mistakenly believing that $e(h)$ is smaller than it truly is. It is this type of misleading to which the "with probability at least $1 - \delta$" caveat of the PAC generalization bound applies. However, given that it is reasonable to ignore the possibility of being misled about the calculation of $e_d(h)$, might it not also be reasonable to ignore the probability-δ possibility of being misled about d being suitable for computing a bound on $e(h)$, as long as δ is sufficiently small, for instance, comparable to the chance of being misled about the computer's calculation of $e_d(h)$?

This thinking suggests that PAC generalization error bounds can be meaningful despite NFL-based Bayesian claims to the contrary. In particular, this statement is true, although potentially vacuous:

> If a rational person is comfortable ignoring the possibility of computer error in calculating $e_d(h)$ and if they are similarly comfortable ignoring the possibility of a misleading training set d being selected randomly given that the probability δ of this event is sufficiently small, then for such δ it is reasonable for them to accept (2) without any probabilistic caveat and without any prior assumptions regard-

ing possible targets.

This statement can be reconciled with a Bayesian analysis by observing that Bayesian analysis, like all of probability theory, informs our thinking in the limit as the number of repetitions of experiments grows without bound. But if the number of experiments is bounded—in particular, if there is good reason to expect that none of the training sets we observe will ever be misleading—then we should be open to the possibility that it is the Bayesian analysis itself that might be misleading. For, as noted earlier, purely mathematical, in-the-limit analyses do not always align well with our finite physical reality.

However, as already suggested in passing, perhaps the conditional statement above is vacuous. In particular, perhaps for every $\delta > 0$ and for every person, rationality prevents that person from being comfortable with ignoring the δ-probability of a misleading training set being drawn. It should be clear that the author believes otherwise and hopes that many readers will be similarly inclined. In fact, as suggested in the betting scenario at the beginning of this section, for a δ of 2^{-50}, or roughly 10^{-15} (one in one quadrillion), it seems that it might be reasonable to ignore the possibility of drawing a misleading training set. Also note that for learning problems for which large training sets are available, the bit-based generalization bound can be reasonably strong even when δ is tiny. For instance, for $b = 64$, $v = 3$, and $\delta = 10^{-15}$, $N = 100{,}000$ gives that, with probability at least $1 - \delta$ over the choice of training set, $e(h)$ is no more than 0.03 greater than $e_d(h)$.

Still, the author's approach to explaining to students how NFL theorems and PAC bounds can be reconciled has been to present a conditional statement similar to the one above and let the students decide for themselves whether they would be comfortable ignoring sufficiently small δ. Although student attitudes toward this approach were not formally surveyed, anecdotally, students seemed very receptive to the material presented in this way.

The conclusion of this paper is that if a machine learning student (practitioner) and their professors (clients) are all comfortable ignoring the possibility of misleading training data at some δ level, the student (practitioner) can—given sufficient training data and a relatively small training error $e_d(h)$—provide meaningful bounds on generalization error without making any prior assumptions regarding targets. On the other hand, short of ignoring the misleading-data possibility, it seems that the warning of [13] must be heeded: Without prior assumptions, generalization claims are meaningless.

References

[1] Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from Data.* 2012.

[2] Anselm Blumer et al. “Learnability and the Vapnik-Chervonenkis Dimension”. In: *Journal of the ACM* 36.4 (Oct. 1989), pp. 929–965.

[3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. 2nd edition. Wiley-Interscience, 2000.

[4] Wassily Hoeffding. “Probability Inequalities for Sums of Bounded Random Variables”. In: *American Statistical Association Journal* 58 (1963), pp. 13–30.

[5] J.M. Keynes. *A Treatise on Probability*. Macmillan and Company, limited, 1921. ISBN: 9780404145637. URL: `https://books.google.com/books?id=YmCvAAAAIAAJ`.

[6] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer New York, 2008. ISBN: 9780387498201. DOI: `10.1007/978-0-387-49820-1`.

[7] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Second edition. MIT Press, 2018. URL: `https://cs.nyu.edu/~mohri/mlbook/`.

[8] Kevin P. Murphy. *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022.

[9] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4th edition. Pearson, 2020.

[10] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[11] L. G. Valiant. “A Theory of the Learnable”. In: *Communications of the ACM* 27.11 (Nov. 1984), pp. 1134–1142.

[12] V. N. Vapnik and A. Ya. Chervonenkis. “On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities”. In: *Theory of Probability and Its Applications* 16.2 (1971), pp. 264–280.

[13] David H. Wolpert. “The Lack of A Priori Distinctions Between Learning Algorithms”. In: *Neural Computation* 8.7 (1996), pp. 1341–1390.

[14] James F. Ziegler. “Terrestrial cosmic rays”. In: *IBM J. Res. Dev.* 40.1 (1996), pp. 19–40. DOI: `10.1147/RD.401.0019`.

FACE: A Framework for AI-driven Coding Generation Evaluation*

Bao Ngo[1], Jack Formato [3], James J. May[2],
Nguyen Ho[3], Hoang Bui[3], Linh B. Ngo[2]
[1]Oberlin College, Oberlin, OH
bngo@oberlin.edu
[2]West Chester University, West Chester PA
{jm1006779,lngo}@wcupa.edu
[3]Loyola University Maryland, Baltimore MD
{jpformato,tnho,hdbui}@loyola.edu

Abstract

Previous work on evaluation code generation solutions is limited to static test cases due to difficulty in manual acquisition of test data. This paper presents a framework that enables the automated study of various code generation solutions using the entirety of an online competitive programming platform. To evaluate the capability of this framework, we exhaustively tested solutions generated from ChatGPT and Gemini for all programming questions on this platform. The resulting statistical and textual analysis highlights the difference between these two platforms and demonstrates the contribution of this framework in enabling researchers to collect and analyze a massive amount of data.

1 Introduction

The recent popularization of AI-enabled tools that are based on large language models has motivated researchers to investigate the long-term impacts and implications of these tools on the labor market. One working paper yet already highly cited finds that information processing industries will be exposed to "high economic impact without distinguishing between labor-augmenting or labor-displacing effects" [8]. From the perspective of software development, there are AI-enabled platforms that have the ability to generate code based on a problem statement. These platforms raise the possibility of partially or completely replacing software developers[6]. A search on Google Scholar with the phrase *ai code generation* yielded more than 18,000 results in 2023 alone, with ten initial pages of results including the majority of work published in IEEE/ACM peer-reviewed conferences and journals.

One of the challenges in study AI-enabled code generation platforms is how to test the quality of generated. Existing literature focuses on utilizing ready-to-use problem statements and test cases collected by manually mining publicly available data[7] [15]. This approach provides a massive amount of problem statements but is limited to only public test cases. Hidden test cases, such as those provided by LeetCode or Kattis, are not accessible to research purposes.

In this work, we present an approach to help alleviate the above problem by designing and implementing a framework for AI-driven code generation evaluation (FACE). Without direct access to all test cases, FACE can still leverage the online testing platforms' API interfaces to capture the evaluation of AI-generated code, thus enabling the investigation of code generation. FACE also allows easy modification and interfacing to multiple AI platforms.

The rest of the paper is as follows. Section 2 discusses the design and implementation of the FACE framework. Section 3 studies the effectiveness of the framework through the case study of two platforms, OpenAI's ChatGPT (GPT) and Google's Gemini (Gemini). A summary of other static test datasets is presented in Section 4. Section 5 concludes the paper and discusses future work.

2 Framework Architecture

2.1 Architecture

We design the system to automate the process of collecting thousands of programming problems from a coding challenge platform, querying an AI platform to generate solutions, and finally submitting the solutions back to the original platform for evaluation. Each programming problem typically consists of a problem description, sample input, and output. The result includes the status of the submission, whether the solution is accepted or rejected, and the reason for rejection. Figure 1 illustrates the overall architecture of our proposed framework, comprising three primary components: the Miner, the Generator, and the Submitter. Each component plays an important role in the seamless functioning of the system.

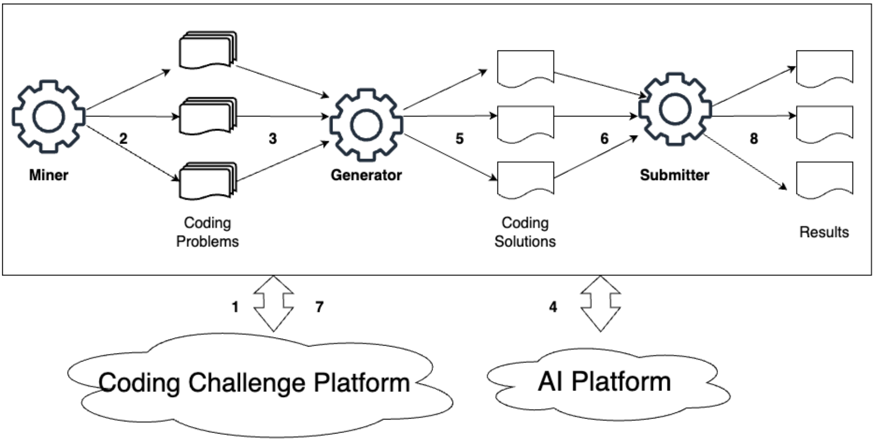


Figure 1: Data Collection Framework

The **Miner** is responsible for extracting the text of individual coding problems from the coding challenge platform. It gathers essential information, including detailed problem descriptions, corresponding test cases, and other requirements. This information serves as the foundational input for the subsequent stages of the framework. The **Generator** leverages the information procured by the Miner, this component queries an AI-enabled platform to generate potential solutions to the coding problem. By interpreting the problem descriptions and test cases, an AI-enabled platform generates a solution that aims to meet the specified requirements and constraints of each problem. The **Submitter** takes over the process of submitting the solution back to the coding challenge platform for evaluation. This component ensures that each solution is assessed by the original platform, with the results being saved in text files, creating a repository of results that can be analyzed, and evaluated to refine the system further.

2.2 Implementation

The 8-stage workflow of FACE is visualized in Figure 1. In our prototype, the Miner obtains coding problems listed on Kattis[1], a popular coding challenge platform. First, the Miner extracts a list of problem names and constructs a URL for each problem. In our implementation, we employ the autokattis2[2] Python library to request (Stage 1) an HTML page for each problem. Then,

[1] https://open.kattis.com/problems
[2] https://www.piwheels.org/project/autokattis/

Metadata	Description	Sample Test Cases
different https://open.kattis.com /problems/different 2.8 Medium	Write a program that computes the difference between non-negative integers. Input Each line of the input consists of a pair of integers. Each integer is between 0 and 2^{15} (inclusive). The input is terminated by end of file. Output For each pair of integers in the input, output one line, containing the absolute value of their difference.	Sample Input 1 10 12 71293781758123 72784 1 12345677654321 Sample Output 1 2 71293781685339 12345677654320

Table 1: An example of a coding problem from Kattis

the problem name, difficulty score, difficulty level, problem description, and sample test cases are extracted from the HTML page. All data associated with each problem is stored (Stage 2) in text files in a single folder.

In Stage 3, the Generator reads the information obtained by the Miner and begins constructing an AI prompt for each problem. Each prompt includes a leading question that describes the objective of the prompt, along with the problem description and the test cases. An example of a leading question is: *Write a python program for the following problem and make sure that the variables' names and functions' names are different, and also, only use internal Python libraries, not external Python libraries.* These constraints are added to ensure that the solutions generated by AI platforms (ChatGPT or Gemini) comply with Kattis' judging system, which only allows the use of internal Python libraries. In Stage 4, the Generator uses OpenAI APIs or Gemini APIs to send the constructed prompt to the specific platform to generate the solution. A copy of the generated solution is saved (Stage 5) locally in a text file before the Submitter takes over and submits the solution to the Kattis' judging system for evaluation.

Once Python solutions generated by ChatGPT/Gemini are obtained, the Submitter reads (Stage 6) the solutions from local storage and submits (Stage 7) them back to Kattis' judging system for evaluation. Based on the number of passed test cases, Kattis' judging system will label the result. The label

Result
Submission received. Submission ID: 13469967. Submission URL: https://open.kattis.com/submissions/13469967 New...New...New...New...New...New...New...New... Test cases: [..] 2 / 2

Table 2: An example of a result file from a Kattis submission

is either Accepted (*AC*), Wrong Answer (*WA*), Runtime Error (*RTE*), Time Limit Exceeded (*TLE*), or Memory Limit Exceeded (*MLE*). We discuss the meaning of these five statuses in the next section. The result is stored (Stage 8) in a text file for further evaluation. Table 2 shows an example of the result from a Kattis submission through the Submitter.

2.3 Technical Discussion

For coding challenge platforms with a large amount of problems (e.g., 3,323 coding problems for Kattis), it is important that FACE does not accidentally flood that platform with a large amount of requests. In addition to the rate limits of ChatGPT and Gemini, various timing delays were included in Stages 1, 4, and 7. We find that random delays between 60 to 100 seconds are adequate for all external platforms.

The additional of timing delays significantly increases the duration it takes to completely generate all solutions. Throughout this process, we encounter various errors such as losing network connection and reaching the monthly quota limits on ChatGPT/Gemini. To make our framework fault-tolerant, we develop a *checkpoint* component that allows the Generator to restart where it failed. To future-proof FACE, we also decide to not overwrite generated solutions but to save multiple versions across different runs. This leads to a change in the Submitter, which is able to identify and load only the latest solution. This will let us later expand FACE to support prompt refinement.

3 Case Study: Analyzing ChatGPT and Gemini

In this section, we use FACE to study the solution generation process using two separate AI platforms: ChatGPT and Google Gemini. The generated solutions are submitted to Kattis and the evaluation results are collected and validated.

3.1 Programming Problem Description

Our study examines 3,323 programming contest problems from Kattis. After the AI-generated solutions are submitted to Kattis via its Python API, FACE captures the returned text, which contains information about how many test cases passed and what the final status is. These results, in addition to input information, create the core features set to be later analyzed. The features can be categorized into two groups, one including attributes of programming problems (*Problem*, *Difficulty*, and *Description*) and the other including attributes of the resulting evaluations (*Status*, *Pass*, and *Total*).

For each programming problem, *Problem* provides a unique problem name, which is used to generate a direct URL to the Kattis problem. *Difficulty* is a number representing Kattis' difficulty ranking for the problem. Kattis' difficulty values are neither fixed nor manually assigned but calculated based on the ratio of successful solutions versus failed attempts. Problems that are solved by many and have few failed attempts have lower difficulty scores. Problems that are frequently tried but have more failed submissions have higher difficulty score. The lowest difficulty score for Kattis problems is 1.1, and the hardest problem has a difficulty of 9.7. *Description* contains the problem's text description, which contains all requirements and information needed to solve the programming contest.

The resulting text captured through Kattis' Python API allows the framework to extract the final evaluation statuses, which include *Accepted* (AC: The submitted solution passed all tests), *Wrong Answers* (WA: The solution ran, but could not pass all tests. Failure to pass a test could mean either incorrect results or incorrect output format of results), *Run Time Error* (RTE: The solution crashed and could not produce a result), *Time Limit Exceeded* (TLE: The solution took too long to run), and *Memory Limit Exceeded* (MLE: The solution required more memory to run than allowed by the problem statement). *Pass* indicates how many tests were successfully passed by the submitted solutions prior to failure, and *Total* indicates how many tests are there in total for the problem.

We were able to collect 3323 solutions to unique Kattis problems from ChatGPT but only 2139 from Gemini. The number of generated solutions from Gemini has been limited by the daily rate limit for our Gemini account. The two platforms share 1981 unique problems. Table 3 presents a breakdown of the status counts across these common problems. Value in cell *(i,j)* of the matrix represent the number of problems with solution status i for Gemini and solution status j for GPT. The final column and the final row represent the total count for each problem status for Gemini and GPT, accordingly.

Table 3: Status summary matrix for Gemini solutions and GPT solutions

	AC	WA	RTE	TLE	MLE	Gemini Total
AC	114	75	20	8	0	217
WA	59	820	296	31	9	1215
RTE	23	142	136	10	2	313
TLE	15	130	38	42	4	229
MLE	0	8	2	1	6	17
GPT Total	211	1175	492	92	21	**1981**

3.2 Exploratory analysis

As shown in Table 3, the numbers of *AC* (186/192), *WA* (1023/1066), and *MLE* (20/15) results sare similar for both platforms. On the other hands, the number of *RTE* (431/278) and *TLE* (71/180) results *MLE* differ from one another, with GPT has more *RTE* statuses and Gemini has more *TLE* statuses. These differences are visualized in Figures 2, and 3.

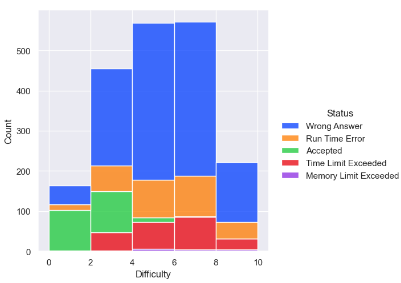


Figure 2: Status per Difficulty Range (Gemini)

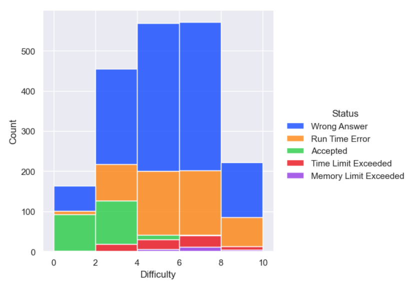


Figure 3: Status per Distribution Range (GPT)

In addition to the evaluation results, FACE also stored the generated Python solutions, enabling the study of the source codes. Sample-generated solutions from Gemini and GPT are presented in Listings 1 and 2, respectively. Visual inspection of the generated solutions shows that Gemini generates codes with more comments and well-defined functions and variable names according to good software engineering conventions. On the other hands, GPT generates more abbreviated code with function and variable names taken directly from the problem' text.

Listing 1: Gemini

```
def evaluate_sound_duration():
    """Compares the duration of Jon Marius' "aaah" with
        ↪ the doctor's requirement."""
    patient_sound = input()
    doctor_sound = input()

    patient_a_count = patient_sound.count('a')
    doctor_a_count = doctor_sound.count('a')

    if patient_a_count >= doctor_a_count:
        print("go")
    else:
        print("no")

evaluate_sound_duration()
```

Listing 2: GPT

```
def sore_throat_test():
    jon_aaah = input()
    doctor_aaah = input()

    if len(jon_aaah) >= len(doctor_aaah):
        return 'go'
    else:
        return 'no'

print(sore_throat_test())
```

A direct text comparison does not work in this scenario. Instead, we utilize Python's Abstract Syntax Tree (AST) and the PyASTSim Python library [13] to compare the generated solutions from the two platforms. PyASTSim first converts the Python source codes to AST trees, removes all comments and docstring, and then normalizes the identifiers. Next, the AST trees are reconverted to source code, and the differences between the source codes are measured using the Damerau-Levenshtein distance [2]. The edit distances are then converted to percentages. Table 4 provides the summary statistics of this similarity percentage scores for different statuses across different difficulty ranges. The selected problems are the ones where both platforms generate the same status. The median similarity scores for AC problems in lower difficulty ranges (0.0-2.0 and 2.0-4.0) are noticeably higher than the similarity scores for other statuses at other difficulty ranges, suggesting that simpler problems are likely to have more similar solutions.

Table 4: Summary statistics of text similarity between Gemini/ChatGPT-generated solutions

Problem Difficulty: 0.0 - 2.0					
Status	**Count**	**Median**	**Minimum**	**Maximum**	**Std. Dev.**
Accepted	65	46.000	8.000	89.000	16.039
Wrong Answer	28	39.500	0.000	69.000	16.344
Run Time Error	2	50.500	40.000	61.000	14.849
Time Limit Exceeded	0	nan	nan	nan	nan
Memory Limit Exceeded	0	nan	nan	nan	nan
Total	95				
Problem Difficulty: 2.0 - 4.0					
Status	**Count**	**Median**	**Minimum**	**Maximum**	**Std. Dev.**
Accepted	48	48.000	0.000	91.000	18.656
Wrong Answer	149	36.000	0.000	82.000	14.848
Run Time Error	24	36.000	0.000	70.000	14.940
Time Limit Exceeded	4	38.000	33.000	71.000	17.569
Memory Limit Exceeded	0	nan	nan	nan	nan
Total	225				
Problem Difficulty: 4.0 - 6.0					
Status	**Count**	**Median**	**Minimum**	**Maximum**	**Std. Dev.**
Accepted	1	24.000	24.000	24.000	nan
Wrong Answer	276	35.000	0.000	79.000	14.344
Run Time Error	45	34.000	0.000	53.000	13.056
Time Limit Exceeded	9	48.000	30.000	76.000	18.824
Memory Limit Exceeded	2	58.000	40.000	76.000	25.456
Total	333				
Problem Difficulty: 6.0 - 8.0					
Status	**Count**	**Median**	**Minimum**	**Maximum**	**Std. Dev.**
Accepted	0	nan	nan	nan	nan
Wrong Answer	268	33.000	0.000	69.000	14.221
Run Time Error	44	32.500	0.000	53.000	12.777
Time Limit Exceeded	15	40.000	22.000	61.000	11.767
Memory Limit Exceeded	2	20.500	7.000	34.000	19.092
Total	329				
Problem Difficulty: 8.0 - 10.0					
Status	**Count**	**Median**	**Minimum**	**Maximum**	**Std. Dev.**
Accepted	0	nan	nan	nan	nan
Wrong Answer	99	34.000	0.000	78.000	13.140
Run Time Error	21	33.000	0.000	54.000	12.557
Time Limit Exceeded	4	41.000	0.000	57.000	24.364
Memory Limit Exceeded	2	33.000	29.000	37.000	5.657
Total	126				

3.3 Statistical Analysis

Figures 4, provide a visual intuition regarding the correlation between ChatGPT and Gemini's performance and the problems' difficulty. Solutions with AC status concentrate primarily between difficulty levels 0 and 3. As the range of difficulty increases, the number of AC solutions declines by a visible amount across both platforms. For *WA*, *RTE*, and *TLE*, the distribution seems to visually fit with a normal distribution with a mean around 6.0 difficulty.

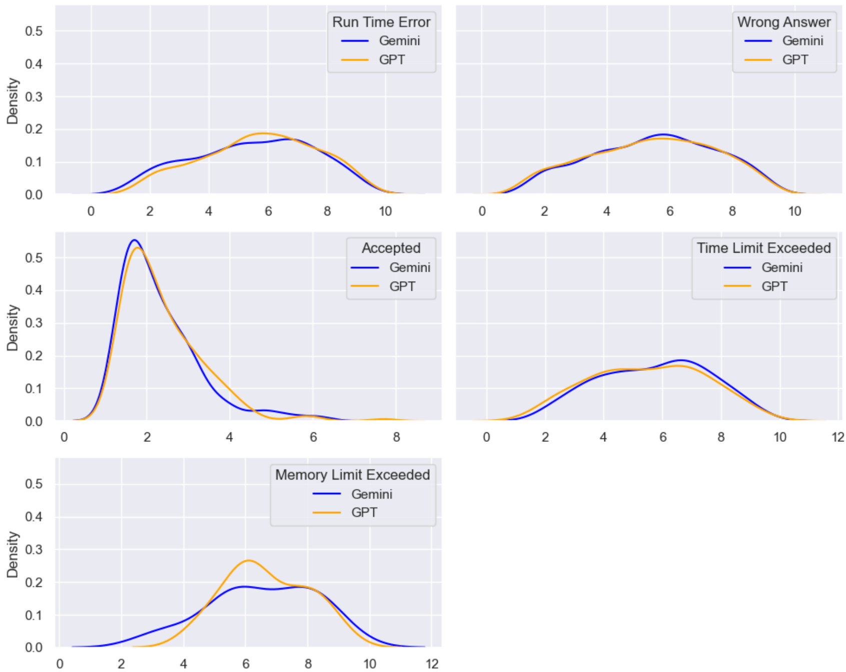


Figure 4: Difficulty Distribution across Different Status

To determine whether there is a statistically significant difference between difficulty distributions and pass-ratio distributions for different statuses across the two platforms, we first apply the Kolmogorov-Smirnov (KS) to the two platforms' set of problem difficulty scores for each status. For the KS test, the null hypothesis is that both score sets come from the same continuous distribution. Next, we apply the t-test with the null hypothesis that both score sets have the same expected value. The same set of procedures is applied to the platforms' pass ratio score for each status. The *p-value* results are shown in Table 5.

From the results, we fail to reject the null hypotheses for both tests in the cases of *AC*, *RTE*, and *MLE*. In other words, we fail to find any statistically significant proof that the distributions of solutions for these statuses from both platforms seem to be drawn from the same distribution. For *WA*, both KS test results for Difficulty and Pass Ratio scores are statistically significant (*p-value* < 0.05), while both t-test results are not. This means that while we cannot reject the null hypothesis that the expected value of these scores distributions are similar across platforms, it is statistically significant that their distributions are different. In other words, there is a significant difference between *WA*-causing solutions generated by Gemini and GPT. For *TLE*, only the null hypothesis of the KS-test for Pass Ratio scores is rejected at 0.0274.

Table 5: Summary statistics comparing Gemini solutions and GPT solutions

Status	**Pass Ratio**		**Difficulty**	
	KS test	**t-test**	**KS test**	**t-test**
Accepted	1.0	0.8294	nan	0.4187
Wrong Answer	0.0048	0.802	0.0171	0.2732
Run Time Error	0.9999	0.4498	0.2739	0.0978
Time Limit Exceeded	0.0274	0.8388	0.3818	0.3321
Memory Limit Exceeded	0.2144	0.8329	0.1534	0.7647

From the Damerau-Levenshtein distance collected in the Exploratory Analysis, we also graph the distribution of similarity scores between solutions for different statuses. In this case, we have to select a union set of solutions that generateHanoi1982! the same status for both platforms. Figure 5 indicates that with the exception of *MLE*, all other statuses have nearly identical distributions for their similarity scores. There is no overwhelmingly similar distribution, indicating some common coding structure and the majority of differences lying in the details of the code generated by the two platforms.

4 Literature Review

There have been many works focusing on studying the quality of code generated through LLM platforms [9, 4, 12, 3, 5, 11, 14]. These work utilized datasets consists of test cases and codes collected and generated previously. One of the more popular dataset is APPS, a benchmark for code generation from natural language specification [9]. It consists of 10,000 problems collected from 7 sources: *codeforces.com*, *atcoder.jp*, *www.codechef.com*, *leetcode.com*, *open.kattis.com*, *www.hackerrank.com*, and *www.codewars.com*. The input/output test cases are collected from publicly available sources. For example, test cases from *open.kattis.com* are the ones available on the problems' pages. Hidden tests are not available from APPS. Another dataset is the Most Basic Programming Problems [1]. These problems were created by crowdsourcing participants to write a short problem statement, a single self-contained Python solution, and three test cases that check for semantic correctness. The Refactory dataset contains 2442 correct and 1783 buggy programs collected from real world students' submission to an introductory programming course at a large public university [10]. FACE provides an additional alternative to these static datasets by allowing users to access authentic and extreme test cases that are not readily available to the public. This, in turn, can provide for a more rigorous study of AI-generated coding platforms.

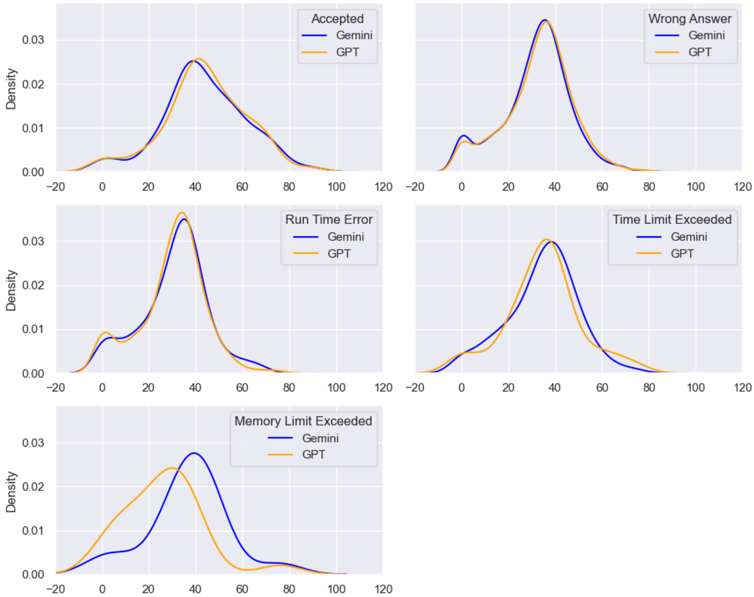


Figure 5: Similarity Score Distribution across Different Status

5 Conclusion

Through FACE, we are able to extensively collect problem statements, capture AI-generation solutions from different online platforms, and evaluate the quality of these solutions for comparison purposes. The analysis of the results demonstrate a clear difference in written convention and coding style, which in turn lead to a noticeable difference in the distribution of final evaluation status among the two platforms in our case study: OpenAI's ChatGPT and Google's Gemini. FACE provides a foundational framework from which the following future study can be carried out:

- Adding prompt engineering capability to customize and resubmit problem statements to generate better codes.
- Investigating approaches to customize and summarize problem statements to reduce token count in order to improve cost without impacting the quality of generated codes.
- Investigating the possibility of combining multiple failed codes into the problem statements to ask the platforms to generate better codes.

References

[1] Jacob Austin et al. “Program Synthesis with Large Language Models”. In: *arXiv e-prints* (2021), arXiv–2108.

[2] Bonnie Berger, Michael S Waterman, and Yun William Yu. “Levenshtein distance, sequence comparison and biological database search”. In: *IEEE transactions on information theory* 67.6 (2020), pp. 3287–3294.

[3] Bei Chen et al. *CodeT: Code Generation with Generated Tests*. 2022. arXiv: `2207.10397 [cs.CL]`.

[4] Mark Chen et al. *Evaluating Large Language Models Trained on Code*. 2021. arXiv: `2107.03374 [cs.LG]`.

[5] Carlos Eduardo Andino Coello, Mohammed Nazeh Alimam, and Rand Kouatly. “Effectiveness of ChatGPT in Coding: A Comparative Analysis of Popular Large Language Models”. In: *Digital* 4.1 (2024), pp. 114–125. ISSN: 2673-6470. DOI: `10.3390/digital4010005`. URL: `https://www.mdpi.com/2673-6470/4/1/5`.

[6] E Dreibelbis. *ChatGPT Passes Google Coding Interview for Level 3 Engineer With* $183K Salary$. 2023.

[7] Nora Dunder et al. “Kattis vs ChatGPT: Assessment and evaluation of programming tasks in the age of artificial intelligence”. In: *Proceedings of the 14th Learning Analytics and Knowledge Conference*. 2024, pp. 821–827.

[8] Tyna Eloundou et al. “Gpts are gpts: An early look at the labor market impact potential of large language models”. In: *arXiv preprint arXiv:2303.10130* (2023).

[9] Dan Hendrycks et al. “Measuring Coding Challenge Competence With APPS”. In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2021.

[10] Yang Hu et al. “Re-factoring based program repair applied to programming assignments”. In: *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2019, pp. 388–398.

[11] Jiawei Liu et al. *Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation*. 2023. arXiv: `2305.01210 [cs.SE]`.

[12] Erik Nijkamp et al. *CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis*. 2023. arXiv: `2203.13474 [cs.LG]`.

[13] *PyASTSim*. https://pypi.org/project/pyastsim/. 2021.

[14] Haoye Tian et al. *Is ChatGPT the Ultimate Programming Assistant – How far is it?* 2023. arXiv: 2304.11938 [cs.SE].

[15] Nghia D Tran et al. “Exploring ChatGPT’s Ability to Solve Programming Problems with Complex Context”. In: *Journal of Computing Sciences in Colleges* 39.3 (2023), pp. 195–209.

Strengthening Financial IoT Systems Against Bank Fraud: Integrating Data Backup and Recovery Solutions*

Syed Rizvi, Steven McKimm, Jonathan Bush,
Lukas Rhyner, Christian Diaz
Department of Information Sciences and Technology
Pennsylvania State University
Altoona, PA, USA
{srizvi, sjm7323, jwb6609, lwr114, csd5429}@psu.edu

Abstract

Bank fraud poses significant challenges for both consumers and providers alike. Protecting the privacy and security of financial data is crucial, as any breaches can result in severe financial losses and damage to reputation. When financial institutions lack adequate data backup and recovery capabilities, the impact of a data breach or cyber-attack becomes even more dire. Therefore, it's imperative to develop robust data recovery solutions to instill trust and confidence among customers. This paper delves into various controls and recommendations aimed at preventing bank fraud, particularly by integrating data backup and recovery capabilities into existing IoT infrastructure. Additionally, the paper addresses the challenges and potential risks associated with this integration. Through developed data backup and recovery plans, it is evident that proactive planning can enable financial institutions to swiftly return to normal operations with minimal data loss and disruptions.

1 Introduction

IoT devices in the financial domain revolutionize how financial services operate and deliver value to stakeholders. Devices enhance data collection and analysis, enabling real-time monitoring and asset management. Smartphones are at the forefront of IoT, and a multitude of devices designed for transactions have been created to provide a seamless customer experience [7]. Additionally, IoT devices support the creation of smart contracts in blockchain technology, ensuring secure and automated financial transactions; smart contracts can match end users with high-reputation edge servers automatically, and the end user can submit a reputation evaluation for smart contracts based on the behavior of edge servers [3]. The use of IoT in financial services also extends to predictive maintenance for ATMs and other critical infrastructure, reducing downtime and ensuring continuous service availability. In ATMs, predictive algorithms were utilized to predict failure and optimize the maintenance route [1]. Despite the benefits of IoT devices in the financial domain, they are vulnerable to various security threats which significantly impact customers and their institutions. Similarly, the immaturity of smart contracts may result in losses or leaked information [11]. These devices often lack robust security measures which make them prime targets for cyberattacks. Data breaches, hacking, and malware are significant attacks which can carry a costly impact. For customers, vulnerabilities leading to unauthorized access to sensitive financial information may result in identity theft and financial loss. For banks and financial institutions, security breaches can erode customer trust, damage reputations, and lead to significant financial penalties. Thus, the development of models that identify these costs is significant in patching security gaps to better manage consumer data [4]. Blockchain, compromised IoT devices, can undermine the integrity of smart contracts and transactional accuracy, potentially causing disruption to the financial ecosystem, so ensuring the security of IoT devices is crucial, requiring continuous updates, stringent security protocols, and proactive monitoring to protect against emerging threats and safeguard trust in the financial domain.

Addressing security issues in the financial domain involves several critical steps to ensure robustness and reliability. First, identifying security gaps through rigorous assessments and penetration testing uncovers vulnerabilities in IoT devices and networks, allowing for targeted improvements that will benefit both the customer and service provider. Developing advanced security models, often leveraging machine learning (ML), helps predict and mitigate potential threats by simulating various attack scenarios and strengthening defenses. Deep neural architecture is utilized in identifying and characterizing threats which operate on precision, recall, and F1 scores to optimize threat mitigation [2]. Hence, developing a more secure environment for operation. Additionally, creating validation software is essential for continuously monitoring IoT devices, ensuring adherence to stringent security standards and providing real-time feedback on their performance. Distributed caching is applied in validating public-key certificates which proposes utilizing cache spaces on IoT devices as a pool to store validated certificates [10]. Implementing cryptography secures data and communications with strong encryption algorithms, ensuring that even intercepted data remains inaccessible to unauthorized users. Cryptography and Lightning Network (LN) are a proposed solution for the issues with traditional blockchain transactions. This provides a delegated channel where the IoT device will open and close LN channels to complete transactions through untrusted gateways with low power consumption while ensuring security[6]. Additionally, blockchain technology enhances security by providing decentralized, tamper-proof records of transactions and smart contracts, ensuring transparency and significantly reducing the risk of fraud. Utilizing blockchain's security enhancements in the realm of IoT can help to implement a resilient framework [5]. Blockchain also finds uses in developing privacy in IoT devices. It can employ a proposed architecture where enabled devices can facilitate sharing through multiple blockchain ecosystems, ensuring exclusive, seamless access through blockchain smart contracts [9]. Finally, blockchain sees another use in a lightweight payment verification protocol on IoT devices. Utilizing a ticket-based verification protocol, blockchain transactions can be completed through a contract manager and transaction verifier which limits the necessity for high-performance embedded systems [8].

Providing comprehensive data to detail the foundation of financial IoT devices and their protection, including related data and backups, is crucial for reinforcing the integrity of modern financial systems. In a digital landscape where trust hinges on the security of information, this research highlights the critical data recovery capabilities needed for a successful restoration in the event of cybersecurity attacks. By presenting detailed information, the authors empower stakeholders to navigate cybersecurity challenges effectively, fostering a shared understanding and collaborative efforts towards resilience and innovation. Ultimately, this research serves as a practical guide for decision-making processes, reinforcing the security of financial ecosystems and ensuring robust data recovery mechanisms are in place.

2 Case Study

CJL Bank is looking to grow its customer base by boosting better fraud detection and prevention than its competitors. To achieve this, CJL is looking to integrate IoT devices into its network to better track and detect fraudulent transactions. The goal is to create a network capable of determining legitimate transactions from fraudulent transactions using ML techniques. For this case study, we have the following assumptions: (1) CJL is an international bank with customers around the world. (2) CJL bank already has current cyber security policies in place for their current IoT infrastructure. (3) CJL provides services such as online banking, ATMs, mobile banking, credit/debit cards, investment portfolios, and 24/7 customer support.

2.1 IoT Model for Financial Domain

Figure 1 represents a financial institution that is incorporating ML into their IoT devices. This model is divided into two sections, Banking and Customer assets. The banking assets include its firewall, cloud servers, and user database with ML being implemented on the cloud servers. This allows for better model training, scalability, and design simplicity while being protected by the bank's firewall. The customer assets are divided into different ways a user interacts with the bank. These include online banking, mobile banking, wearable banking, credit/debit cards, and smart wallets for digital assets.

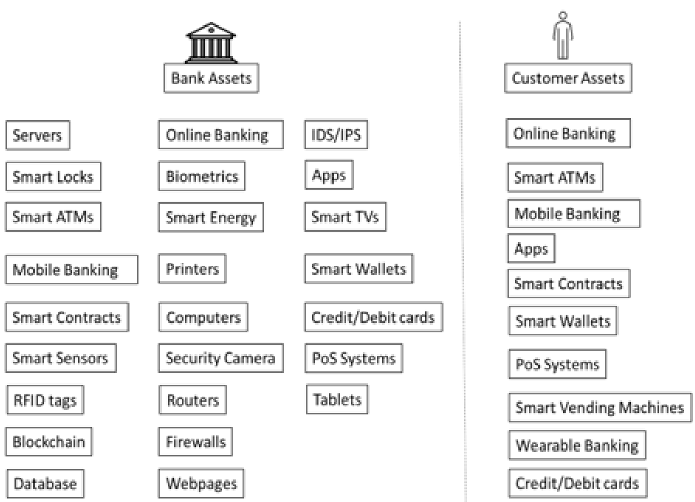


Figure 1: Financial IoT domain

2.2 Scenario

A customer has been using their debit card to get gas in California for the last several months and then a sudden charge for gas in Arizona occurs. This represents a case of credit fraud as the legitimate customer's card information has been stolen. Without ML, it would be up to the individual to prove they did not make the purchase. With the use of ML, the algorithm can detect this drastic change in purchasing location to prevent the transaction, and then place a "hold" on the card. Once the fraudulent charge has been prevented, a bank representative can contact the account holder based on the contact information recorded on file. This will then determine if the transaction was legitimate and allow the charge or fraudulent and place a "freeze" on the account. The incorporation of ML will enable data to be collected on customers and construct a baseline of activity. After establishing a baseline, outliers can be detected to prevent fraudulent activity on customer accounts. The algorithm trains itself based on the customer's actions using location data, online interactions, device information, credit history, customer behavior, and transaction data. After enough data has been collected and analyzed, the algorithm can then predict how a customer generally interacts with the banking system.

3 Outcome of Risk Management Framework

The risk management framework is presented in Fig. 2 which provides an effective methodology for assessing IoT devices used in the financial domain, ensuring a comprehensive approach to risk management.

Figure 2: Risk Management Framework

Utilizing this framework is a crucial step before performing a Business Impact Analysis (BIA) and developing data backup and recovery capabilities, as it lays the groundwork for understanding and mitigating potential threats. By systematically managing risks, financial institutions can enhance the security and reliability of their IoT infrastructure.

3.1 Risk Assessment

The risk assessment process highlights the multifaceted nature of risks associated with integrating ML into IoT infrastructure for fraud prevention at CJL Bank. Identified risks encompass financial losses from fraudulent transactions, potential data breaches compromising sensitive customer information, and operational disruptions affecting regular banking operations and customer experience. By systematically identifying these risks, the assessment underscores the need for a proactive approach to risk management. Moreover, it emphasizes the importance of evaluating the likelihood and impact of each risk, considering factors such as the criticality of banking functions, the sensitivity of customer data, and the potential for customer dissatisfaction. This holistic risk assessment approach provides a foundation for implementing targeted mitigation strategies and ensuring the resilience of CJL Bank's operations against evolving threats.

3.2 Risk Identification

There is a risk of financial loss if the ML algorithms fail to accurately identify and prevent fraudulent transactions. With the integration of IoT devices and the storage of large volumes of user data, there is an increased risk of data breaches, exposing sensitive customer information. Implementing changes to online or mobile banking for account holds may lead to disruptions in regular banking operations and customer inconvenience. It is essential to identify all IoT devices integrated into banking operations. This includes cataloging devices such as smart ATMs, card readers, biometric authentication systems, and sensors for environmental monitoring. Understanding the purposes served by each device, whether transaction processing, customer authentication, or facility management, helps assess their criticality to banking operations. Identifying vulnerabilities and exploits is another essential aspect of IoT device assessment. Conducting vulnerability assessments and staying informed about known vulnerabilities targeting IoT devices enables banks to assess the risk of exploitation and take appropriate mitigation measures.

3.3 Risk Evaluation

Risk evaluation involves assessing various factors, such as potential financial losses and determining acceptable risk levels. This includes comparing the cost of implementing machine learning (ML) against potential fraud losses and evaluating the likelihood and impact of data breaches involving sensitive customer information. Implementing mitigation strategies is essential. Consider operational disruptions and their impact on customer satisfaction and develop strategies to minimize these during implementation. Evaluating IoT device security features, such as authentication, encryption protocols, and access controls, is crucial to prevent unauthorized access and data breaches. Additionally, analyzing connectivity options and data transmission protocols helps understand security implications. Lastly, assess vendor security practices by reviewing their certifications, adherence to industry standards, and responsiveness to security vulnerabilities through timely updates.

3.4 Risk Prioritization

Risk prioritization is a critical step in effectively managing the integration of ML into IoT infrastructure for bank fraud prevention. It is essential to evaluate the potential impact of various risks on financial loss, data security, operational continuity, and customer trust. High-priority risk is the potential for data breaches compromising sensitive customer information. A data breach could not only result in financial losses but also damage the bank's reputation and erode customer trust. Therefore, implementing data security measures, such as encryption, access controls, and regular audits, is essential to mitigate this risk effectively. Operational disruptions in online and mobile banking systems also rank high in terms of risk prioritization. Changes to these systems to accommodate ML-driven fraud prevention measures may introduce technical challenges or errors, leading to service interruptions or delays. Such disruptions could inconvenience customers, undermine their confidence in the bank's services, and potentially result in financial losses if transactions are not processed timely or accurately. Therefore, proactive measures to identify and mitigate potential operational risks, such as conducting pilot tests, extensive staff training, and effective communication with customers, are essential to ensure smooth implementation and minimize disruptions.

Additionally, the risk of customer dissatisfaction stemming from false positives or account holds should not be overlooked. Balancing the need for fraud prevention with the desire to provide seamless banking experience is crucial. Therefore, strategies to promptly address false positives, such as customer verification processes and clear communication channels, are essential to maintain customer satisfaction and trust in the bank's services. By prioritizing these risks and implementing appropriate mitigation measures, CJL Bank can effectively navigate the challenges associated with integrating ML into its IoT infrastructure for fraud prevention while safeguarding its operations and reputation. Table 1 provides a summary of the risk assessment and prioritization

IoT Devices	Risk Assessment	SLE Score	Risk Mitigation (RM)	Overall Risk Prioritization
Smart ATMs	High likelihood of exploitation due to frequent customer interactions.	High	Implement real-time transaction monitoring and encryption for secure transactions.	1
Card Readers	Vulnerable to skimming attacks leading to unauthorized card data access.	High	Regularly inspect and maintain card readers, implement chip-based authentication, and deploy anti-skimming technology.	2
Biometric Authentication Systems	Risks associated with biometric data theft leading to identity theft. Potential compromise of customer identity and privacy.	Medium	Implement encryption for biometric data storage, conduct regular security audits.	3
Environmental Monitoring Sensors	Low likelihood of exploitation but potential impact on system integrity if compromised. Potential disruptions to facility management systems.	Low	Implement access controls for sensor data, monitor for anomalies, and ensure physical security of sensor installations.	4

Table 1: Summary of Risk Prioritization and Mitigation Strategies for Financial IoT

of various financial IoT devices, highlighting their risk levels, Single Loss Expectancy (SLE) scores, and recommended risk mitigation strategies. Each device is ranked based on the likelihood of exploitation and the potential impact of such exploitation.

3.5 Risk Analysis and Control

Implement real-time monitoring and response mechanisms to quickly identify and prevent fraudulent transactions. Establish a customer verification process to resolve false positives promptly. Implement strong encryption protocols, regular security audits, and employee training programs to reduce the likelihood of a data breach. Ensure compliance with data protection regulations. Develop a phased implementation plan, conduct extensive training for bank staff and customers, and provide effective communication channels for issue resolution. The efforts to enhance fraud detection through ML integration, risk acceptance involves acknowledging certain inherent risks without actively attempting to mitigate or transfer them. For instance, the bank might accept a certain level of financial risk associated with potential false positives in the ML algorithms, understanding that a strict approach might inconvenience legitimate customers. This acceptance could be based on a thorough evaluation of the cost-benefit analysis, where the expense of eliminating all risks may outweigh the benefits. Outsourcing specific aspects of fraud detection to specialized third-party service providers or forming contractual agreements with vendors for liability distribution are examples of risk transference strategies that CJL Bank will adopt to alleviate the potential financial and operational consequences associated with these risks. While risk transference can be a valuable component of a comprehensive risk management strategy, it should be complemented by other proactive risk management measures, including continuous monitoring, evaluation, and adaptation of security protocols.

4 BIA for IoT Financial Domain

The outcome of Risk Management provides valuable information that can be effectively utilized as part of performing a BIA. Each identified risk corresponds to a potential impact on specific business functions, enabling the BIA team to pinpoint areas vulnerable to disruptions. The section assists in assessing potential impacts by detailing the consequences of identified risks such as financial losses, data breaches, and operational disruptions. Understanding the severity and magnitude of these impacts is crucial for quantifying the effect of disruptions on revenue, reputation, customer service, and regulatory compliance, which are integral aspects of BIA.

The risk prioritization and device ranking sections provide insight into the criticality of different IoT devices and associated risks. This aligns with BIA's objective of prioritizing recovery objectives based on the criticality of business functions and their dependencies on IT systems and infrastructure. The outcome of risk management aids in identifying dependencies and interdependencies between business functions, processes, and IT systems, essential for developing comprehensive recovery strategies. For instance, understanding how disruptions in online/mobile banking systems can affect transaction processing, customer service, and revenue generation facilitates better planning for recovery. This section assists in estimating recovery time objectives (RTO) and recovery point objectives (RPO) by providing insights into potential downtime and data loss associated with different risks. This information is essential for setting realistic recovery objectives and ensuring timely restoration of critical business functions in the event of disruptions.

4.1 Recovery Criticality for Business Processes

The outcome of this step is a comprehensive understanding of the mission-critical and essential business processes within CJL Bank. These processes include activities related to banking operations, customer service, transaction processing, and fraud detection and prevention. By mapping out these processes, the bank can prioritize which functions are most critical to its operations and require immediate recovery in the event of a disruption. As a result, CJL Bank identifies key business processes such as online banking, ATM operations, transaction processing, customer support, and fraud detection as mission critical. Each process is assessed for its recovery criticality, considering factors such as financial impact, regulatory compliance, customer trust, and legal obligations.

4.2 Identify Resource Requirements

The outcome of this step is the identification of the resources necessary to support the recovery of mission-critical processes. These resources encompass both tangible assets (such as IT infrastructure, data storage, and physical facilities) and intangible assets (such as employee expertise and vendor support). By understanding resource requirements, CJL Bank can ensure adequate preparedness for potential disruptions. As a result, CJL Bank conducts a thorough assessment of resource requirements for each critical business process. This includes identifying the IT systems, hardware, software, data backups, communication channels, and personnel needed to resume operations. Additionally, the bank evaluates external dependencies, such as third-party vendors and regulatory compliance requirements, to ensure comprehensive recovery planning.

4.3 Identify Recovery Priorities for System Resources

The outcome of this step is the establishment of recovery priorities for system resources based on the criticality of business processes. This involves determining the sequence in which resources will be allocated and restored during the recovery phase to minimize downtime and mitigate business impact. As a result, CJL Bank establishes clear recovery priorities for system resources based on the criticality of each business process. High-priority resources, such as core banking systems, fraud detection algorithms, and customer communication channels, are identified for immediate restoration. Lower-priority resources, such as non-essential applications or services, may be deferred until critical functions are restored. This prioritization ensures efficient resource allocation and minimizes the impact of disruptions on essential operations. The outcomes of each step of the BIA process enable CJL Bank to effectively prioritize its recovery efforts, allocate resources efficiently, and ensure the continuity of critical business operations in the face of potential disruptions. By understanding the mission-critical processes, resource requirements, and recovery priorities, the bank can enhance its resilience and responsiveness to various threats, including cyberattacks, natural disasters, and operational failures.

4.4 Recommendations for Contingency Policies/Plans

Incident response plan: CJL Bank's IRP is designed to quickly detect and mitigate incidents from the integration of ML into our IoT infrastructure for fraud prevention with a focus on minimizing downtime and data loss. The roles and responsibilities of the incident teams consist of IT, cybersecurity, legal, and management personnel to meet the RTO timeline. Continuous monitoring is to be conducted to ensure incorporating ML does not result in an incident with unnecessary downtime to critical systems. The RTO set is near zero as excessive downtime is not acceptable and critical systems must be returned to functional status as soon as possible.

Disaster Recovery Plan: CJL Bank's DRP focuses on the rapid recovery of critical systems and services. Based on the BIA and RA, data backups are conducted every 5 minutes to reduce the potential for loss of data. These backups will result in rapid recovery in the case of an incident. CJL Bank operates multiple servers for redundancy so if one should go offline, the other will immediately take over until the primary server can be returned to operational status.

Business Continuity Plan: CJL Bank's BCP is to ensure the uninterrupted operation of critical business functions and to align with our zero-downtime objective. The critical business processes include online banking, ATM operations, transaction processing, customer support, and fraud detection. These processes have short RTO times and need to be restored immediately. The RTO is set to 5 minutes, with WTR being 10 minutes for a total WTD of 15 minutes. In the event of transitioning to an alternate site, customer support will be handled by phone calls being redirected to the bank's call center rather than the specific branch. CJL Bank consists of multiple branches so day-to-day functions will be diverted to another branch location until the primary location is restored. Each branch contains a local server that can be integrated into the banking network should one go offline; another branch will automatically receive the data traffic.

Table 2 highlights the critical components and tolerance parameters of CJL Bank's. Understanding the RPO, RTO, WRT, and Maximum Tolerable Downtime (MTD) is essential for developing an effective data backup and recovery plan for the financial domain. By defining these parameters, financial institutions can ensure that they have a robust framework in place to minimize data loss, swiftly recover critical systems, and maintain continuous operation during and after incidents. Integrating these tolerance parameters into the data backup and recovery strategy helps financial institutions align their recovery efforts with their operational and compliance requirements, ensuring resilience in the face of potential disruptions.

Plan	Key Objectives	Roles and Responsibilities	Key Features	RPO	RTO	WRT	MTD
Incident Response Plan (IRP)	Quickly detect and mitigate incidents from ML integration in IoT for fraud prevention.	IT, cybersecurity, legal, and management personnel to meet the RTO timeline.	Continuous monitoring to prevent unnecessary downtime.	Near zero	Near zero	N/A	N/A
Disaster Recovery Plan (DRP)	Rapid recovery of critical systems and services.	IT and management personnel for ensuring data backup and server redundancy.	Data backups every 5 minutes.	5 min	5 min	N/A	N/A
Business Continuity Plan (BCP)	Ensure uninterrupted operation of critical business functions.	IT, customer support, and branch management.	Short RTO and WRT Multiple branches with local servers to manage data traffic	5 min	5 min	10 min	15 min

Table 2: Summary of CJL Bank's IR, DR, and BC with Tolerance Parameters

5 Data Protection and Recovery Criticality

The data classification is based on public, internal, confidential, and restricted. Most data should be classified as confidential if it handles users' personal identifiable information (PII). The classification will be based on the highest level of classification the document handles or contains to prevent under classifying data. Internal data will be of that that stays within the banking network. The retention schedule for individual and banking records will be retained and stored for 2 years before being archived for an additional 3 years. This covers all confidential and restricted data. Internal only data will be stored for one year. This is to maintain record keeping of employee logs and video evidence in the event of threat of incidents. No public data will be stored, as once it receives PII, it will be classified as confidential. The main governing factors for our organization are HIPAA, PCIDSS, and the bank protection act. These are designed to protect users PII and must be upheld to prevent lawsuits and damages

5.1 Real-time Protection and Server Recovery

The critical assets of the bank are backed up using mirroring for real-time protection. This creates multiple exact copies of data to prevent data loss and interruption in real-time. Mirroring is the preferred method since we deal with financial records and data needs to be captured and stored in real-time to prevent losses. The use of mirroring also serves as the recovery method for minor incidents in conjunction with back up from major outages. Server recovery will be handled by the additional servers classified as warm servers. In the event the main server goes offline, the warm server will become the temporary hot server. The downed server will then transition to a cold server status to repaired and tested. Once testing has been completed and there are no issues, the server will then function as the hot server again

5.2 Application and Site Recovery

Clustering is used for application recovery. This recovery method is in conjunction with the hot and warm servers. Should the primary server fail, the applications will then run on the warm server. The two servers are configured identically so if one application fails, all applications will then switch to the warm server. The application will continue to run on the warm server until the hot server is repaired. Site recovery method will be transitioning to a mirrored site. There are multiple branch locations in our network. Should a location become unusable, all operations will be transferred to an alternative location until the site can be reopened. Since all branches operate the same, no additional resources will need to be transitioned. Critical employees will continue to work from home using their work computer. Should the expected downtime of the site be greater than five days, the money on site will be transferred to the alternative location and stored in the safe.

5.3 Recovery Criticality in Financial Domain

Backup Creation: Table 3 provides an overview of different IT components that are found in our business. Table 3 shows the type of data backup, frequency of backup, backup methods, backup strategy, and backup locations for each IT component.
Backup Verification: Table 4 outlines different IT components that are found in our business. The tables include the following for each IT component: Verification method, integrity, testing backup method, frequency of verification, frequency of integrity check, and frequency of testing.
Data Storage and Data Encryption: Each component requires careful consideration of storage, access control, data retention, backup iterations, and encryption methods to ensure security, compliance, and operational efficiency. Servers housing critical banking data should be stored in secure data centers with redundant storage, access controlled through role-based permissions, and data encrypted both at rest and in transit. Regular backups, including incremental backups for transactional data, are essential to mitigate the risk of

	Backup Creation				
IT Components	**Type of Data Backup**	**Frequency of Backup**	**Backup Method**	**Backup Strategy**	**Backup Location**
Servers	Full	Daily	Cloud	Incremental Backup	Offsite
Smart Locks	Configuration	Weekly	Local	Differential Backup	Onsite
Smart ARTs	Artwork	Monthly	Cloud	Full Backup	Offsite
Mobile Banking	Transaction	Real-time	Cloud	Real-time Replication	Cloud
Smart Contracts	Contracts	Daily	Local	Incremental Backup	Onsite
Smart Sensors	Sensor Data	Hourly	Cloud	Continuous Backup	Cloud
RFID Tags	Inventory Data	Weekly	Local	Full Backup	Onsite
Blockchain	Ledger	Real-time	Cloud	Real-time Replication	Cloud
Databases	Data	Daily	Local	Incremental Backup	Onsite
Biometrics	Authentication Data	Daily	Local	Incremental Backup	Onsite
Tablets	Device Data	Weekly	Local	Full Backup	Onsite
Computers	System Data	Daily	Cloud	Incremental Backup	Offsite
Security Cameras	Footage	Daily	Local	Differential Backup	Onsite
Credit/Debit Cards	Transaction Data	Real-time	Cloud	Real-time Replication	Cloud
PoS Systems	Sales Data	Daily	Cloud	Incremental Backup	Offsite

Table 3: IT Component Backup Recommendations

data loss. Smart devices like locks and ATMs require similar security measures, with data encrypted and stored locally on devices and backed up in real-time to centralized servers. Mobile banking apps demand strong encryption for data transmission and storage, multi-factor authentication for user access, and regular backups to protect against data loss. Blockchain technology is utilized for smart contracts, ensuring tamper-proof data storage with decentralized access controls and cryptographic hashing.

	Backup Verification			
IT Components	**Verification Method**	**Integrity Method**	**Testing Backup Method**	**Frequency of Testing**
Servers	Manual audits, automated scans	Hashing, checksums	Backup & restore tests	Monthly
Smart Locks	Manual checks, firmware updates	Digital signatures	Backup codes	Bi-annually
Smart ARTs	Automated logs analysis	Blockchain	Redundant systems	Weekly
Mobile Banking	Penetration testing	Encryption algorithms	Cloud backups	Daily
Smart Contracts	Code reviews, auditing	Distributed ledger tech	Offsite data storage	Monthly
Smart Sensors	Sensor calibration checks	Data hashing	Secondary data storage	Bi-monthly
RFID Tags	Reader integrity checks	Tag authentication	Redundant databases	Quarterly
Blockchain	Consensus algorithms	Immutable ledger	Node replication	Continuous
Databases	Database audits	Data encryption	Database replication	Weekly
Online Banking	User authentication checks	SSL/TLS	Backup servers	Daily
Biometrics	Biometric data encryption	Biometric templates	Biometric database backup	Monthly
Tablets	Device management tools	Encryption	Cloud sync	Weekly
Computers	Antivirus scans	Data encryption	System image backups	Daily
Security Cameras	Footage review	Encryption	Cloud storage	Weekly
Smart Wallets	Transaction logs	Encryption	Offline backups	Continuous
Credit/Debit Cards	Transaction monitoring	Chip authentication	Fraud detection systems	Continuous
PoS Systems	Transaction audits	Point-to-point encryption	Backup terminals	Daily

Table 4: IT Component Backup Verification Recommendations

6 Conclusion

This paper has highlighted the importance of effective risk management strategies in strengthening fraud detection and prevention efforts while protecting customer data within financial institutions. Through a careful analysis of challenges and proposed recommendations, our research emphasizes the critical need for a comprehensive approach. Key among these strategies is conducting a thorough BIA focused on IoT devices used in financial transactions. Our contributions extend to promoting regular audits, multiple-factor authentication, thorough employee training, and the strategic use of ML algorithms to enhance the privacy and security of customer data. Additionally, developing customized data backup and recovery plans for each IoT device emerges as a crucial step to mitigate the risks of potential disruptions or data loss incidents. By adopting these proactive measures, financial institutions can strengthen their resilience against evolving threats, ensuring continuous operations, and maintaining customer trust in an increasingly digital landscape.

References

[1] L. Calle-Sarmiento et al. "Neural Networks and Genetic Algorithms applied to the Maintenance Process in an ATM Network". In: *2022 IEEE Sixth Ecuador Technical Chapters Meeting (ETCM)*. Quito, Ecuador, 2022, pp. 1–7.

[2] R. Pratap Singh Chauhan et al. "Optimizing IoT Threat Mitigation with Artificial Intelligence in Banking: A Multi-Objective Approach". In: *2024 2nd International Conference on Disruptive Technologies (ICDT)*. Greater Noida, India, 2024, pp. 296–301.

[3] H. Cheng et al. "Trusted Resource Allocation Based on Smart Contracts for Blockchain-Enabled Internet of Things". In: *IEEE Internet of Things Journal* 9.11 (June 2022), pp. 7904–7915.

[4] C. H. O. Huamán et al. "Critical Data Security Model: Gap Security Identification and Risk Analysis In Financial Sector". In: *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*. Madrid, Spain, 2022, pp. 1–6.

[5] Z. A. Khan and A. S. Namin. "The Applications of Blockchains in Addressing the Integration and Security of IoT Systems: A Survey". In: *2021 IEEE International Conference on Big Data (Big Data)*. Orlando, FL, USA, 2021, pp. 2421–2426.

[6] A. Kurt et al. "LNGate22: Secure Bidirectional IoT Micro-Payments Using Bitcoin's Lightning Network and Threshold Cryptography". In: *IEEE Transactions on Mobile Computing* 23.5 (May 2024), pp. 6027–6044.

[7] V. P. Parandhaman. "A Secured Mobile Payment Transaction Handling System using Internet of Things with Novel Cipher Policies". In: *2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*. Chennai, India, 2023, pp. 1–8.

[8] A. Pouraghily and T. Wolf. "A Lightweight Payment Verification Protocol for Blockchain Transactions on IoT Devices". In: *2019 International Conference on Computing, Networking and Communications (ICNC)*. Honolulu, HI, USA, 2019, pp. 617–623.

[9] A. Pouraghily et al. "Poster Abstract: Privacy in Blockchain-Enabled IoT Devices". In: *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. Orlando, FL, USA, 2018, pp. 292–293.

[10] M. Wang et al. "Collaborative Validation of Public-Key Certificates for IoT by Distributed Caching". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. Paris, France, 2019, pp. 847–855.

[11] Q. Zhou et al. "Vulnerability Analysis of Smart Contract for Blockchain-Based IoT Applications: A Machine Learning Approach". In: *IEEE Internet of Things Journal* 9.24 (Dec. 2022), pp. 24695–24707.

Programming and Control of Physical Autonomous Robots via ROS 2*

Lili Ma[1], Christian Rosa[1], Xiaohai Li[1], Yu Wang[1]
Benito Mendoza[1], Andy S. Zhang[2]
[1]Deptartment of Computer Engineering Technology
[2]Deptartment of Mechanical Engineering Technology
CUNY-New York City College of Technology
New York, NY 11201

Abstract

This paper describes two exemplary projects on physical ROS-compatible robots (i.e., Turtlebot3 Burger and Waffle PI) for an undergraduate robotics course, aiming to foster students' problem-solving skills through project-based learning. The context of the study is a senior-level technical elective course in the Department of Computer Engineering Technology at a primarily undergraduate teaching institution. Earlier courses in the CET curriculum have prepared students with programming skills in several commonly used languages, including Python, C/C++, Java, and MATLAB. Students' proficiency in programming and hands-on skills make it possible to implement advanced robotic control algorithms in this robotics course, which has a 3-hour companion lab session each week.

The Robot Operating System (ROS) is an open-source framework that helps developers build and reuse code between robotic applications. Though mainly used as a research platform, instructors in higher education take action in bringing ROS and its recent release of ROS 2 into their classrooms. Our earlier work controlled a simulated robot via ROS in a virtual environment on the MATLAB-ROS-Gazebo platform. This paper describes its counterparts by utilizing physical ROS-compatible autonomous ground robots on the MATLAB-ROS2-Turtlebot3 platform.

The two exemplary projects presented in this paper cover sensing, perception, and control which are essential to any robotic application. Sensing is via the robot's onboard 2D laser sensor. Perception involves pattern classification and recognition. Control is shown via path planning. We believe the physical MATLAB-ROS2-Turtlebot3 platform will help to enhance robotics education by exposing students to realistic situations. It will also provide opportunities for educators and students to explore AI-facilitated solutions when tackling everyday problems.

1 Introduction

Robotics engineering is a multidisciplinary field built upon electrical, mechanical, and computer engineering. It deals with designing, building, operating, and engineering robots and robotic systems based on theoretical understanding and practical application. From its inception, robotics has been an inherently interdisciplinary field, bringing together diverse domains such as engineering, cognitive science, computer science, and knowledge from social sciences and humanities [9]. When teaching robotics in higher education, it is thus important to keep up with the latest developments in robotics as well as many related fields such as Artificial Intelligence (AI), data science, computer vision, Internet of Things (IoT), and Cybersecurity.

The context of this study is an undergraduate robotics course offered as a technical elective in the Department of Computer Engineering Technology (CET) at a primarily four-year teaching institution. Preceding courses in the CET curriculum have equipped students with knowledge and skills in mechatronics, embedded systems, programming, and cyber-physical systems. This robotics course aims to provide hands-on experience working with complex computer-controlled systems that integrate physical components such as sensors and actuators.

Autonomous mobile robots with a simple arm on top were used as the physical robotic platform for this course, built from VEX robotic kits using the Cortex microcontroller that comes with the kit [5, 6]. Driven by the need to also serve the recently approved Software Engineering Technology (SET) curriculum, we have been exploring robotic systems that allow for complex software and hardware integration.

This paper describes the development of two exemplary projects using the Turtlebot3 robots. The experimental setup involves a host computer running MATLAB and a physical robot (Turtlebot3 Burger or Waffle PI), each being a ROS node on the ROS network. ROS stands for the Robot Operating System [11], which is a set of open-source software libraries and tools that help researchers and robotic engineers build robot applications. ROS has been widely used by researchers and developers to build and reuse code between robotics applications. However, due to the demanding requirements of C++/Python/Java programming skills and familiarity with Linux, the adoption of ROS in an undergraduate curriculum is still rare. Recently, MathWorks released its ROS Toolbox [2], making it easier to interact with both simulated robots [3] and physical ROS-supported robots (see Fig. 1).

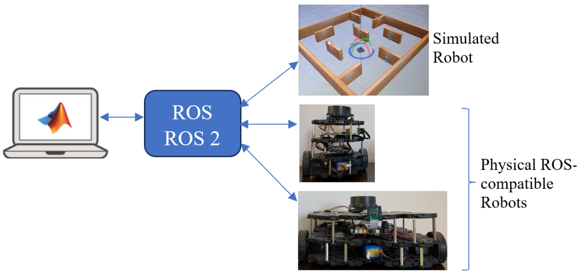


Figure 1: Connecting to ROS robots using MathWork's ROS Toolbox.

Our earlier work in [8, 7] explored the usage of ROS (particularly, ROS 1) for programming and control of a simulated robot in the Gazebo simulation environment [Fig. 2 (left)]. This paper presents its counterpart and extension, which programs and controls a physical ROS-compatible robot (Turtlebot3 Burger or Waffle PI) on ROS 2 [Fig. 2 (right)]. Note that ROS 2 is the second generation of ROS representing a step forward in the robotic framework [12]. While getting access to the robot's onboard sensors may be different due to the available topics and their types, control algorithms that were developed for the simulated robot can be readily applied to the physical robots, demonstrating the re-usability of the codes & algorithms.

The physical robotic platform, i.e., the MATLAB-ROS2-Turtlebot3 platform, utilizes benefits from both sides of MATLAB and ROS 2:

- The physical ROS-compatible robot provides students with experience in authentic Linux operating systems and ROS 2 programming.
- MATLAB already has many other toolboxes dedicated to education and research (such as computer vision toolbox, artificial intelligence toolbox, and machine learning toolbox). These toolboxes will significantly shorten the algorithm development curve.
- Both Mathworks and ROS have well-established mechanisms for developers to share their codes and experience. This allows students to use support and resources from the community.

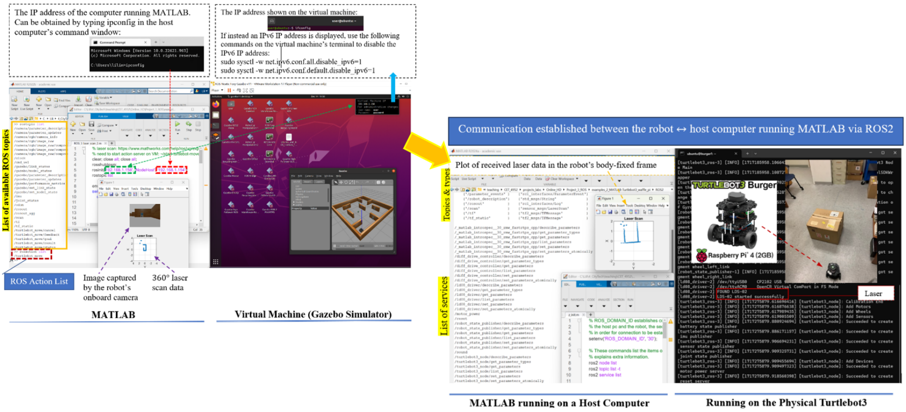


Figure 2: Using MATLAB to control the simulated robot (left) and physical robot (right).

- Students can collect and save data from the robot's onboard sensors during classes. They can conduct post-processing and refine their algorithms outside of the classroom.

The objective of this paper is to explore the feasibility of using the MATLAB-ROS2-Turtlebot3 platform to enhance robotic education and undergraduate research. The subsequent sections of this paper describe the operation on the robot side, communication with and control of the robot issued from the host computer (the MATLAB side), and two exemplary projects that incorporate sensing, perception, and control.

2 Turtlebot3 Startup and Operation

Before implementing our algorithms to control the robot, we initially learned how to operate it by running the pre-installed programs. It is worth noting that the results outlined in this paper were achieved on Turtlebot3 Burger, but all algorithms can be applied to the Turtlebot3 Waffle PI due to ROS compatibility. As an illustration, the following presents the results of running the pre-installed SLAM program on Burger and provides the necessary sequence of operations under ROS 1.

SLAM, short for Simultaneous Localization And Mapping, is a common topic for autonomous mobile robots, as seen in commercially available robots like robot vacuums. The Turtlebot3 Burger employs SLAM using its onboard 2D range sensor, the LiDAR (Light Detection and Ranging), mounted on top of the robot. The LiDAR rotates 360 degrees at a speed of 6 rotations per second. It emits a laser beam to measure the closest objects at roughly an incremental angle of 1.5 degrees. The data gathered in this process can be used to discern between walls and other objects, aiding Turtlebot3 in map construction and navigation.

Figure 3 shows the results of the robot performing SLAM in a room, where the green points denote LiDAR scan data, the black area represents walls, and the grey area shows the movable area (i.e., open space).

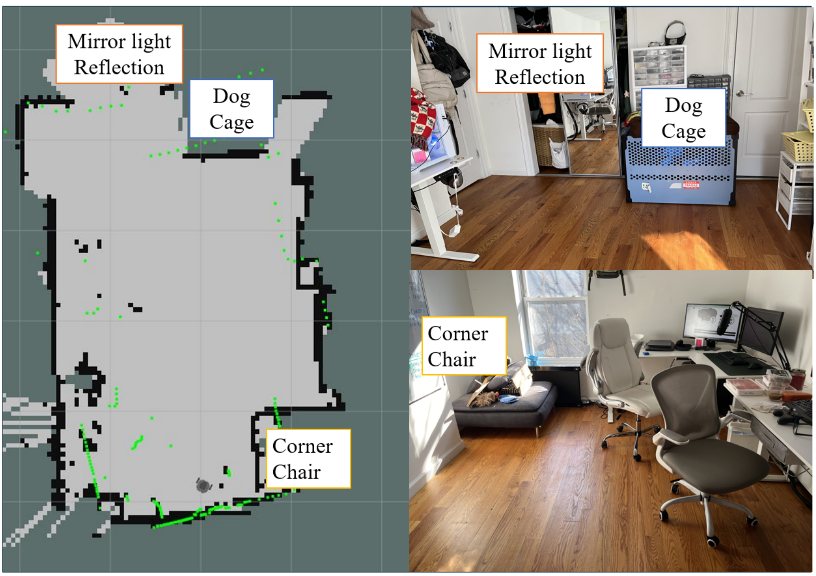


Figure 3: Turtlebot3 SLAM.

The Turtlebot3 operates on Linux. Following the installation instructions as given in Turtlebot3's e-manual [10], a server version is installed on the Raspberry PI on the robot. To use any graphic visualization tools, we installed a Virtual Machine (VM) on the host computer to run the desktop version of Linux. A Virtual Machine is a digital replica of a physical computer, enabling one to emulate Linux on a Windows system. We then need to select the ROS version. As each ROS version runs on different Linux versions and many ROS versions were no longer maintained, we opted for ROS 1 Noetic, which allows fully functioning SLAM and Navigation features as shown in Fig. 4.

To initiate the SLAM task, we began by establishing the communication/-connection between Turtlebot3 and the host computer by configuring them on the same network. We then executed the startup sequence on the Turtlebot3 robot. On the host computer, we started the Virtual Machine, from where we run three terminals each of which handles different tasks:

- Referring to Fig. 4(a), the terminal located in the upper left connected to the Turtlebot3 robot. This terminal facilitated a wireless connection to the Turtlebot3 Operating System, enabling us to operate the robot without requiring peripherals like a monitor, mouse, and keyboard.
- The terminal positioned in the upper right served as the 'roscore' terminal. 'roscore' encompasses a set of nodes and essential programs necessary for a ROS1-based system. Under ROS 1, it is necessary to have 'roscore' up-running to facilitate communication among ROS 1 nodes.
- The terminal located at the bottom is dedicated to Teleoperation, a pre-installed program. This terminal enabled us to navigate the robot using the keyboard on the Virtual Machine.
- The above three terminals collectively enable the operation of the Turtlebot3. An additional terminal will initiate the SLAM functionality [Fig. 4(b)].

(a) Startup (b) SLAM

Figure 4: Turtlebot3 startup and operation on ROS 1.

As shown in Fig. 4, using pre-installed programs, we have successfully controlled the Turtlebot3 Burger to move and navigate in an environment, collect laser data, and use the collected laser data to aid in mapping and navigation. These investigations demonstrate that the robots were correctly assembled and that all operating systems, libraries, and software were properly installed. We are now ready to control the robots using our programs.

3 Control the TurtleBot3 using MATLAB

The MATLAB-ROS2-Turtlebot3 experimental platform was set up by following the Quick Start Guide of the ROBOTIS e-Manual [10] for the construction and installation of the robot. On the host computer that runs MATLAB, MathWork's ROS Toolbox is required and installed. Successful installation and communication between these two ROS nodes is shown in the right figure in Fig. 2. Some details are:

- On the robot's processor (Raspberry PI), we installed Ubuntu Server 22.04 and ROS 2 Humble Hawksbill. We used SSH to access the Raspberry PI and brought up basic packages to start TurtleBot3 applications.
- The ROS domain ID on MATLAB needs to be set the same as that of the robot so that these two can establish a connection in between. For example, we used 30.
- Once communication is established (by using the same Domain ID), the command "ros2 node list" lists all nodes on the ROS 2 network; "ros2 topic list -t" lists all available topics and their types; and "ros2 service list" lists all available services. These commands help to confirm that communication/connection has been successfully established.
- Figure 5 shows sample MATLAB codes that subscribe to topics (odometry and laser scan data), as well as establish a publisher that modifies the robot's linear & angular velocities.
- Visualization of the ROS 2 graph is displayed via the "rqt" tool.

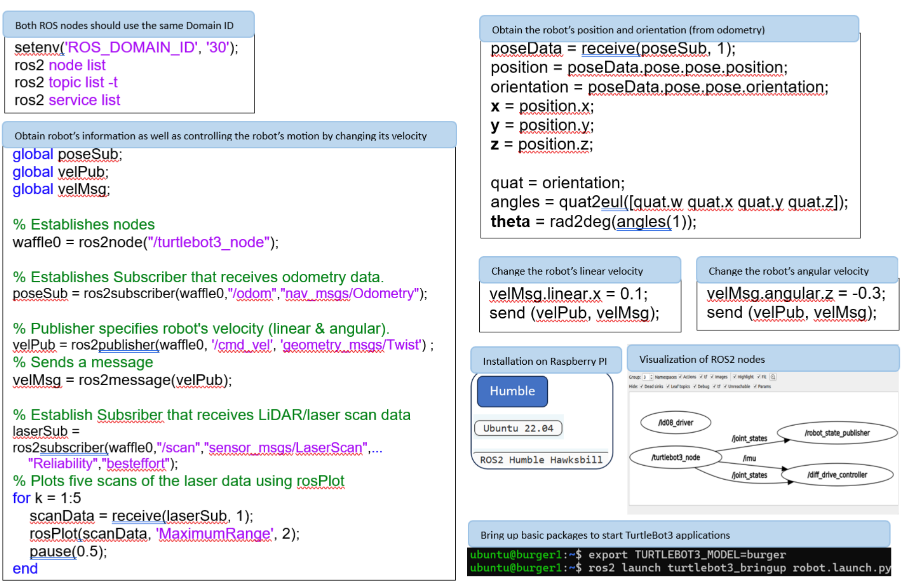


Figure 5: Sample MATLAB codes.

After successfully obtaining raw sensor data from the robot and modifying its behavior, the next task is to develop algorithms for perception and decision-making. This will be done on the MATLAB side, as demonstrated via two exemplary projects in Secs. 4 and 5.

4 Path Planning

This project provides a simple scenario for students to implement algorithms that allow an autonomous mobile robot to work in an unknown environment. The robot is assumed to have an onboard range sensor. Particularly for the Turtlebot3 robots, its 360 Laser Distance Sensor (LDS-02) is a 2D laser scanner capable of sensing 360 degrees around the robot. Initially, the robot will explore its surroundings, getting to “know” its environment by sensing and recording its laser data.

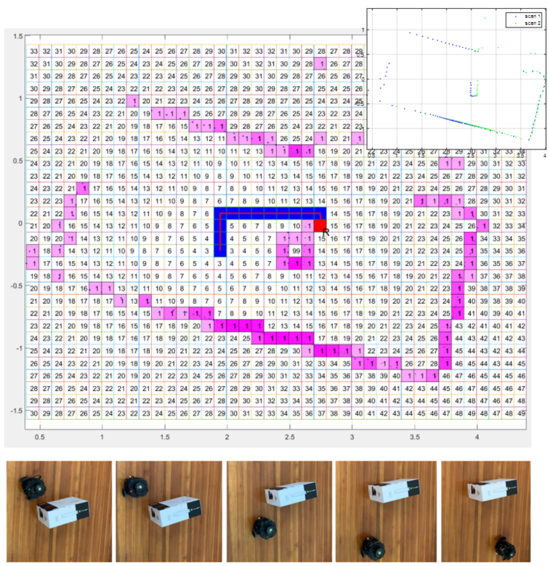


Figure 6: Path planning using Wavefront algorithm.

Areas where objects are detected will be considered as “occupied" or “inaccessible." After getting familiar with its surroundings, the robot is expected to find its way to a specified goal location without collision with obstacles. Solutions to this path-planning problem provide the robot with an obstacle-free path.

The Wavefront algorithm is the most basic but powerful approach to tackling path planning. The workspace is modeled as a 2D grid map. Locations occupied by the detected objects are marked inaccessible by denoting their values as "1". Grids representing open spaces will be assigned a non-zero value, following the algorithm's policy. Eventually, after all the open spaces have gotten their values updated/assigned, the robot will find its way from its current location (denoted by the red grid in Fig. 6) to the specified goal location by counting down the value one less than before. Several snapshots of the robot's movement as it follows the path are also shown.

Through this simple path planning project, the MATLAB-ROS-Turtlebot3 platform demonstrates its capability in fast algorithm development and implementation for processing large amounts of data.

5 LiDAR Data Processing

Transformation of laser data to the inertial frame: In many cases, data needs to be integrated. For example, in the path planning project described in Sec. 4, the robot collects laser data in several locations to obtain a more comprehensive idea about its environment. Laser data is collected in the robot's body-fixed frame. To transform these data to the same frame, i.e., the inertial frame, the robot's position and orientation at the time of sensing need to be used. This is a good example for students to understand the importance of homogeneous transformation, a common topic in robotics. Figure 7 shows an illustration of integrating three individual laser scans to form a better representation of the robot's environment.

The path planning algorithm described in Sec. 4 used raw laser data without going through additional processing. This project aims to familiarize students with advanced perception processes, focusing on object and shape recognition using laser scan data, specifically rectangle fitting. Based on data preprocessing (to obtain the point cloud to be processed), in the next, we will describe our simple segmentation (also called cluster detection) and rectangle-fitting schemes.

Distance-based segmentation: To conduct segmentation of 2D point clouds, the distance between consecutive points is typically used to determine if they are part of the same object. Using distance as the criterion, laser points are grouped as one cluster if they are close to each other (i.e., the distance in between is less than a pre-specified threshold). For each laser point, we first compute the minimal distance between this point with all existing clusters. If the minimum of these distances is greater than the specified threshold, this point is considered to belong to a new group. Otherwise, it is assigned to the one closest to it.

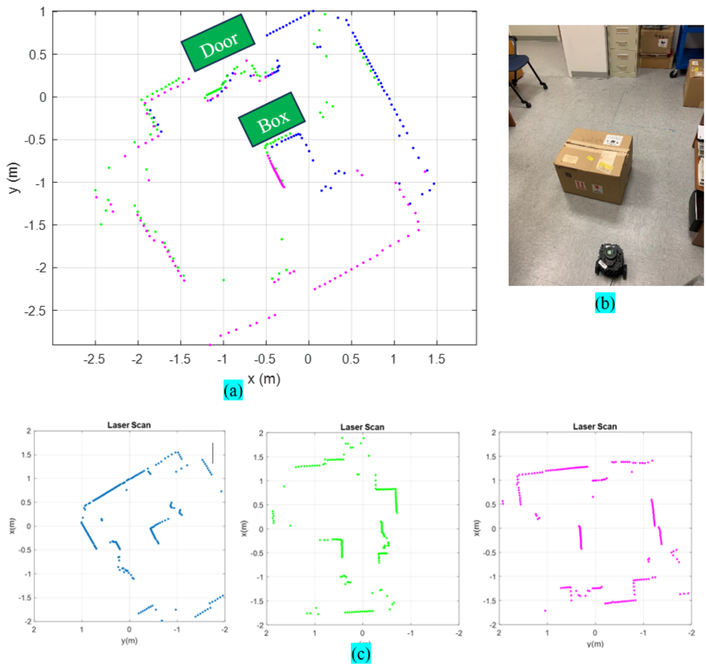
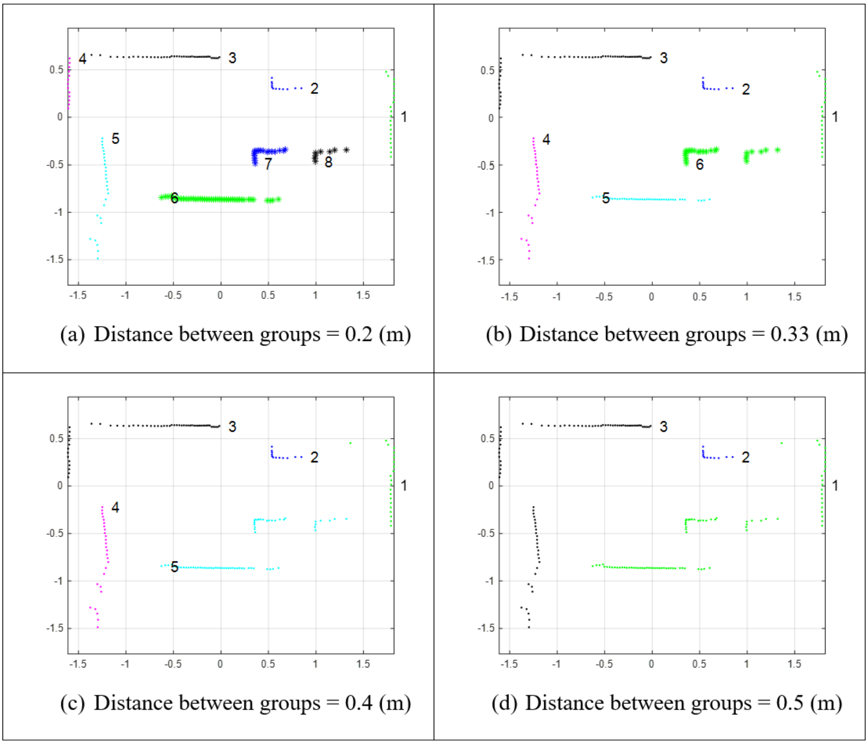


Figure 7: Transformation of laser data to the inertial frame, a good example of applying homogeneous transformation.

Figure 8 highlights the effect of selecting different parameters/values. Data points that were thought to belong to the same object are plotted in the same color. Different clusters are plotted in different colors. An index is written next to each cluster. It can be seen that setting the thresholds higher will result in fewer clusters.

Rectangle fitting to L-shape data: After data segmentation and clustering, we performed rectangle fitting to each cluster. We computed the sum of distances (i.e., errors) from each point in this cluster to the fitted model. If this sum of errors is "small" enough and the width & length of the fitted model are "large" enough, we will consider this cluster of points to represent a rectangle. The motivation for considering this L-shape rectangle fitting problem is to derive vehicle pose estimation in the Advanced Driving Assistance Systems (ADAS) scenario [13] in the future.

Figure 9 gives an illustration of the rectangle-fitting process. The raw laser data is shown in (a). Data segmentation is given in (b), which outputs 6 identified clusters. Rectangle fitting via MATLAB's LiDAR point cloud



(a) Distance between groups = 0.2 (m)

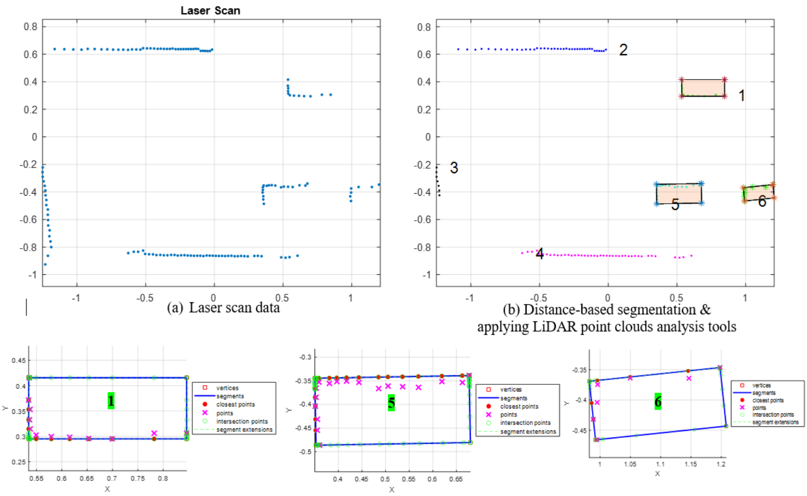
(b) Distance between groups = 0.33 (m)

(c) Distance between groups = 0.4 (m)

(d) Distance between groups = 0.5 (m)

Figure 8: Clustering algorithms applied to the obstacle points using distance-based segmentation. The minimal distance between groups is set to be 0.2, 0.33, 0.4, and 0.5 in (a), (b), (c), and (d), respectively.

analysis tools (particularly the pcfitcuboid() function) is used to fit a rectangle to each cluster [4]. The sum of the minimal distance from each point in the cluster to the fitted model is computed using the files shared on MATLAB's File Exchange Center, i.e., the "Distance from points to polyline or polygon" routines [14]. The final fitted rectangles are displayed in (b). Calculation of the distance error is shown in Fig. 9(c), that is, to compute the distance between each point (denoted by a pink cross) to its closest point (denoted by a red dot) on the rectangle.



(a) Laser scan data

(b) Distance-based segmentation & applying LiDAR point clouds analysis tools

(c) Closest points on the rectangle to the point cloud

Figure 9: Rectangle fitting routine by applying distance-based segmentation and MATLAB LiDAR point cloud analysis tools.

6 Discussion, Conclusion, and Future Work

Adopting proper simulation and/or experimental platforms in a robotics course is essential for students to apply classroom knowledge to practical situations. Due to the widespread use of ROS in research and industry, as well as the emerging integration of Artificial Intelligence (AI) with robotics and the need to align with a new Software Engineering Technology (SET) curriculum, we are currently investigating different options to update the existing physical robotic systems used in an undergraduate robotics course (CET 4952: Robotics Technology). The Turtlebot3 robots are excellent candidates for various features:
Being low-cost, small-size, sturdy, and portable. A low price will allow the department to purchase enough number of sets. Currently, we are thinking of providing one robot to each group consisting of two students. Being small-sized will allow the robots to be stored in cabinets inside the laboratory for students to check out and then check in during the lab sessions. Being sturdy will allow these robots to be used semester after semester since the robotics course is offered in both the spring and fall semesters. Being portable will allow these robots not to interfere with other courses that use the same lab.
Being able to provide students with hands-on experience in two fundamental areas (Autonomous Mobile Robots and Robotic Manipu-

lator): For undergraduate robotic education, it is a good practice to introduce these two areas either in one course or a sequence of two courses. So, we are looking for robotic systems that can cover both. The Burger and Waffle PI robots will allow students to fully explore almost all aspects essential to autonomous mobile robots, including sensing, perception, laser data processing, image processing, computer vision, map building, navigation, path planning, coordinated control, and integration of AI. The 5-DOF robotic manipulator, as shown in Fig. 10, will help students reinforce their understanding of homogeneous transformation, forward and inverse kinematics, and trajectory generation. Placing the robotic arm on top of the Waffle PI robot results in a robotic system that combines mobility with action (Fig. 10). The results presented in this paper focused on autonomous mobile robots. Future investigations will be conducted for the 5-DOF robotic arm and the integrated robotic system.

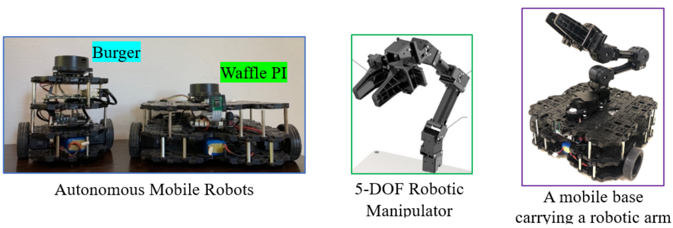


Figure 10: Autonomous mobile robots and robotic manipulators. The second and third pictures are from ROBOTICS website [10].

Being open-source and having well-established mechanisms for sharing and support within the community/society: Robotics is a rapidly evolving field that intertwines with many other areas such as electronics, communication, cybersecurity, computer science, signal & image processing, and mathematics. Advancements in these closely related areas will in turn have huge impacts on robotics. Algorithms and methods that enhance the autonomy of robots are developed much faster than traditional sources such as textbooks and conference proceedings. Researchers, educators, and students begin to use "new" ways to obtain timely support and keep up with the most recent developments in this field. Open source, which has become a trend since the last decade, has transformed into a global tendency, especially in fields like robotics. The demand for open source makes ROS-compatible robots more intriguing since ROS is a set of open-source software libraries and tools. The ROS forum allows users to ask questions, comment on others' discussions, and thus provide/receive support from the community. Similarly, MathWorks' File Exchange Center allows one to share/post their developments. The MATLAB-

ROS2-Turtlebot3 experimental platform utilizes benefits from both ends, including algorithm development, community support, and ready adoption of the developed algorithms to other robots.

Being able to serve as an undergraduate research platform and integrate the AI computational system with a robot: Due to the rapidly evolving nature of the robotics field, research needs to be seamlessly integrated with teaching to expose students to the latest developments. Exponential increases in computing power, sensor actuators, and communication transceivers have made producing robotic systems economically feasible. Students can now get access to fully-functioning robots at a much cheaper cost. Research opportunities thus become more available to undergraduates. We think robotics-related activities (courses and projects) should provide a propelling force in promoting undergraduate research. Further, in response to the emerging trend of using AI to find better solutions, integrating AI with robotics should play a leading role since this integration is inherent and embedded [1]. For example, existing feature extraction and face recognition algorithms in computer vision already have AI flavors. The distance-based segmentation and rectangle-fitting routines as described in Sec. 5 could be improved by AI-facilitated adaption and self-learning in determining the specified thresholds (i.e., the minimal distance among groups and the lower/upper bounds of the rectangles' dimensions).

The two exemplary projects presented in this paper confirmed the usage of the MATLAB-ROS2-Turtlebot3 robotic platform in a robotic course, by providing engaging and leaning environment through realistic scenarios. They will also help others to develop and teach similar robotic courses, which is inline with the nation's trend in Artificial Intelligence & Machine Learning curriculum. The physical platform can help boost undergraduate research by allowing students to explore AI-facilitated solutions to improve the robot's functionalities including mapping, navigation, obstacle avoidance, and coordination. In future investigations, we will explore: a) vision-based control via the onboard Raspberry PI camera; b) control of the 5-DOF (degree-of-freedom) robotic arm for pick-and-place tasks and then the integrated robotic system for warehouse applications; and c) formation control of multiple Robots.

7 Acknowledgement

The National Science Foundation, Award 2240516, supported this research.

References

[1] Robert Avanzato. "Deep Learning Projects for Multidisciplinary Engineering Design Students". In: *ASEE Annual Conference and Exposition*. 2023.

[2] Robert Avanzato and Culllen Wilcox. "Introductory Mobile Robotics and Computer Vision Laboratories Using ROS and MATLAB". In: *ASEE Annual Conference and Exposition*. 2018.

[3] Siavash Farzan. "Project-Based Learning for Robot Control Theory: A Robot Operating System (ROS)-Based Approach". In: *ASEE Annual Conference and Exposition*. 2023.

[4] Felipe24 Jiménez and Miguel Clavijo. "LiDAR point clouds analysis computer tools for teaching autonomous vehicles perception algorithms". In: *Computer Applications in Computer Education* (Feb. 2024), pp. 1–17.

[5] Lili Ma. "Teaching Undergraduate Robotic Courses using Enhanced VEX Robots". In: *Journal of STEM Education: Innovations and Research* (July 2021).

[6] Lili Ma et al. "Development of a Raspberry PI-Controlled VEX Robot for a Robotics Technology Course". In: *ASEE Annual Conference and Exposition*. June 2023.

[7] Lili Ma et al. "Introducing ROS-Projects to Undergraduate Robotic Curriculum". In: *ASEE Annual Conference and Exposition*. June 2023.

[8] Lili Ma et al. "Online Robotics Technology Course Design by Balancing Workload and Affect". In: *Transactions of the SDPS: Journal of Integrated Design and Process Science* (Jan. 2022).

[9] O. Michalec, C. O'Donovan, and M. Sobhani. "What is robotics made of? The interdisciplinary politics of robotics research". In: *Humanities and Social Sciences Communications* 65.8 (Mar. 2021), pp. 1–15.

[10] Robotics. *Turtlebot3 e-Manual*. `https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/`.

[11] ROS. *Robot Operating System*. `https://ros.org/`.

[12] S. Macenski et al. "Impact of ROS 2 Node Composition in Robotic Systems". In: *IEEE Robotics and Autonomous Letters* (2023).

[13] Xiaotong Shen, Scott Pendleton, and Marcelo H. Ang Jr. "Efficient L-shape Fitting of Laser Scanner Data for Vehicle Pose Estimation". In: *IEEE Conference on Robotics, Automation and Mechatronics*. July 2015.

[14] Michael Yoshpe. *Distance from points to polyline or polygon*. `https://www.mathworks.com/matlabcentral/fileexchange/12744-distance-from-points-to-polyline-or-polygon`.

Comparing K-8 Computing Education Implementations Between South Africa and Sweden*

Michael J. Lee[1], Annie Lang[2], Bruce Ferwerda[2]
[1]New Jersey Institute of Technology, Newark, NJ 07103
[2]Jönköping University, Småland, Sweden, 551 11

mjlee@njit.edu, laan18no@student.ju.se, bruce.ferwerda@ju.se

Abstract

Recently, the governments of South Africa and Sweden have mandated that the youth in their respective countries receive compulsory computing education. In this study, we explored how in-service teachers in these countries learned about and taught coding to their students. Moreover, we asked students from these respective countries to report on their prior coding experience, and had them take two coding-related tests. Speaking with six teachers (3 from each country) and surveying 96 students (47 from South Africa and 49 from Sweden), we discovered several interesting results. Teachers from both countries used similar online resources to learn coding themselves. However, teachers from South Africa preferred to provide their students with direct, concrete experiences with coding using existing physical and online resources. Their counterparts in Sweden preferred to teach their students abstract concepts, integrated into other topic areas. However, regardless of the type of instruction, students from both countries did equally well on validated tests measuring their understanding of coding concepts. These results highlight each countries' teachers' dedication and effectiveness in adapting materials for their students, and the students' receptiveness and measurable learning outcomes attributed to these methods.

1 Introduction

In today's increasingly digital world, equipping students with the ability to think computationally is crucial [17, 18, 20]. Computational thinking (CT) refers to a problem-solving methodology that draws on concepts fundamental to computer science [29]. It emphasizes the decomposition of complex problems into smaller, more manageable steps, the development of algorithms (step-by-step instructions) to solve those problems, and the iterative process of testing and refining solutions [30]. This skillset goes beyond computing, and can be applied to other subjects and aspects of life [11, 28].

Integrating CT into K-8 education offers a multitude of benefits. Research suggests that CT can strengthen core academic skills like critical thinking, problem-solving, and collaboration [30]. By breaking down problems into logical steps and identifying patterns, students develop analytical thinking abilities that are applicable across disciplines. Furthermore, CT fosters creativity and innovation as students design solutions and test their effectiveness, encouraging them to experiment and adapt their approaches [29].

The importance of CT in K-8 education is underscored by recent initiatives such as South Africa and Sweden requiring all of their primary and secondary school students to receive computing education. However, it has only been a few years since these requirements were mandated, so little is known about how instructors are teaching CT topics, and how well students understand these lessons. To address this, we explored how in-service teachers in South Africa and Sweden learned about and taught CT concepts to their students. Moreover, we asked students from these respective countries to report on their prior coding experience, and had them take two coding-related tests.

2 Background

2.1 Computational Thinking Education in Sweden

Sweden is a highly developed country ranked fifth in the Human Development Index. Prior to 2018, computing was not a separate subject in Sweden's K-12 curriculum. However, a major reform introduced programming concepts into the national curriculum for compulsory school (grades 1-9), though it is important to note that this did not result in standalone computing courses. Rather, computing and coding concepts are integrated into subjects like mathematics, technology, and even civics. Like many other countries, in-service teachers—who often do not have formal or prior experience with coding—are often tasked with teaching computing [10]. The in-service teachers often learn the content on their own, and find ways of combining their subject expertise with CT concepts, typically employing pseudocode or CS unplugged activities [1, 14].

2.2 Computational Thinking Education in South Africa

South Africa is considered a developing country, ranking 109th on the Human Development Index. In 2021, South Africa's Department of Education started their pilot and phased implementation of their "Coding and Robotics" curriculum [5]. This curriculum focuses on computational thinking concepts, and the engineering design process [5, 12]. Similar to their Swedish counterparts, many of South Africa's in-service teachers do not have formal or prior experience with coding [21, 24]. Since the "Coding and Robotics" course implementation, many instructors use or adapt existing online materials to learn and teach coding on their own, and find ways of combining their subject expertise with coding [24], most often using physical robotics [19] and/or Scratch, and then moving onto text-based programming as their students progress [7, 12].

3 Method

3.1 Participant Recruitment

We collected data from middle middle schools in South Africa and Sweden (two schools from each country). We contacted various schools in both countries, and received positive responses from two headmasters each from South Africa and Sweden. Each headmaster connected us with teachers from their respective schools (all co-ed) who were willing to participate in the study.

The school year for South African students is split into four terms, divided evenly into four quarters (3 months each, starting in January). The school year for Swedish students is split into two terms, Fall (August-December) and Spring (January-July). Due to the timing of responses from the headmasters, and a national holiday in South Africa, we collected data in September (Term 3 in South Africa, and Fall Term in Sweden) and December (Term 4 in South Africa, and Fall Term in Sweden) of the same year. This was to ensure that all of the participants from both countries were in the same annual term.

We surveyed 13-14 year old students (Grade 8 in South Africa; Grade 7 in Sweden), as this coincides with the first class of students in Sweden to start receiving their compulsory computing education (in 2018) and the first in South Africa to receive the pilot Coding and Robotics curriculum (in 2021). We confirmed that all student participants received their respective country's computing education from initiation to the time of the study. All experimental procedures and materials were reviewed and approved by an Institutional Review Board (IRB). We collected signed consent forms from teachers (and parents), and signed assent forms from all the student participants.

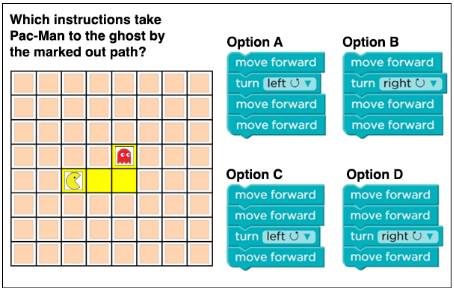


Figure 1: An example CTt question. Figure 2: An example Bebras question.

3.2 Instruments

We used **semi-structured interviews** to speak with the teachers. The interviews were 20 minutes each, all conducted by one researcher, and were guided by the following four questions: 1) *What is your area of teaching expertise and/or training?*; 2) *Did you have any prior experience teaching computational thinking or computer coding before (implementation of {respective country}'s requirements)?*; 3) *How do you teach computational thinking or coding?*; and 4) *What tools/resources do you use to teaching computational thinking or coding?*

Next, to measure students' computational thinking and coding knowledge, we used two validated measures to survey them using Google Forms: the Computational Thinking Test (CTt), and the annual Bebras tasks. It is important to note that both of these tests are "multi-lingual" [8], meaning that they were designed to cover important computational thinking/coding concepts across many languages that students may have learned or encountered (in these cases, using pseudocode-based assessments). We also asked students if they had any prior experience with block-based or text-based languages, each with a list of languages they could choose from (i.e., any that apply).

The **Computational Thinking Test (CTt)** [6, 32]—is a validated, multiple choice test specifically designed for our participant age range, consisting of 28 questions which measures different programming concepts such as basic instructions (Questions 1-4), loops (Questions 5-12), conditionals (Questions 13-24), and functions (Questions 25-28). The official recommended time is for students to complete it within 45 minutes.

The **Bebras Tasks** [4, 15]—are a part of the larger Bebras International Challenge, an increasingly popular initiative in promoting computational thinking skills [3, 4]. With participation from over 30 countries, the challenge provides a standardized tool for educators to gauge students' CT development on a global scale. More specifically, Bebras tasks are an assessment tool that utilizes engaging challenges to evaluate CT competency in students of diverse age groups. Unlike traditional assessments that might emphasize coding proficiency, the Bebras tasks prioritize the underlying thought processes and problem-solving strategies employed by students. This aligns with the framework proposed by Patten et al. [22], who advocate for evaluating CT by considering both coding skills and problem-solving approaches. Bebras tasks are updated annually, offered in various difficulty levels catering to different age groups with 18-24 questions, and similarly to the CTt, the challenges are to be completed within 45 minutes [2].

3.3 Procedure

Once we received confirmation of participation from each school's respective teacher(s), we e-mailed them to set up an initial one-on-one video call (Zoom) for a short semi-structured interview, along with a consent form. The first 5 minutes were used for introductions, 20 minutes for the interview, and 5 minutes to explain our study and give directions about the student surveys. All interviews were conducted in English and automatically transcribed in Zoom.

After the interviews, we sent all the teachers email instructions along with separate Google Form links for the CTt, the Bebras tasks, and consent/assent forms. To simplify keeping track of the tests and schools, we duplicated the Google Forms with a unique identifier for each class. We asked teachers to share each respective link with students on separate weeks during the time of the day/week they normally have computing related activities. On Week 1, the teachers administered the CTt, giving their students 45 minutes to complete it. Similarly, on Week 2, the teachers administered the Bebras tasks, giving their students 45 minutes to complete it. All schools had tablets and/or laptops with WiFi connections that students used to connect to the Google Forms.

For our CTt questions, we used those available from *csedresearch.org*, which was a PDF document containing 28 questions from Román-González's 2017 publication [23]. We copied the images and text from the PDF, in the exact same order, to create our Google Form questions.

For our Bebras tasks, we used the United Kingdom's (*bebras.uk*) set from 2022, for "Inters" (12-14 years olds), which included 18 questions. These questions were ordered in three ascending difficulty brackets A, B, and C (where C was the most difficult), with 6 questions each. We reused the images available on the UK Bebras website to create our Google Form questions, in the exact same order. For the tasks requiring clicking or dragging to place objects on the original UK Bebras, we overlaid appropriate location labels or coordinates with corresponding answer choices for our Google Form version.

4 Results & Interpretations

We received responses from two Swedish schools (3 classes; 3 different teachers), all based in Stockholm. We also received responses from two South African schools (3 classes; 3 different teachers), all based in Johannesburg. We interviewed all six of these teachers, who all identified as female, and had a median of 5 years of teaching experience for Sweden (range 3-10), and 7 years for South Africa (range 3-9). All reported that they were comfortable speaking English, and there were no communication issues during the interviews or follow-up email correspondence.

Next, we received a grand total of 96 student responses, with 47 from South Africa and 49 from Sweden. We combined all the responses within each country, as analyses of the scores within groups did not reveal any observable or statistically significant differences. All participants were 13-14 years old, and in either Grade 7 (Sweden) or Grade 8 (South Africa). Unfortunately, we did not collect any demographic information about the children outside of grade level and age (which were provided by the teachers).

4.1 Teachers' Experiences

We reviewed the auto-generated transcripts from the teachers' interviews, corrected errors (which consisted mainly of fixing words and punctuation), and organized by themes (and occurrences). Independent of country, we found commonalities in all of the teachers' background and training, and the way they looked for educational tools for themselves to learn the material. We also found major differences in how teachers presented and taught the information to their students, by country.

4.1.1 Not Experts in Computational Thinking or Coding

While all six of our teachers taught multiple subjects at their respective schools, they reported that their expertise/training in fields outside of computer science (or any closely related field). The teachers had a range of college degrees from their respective countries, with our Swedish teachers having degrees in English literature, human development, and biology; and our South African teachers having degrees in child/human development, psychology, and mathematics.

Only one out of the six teachers reported any significant experience coding or learning about coding prior to their government requirements. The teacher with prior experience, from South Africa, reported that she learned "a little bit of coding" (Python and Matlab) in college as a mathematics major, but was not initially comfortable teaching coding or computational thinking at school. All of the other teachers reported that they had seen or tinkered with programming online, using tools such as code.org or Scratch, but did not consider themselves at all experienced with coding or CT before having to learn it to teach it.

Much prior work in other countries have also found that existing in-service teachers are often assigned to teach technology/computing/coding courses, even if it is not necessarily within their expertise or training [9, 16, 31]. With the governments of South Africa and Sweden recently mandating that their K-8 students learn programming, it is unsurprising that all of the teachers that we interviewed were later assigned to teach their respective school's coding courses. If they are not already doing so, teachers from these countries might benefit from professional development courses in coding to increase their confidence and knowledge about coding.

4.1.2 Major Differences in Teaching by Country

We also found that the teachers from both countries used similar resources in learning about programming, but that the different countries used very different teaching styles and tools when teaching.

First, all teachers reported that they used code.org as a major resource for themselves to *learn* new materials, and also to find links to other coding resources. Five of the teachers (3 from South Africa, 2 from Sweden) also reported using Codecademy very often, especially to learn Python, Java, and HTML/CSS. All of them commented that they liked code.org for its separate resources for teachers, specific content grouped by age, and examples that they could adapt to their own teaching. The South African teachers also mentioned using Kid Spark Education (especially due to content about robotics), CodePen, and Scratch. The Swedish teachers mentioned CS unplugged, CoderDojo, and Scratch. In addition, there was much more emphasis on using printed text (i.e., textbooks) and examples from the Swedish Ministry of Education and (teacher) discussion boards/forums online.

The South African teachers talked about computational thinking as understanding the problem, and subsequently solving the problem by creating code. They tended to use coding resources directly, having students make accounts for Scratch and code.org to create projects and follow established curricula. They reported that they would use class time to follow the online guides, providing help and support to the students when needed [27]. In addition, the South African teachers would have the students make websites and practice coding in Javascript using CodePen. In this way, the South African students had direct/concrete experiences with coding and coding activities.

In contrast, the Swedish teachers talked about computational thinking as the understanding of logic, breaking down problems, and algorithms. All of them reported that they tended not to use online resources at all (or very sparingly), instead, using pseudocode or real-life examples to explain different coding concepts. Also, in addition to having some time set aside for general technology lessons ("teknikprogrammet"), most of the instruction about computing/coding were integrated and blended into other school subjects [10].

Two of the teachers mentioned using CS unplugged activities often, or assigning collaborative problem-solving tasks that would involve algorithms (both supporting findings by Bjursten et al. [1]). When we asked during the interviews why this was the case, the teachers responded that this was the way other teachers in their school(s) did it, and one specifically mentioned that she disliked giving the students too much screen time (following the Swedish shift to reduce screen time in classrooms in favor of textbooks [2, 13]). In this way, the Swedish students tended to receive indirect/abstract experiences with coding and coding activities.

These differences in teaching styles and preferences between countries were very interesting, especially given that the teachers from both countries used similar resources to learn about coding themselves. Based on our interviews with the teachers, we got the sense that the South African teachers generally valued their students getting practical, hands-on experience, coding with concrete, direct interaction with text/blocks. In contrast, the Swedish teachers generally prioritized their students understanding abstract concepts and algorithms in the context of other subjects and everyday life.

4.2 Students' Experiences

We reviewed all of the Google Form responses from the students. When applicable, we use nonparametric Wilcoxon Rank Sums tests with $\alpha = 0.05$ confidence throughout our analyses—as our data was not normally distributed—to compare our groups' (South Africa and Sweden) responses/scores. We report our statistically significant results with the understanding that our sample size is relatively small and that the findings may not be widely generalizable.

4.2.1 Major Differences in Prior Programming Experience

We compared the countries' students' response to the questions asking if they have prior coding experience in any block-based languages (e.g., Scratch, Snap!, Blockly) and if they have prior coding experience in any text-based languages (e.g., Javascript, Java, Python, C++). We found that nearly all of the South African students reported having experience in both/either block-programming or text-programming. In contrast, most of the Swedish students reported that they did not have experience in either block-programming or text-programming. This is in contrast to recent findings by Zhang et al., who found that 135 in-service Swedish teachers from 5 cities taught coding to Grades 4–6 using interactive robots (Bluebot and Beebot) and block-based programming languages, then from Grades 7-9, started transitioning to using more text-based programming languages [33].

Examining this more closely, we found a significant difference in the responses to both questions between countries. First, there was a significant difference in the number of students that reported prior experience with block-programming between the two groups ($W = 2668.5.5, Z = 3.4982, p < .05$), with the South African students reporting more experience. Second, there was a significant difference in the number of students that reported prior experience with text-programming between the two groups ($W = 2585.5, Z = 2.6649, p < .05$), with the South African students again reporting more experience.

While the results of Swedish students largely reporting that they did not have prior experience with either block or text coding was surprising in the context of others' prior work [33], it supports the teaching methods described by our Swedish teachers' interview results (as described in the previous section).

4.2.2 CTt & Bebras: No Differences Detected Between Countries

We compared the respective country's students' performance on the CTt questions, overall, and split by the test's topic areas. We found no detectable difference of students' scores between the countries overall ($W = 2052, Z = -1.6690, n.s.$), or when split by concepts: basic instructions ($W = 2152, Z = -0.9705, n.s.$), loops ($W = 2171.5, Z = -0.7959, n.s.$), conditionals ($W = 2103.5, Z = -1.2921, n.s.$), or functions ($W = 2105.5, Z = -1.3178, n.s.$).

Next, we compared the respective country's students' performance on the Bebras task questions, overall, and split by task difficulty. Similar to our CTt results, we found no detectable difference between groups on Bebras task performance between groups overall ($W = 2362, Z = 0.6037, n.s.$). We also did not find any detectable difference between countries by task difficulty: Difficulty A (easy) questions ($W = 2363.5.5, Z = 0.6218, n.s.$), Difficulty B (medium) questions ($W = 2238.5, Z = -0.3018, n.s.$), or Difficulty C (difficult) questions ($W = 2394.5, Z = 0.8513, n.s.$).

These results indicate that the students from South Africa and Sweden performed similarly (i.e., did equally well) on both the CTt and Bebras tasks. This suggests that both the South African style of providing their students with direct, concrete experiences with coding (and robotics) was equally as effective as their Swedish counterparts, who provided their students with more abstract experiences with coding that were integrated into other subjects. While we cannot say for certain that one country's style would work just as well for the other (and something that we could explore in a future study), we can say that each country's method works well in their specific contexts.

Since we only surveyed two schools from each country, our results may not be widely generalizable. Therefore, it would be worthwhile to explore a wider range of schools in both countries using the same methodology to see if we get similar results. Next, while it is likely clear to the South African students when they are learning about coding, it may also be interesting to explore whether Swedish students know specifically when they are being taught coding. Similarly, future work could explore whether these country's students think of themselves as being proficient coders (especially Swedish students, since they mostly indicated they did not have prior coding experience), if they are surprised by their high scores on the CTt and Bebras task, and what they think about their respective country's teaching method (e.g., how South African students view being taught by their teachers and online interactive computer tutors [25, 26]; or how Swedish students might respond to more direct/concrete CT and coding instruction).

5 Conclusion

Recently, the governments of South Africa and Sweden have mandated that the youth in their respective countries receive compulsory computing education. In this study, we explored how in-service teachers in South Africa and Sweden learned about and taught coding to their students. Moreover, we asked students from these respective countries to report on their prior coding experience, and had them take two coding-related tests.

Interviewing six in-service teachers (3 from each country), we found that all of them used similar resources for learning about computing and coding themselves (e.g. code.org, Scratch, and online forums). However, we found major differences between countries in the way they taught their students afterwards. South African teachers took a direct approach, having their students use existing online resources to get concrete experience with interacting with code. Swedish teachers took a contrasting approach, integrating computing lessons into other school topics, often abstracting CT concepts into psuedocode or physical activities (e.g., CS unplugged). A major limitation to this work is the small sample of teachers, and the relatively short time we interviewed them. They all volunteered to participate in the study, were from major cities from their respective countries, and identified as female. These shared characteristics may have led to similar experiences and points-of-view, which may not necessarily reflect the larger population. Therefore, in future studies, we may surface additional or different perspectives by recruiting from a wider area of each country, and include a larger and more diverse sample of teachers.

Surveying 96 students (47 from South Africa and 49 from Sweden) also yielded interesting results. Nearly all of the South African students reported that they had prior experience with numerous coding languages and styles (block- and text-based code), whereas their Swedish counterparts said they had no (or very little) experience with block- or text-based languages. However, we found that both countries' students did equally well on both the Computational Thinking test and Bebras tests, which are both designed to measure students' knowledge of CT concepts. This suggests that the teachers are learning coding material well, and also able to adapt what they learned into meaningful and effective instruction for their students. Future studies would benefit from surveying a wider and larger group of students, perhaps from other parts of each respective country. Collecting demographic information would also allow us to see if there are any detectable differences by factors such as gender and socio-economic status.

While our sample of teachers and students for this study was relatively small, our findings can inform computing education researchers and educators, as both concrete and abstract experience with code both appear to be effective ways for students to gain understanding of computational thinking concepts. All of the teachers and students in our study appeared to be doing very well with learning (and teaching) CT concepts, especially given the short time since it was introduced into the compulsory curricula for each country. Future work will examine the nuanced differences between countries further, expanding our dataset to include more teachers, students, and possibly additional countries.

6 Acknowledgements

This work was supported in part by the Google Research Scholar Program. Any opinions, findings, conclusions, or recommendations are those of the authors and do not necessarily reflect the views of Google Research or other parties.

References

[1] Eva-Lena Bjursten, Tor Nilsson, and Lena Gumaelius. Computer programming in primary schools: Swedish technology teachers' pedagogical strategies. *International Journal of Technology and Design Education*, 33(4):1345–1368, 2023.

[2] José Cano. We don't need no digital education? Tech-Savvy Sweden leads push to remove screens from schools. *World Crunch*, March 2024.

[3] Valentina Dagienė and Gerald Futschek. Bebras international contest on informatics and computer literacy: Criteria for good tasks. In *Informatics Education-Supporting Computational Thinking, ISSEP*, pages 19–30. Springer, 2008.

[4] Valentina Dagienė and Sue Sentance. It's computational thinking! Bebras tasks in the curriculum. In *Informatics in Schools: Improvement of Informatics Knowledge and Perception, ISSEP*, pages 28–39. Springer, 2016.

[5] Kareen Fares, Brian Fowler, and Emiliana Vegas. How South Africa implemented its computer science education program. *Brookings Institution*, 2021.

[6] M Román González. Computational Thinking Test: Design guidelines and content validation. In *EDULEARN*, pages 2436–2444. IATED, 2015.

[7] Jean Greyling. Coding unplugged—a guide to introducing coding and robotics to South African schools. *Transforming Entrepreneurship Education*, 155, 2023.

[8] Mark Guzdial. Holding ourselves to a higher standard: Language-independent just doesn't cut it. *Computing Ed Research – Guzdial's Take*, August 2019.

[9] Per Dahl Haagensen. Supporting in-service teachers of programming in secondary schools. Master's thesis, NTNU, 2021.

[10] Fredrik Heintz, Linda Mannila, Karin Nygårds, Peter Parnes, and Björn Regnell. Computing at school in Sweden–experiences from introducing computer science within existing subjects. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 118–130. Springer, 2015.

[11] Peter B Henderson, Thomas J Cortina, and Jeannette M Wing. Computational thinking. In *SIGCSE*, pages 195–196. ACM, 2007.

[12] Jana Heyns. *Development of a framework of factors essential to the optimal implementation of the Coding and Robotics subject in South African schools.* PhD thesis, Stellenbosch University, 2023.

[13] Anne-Françoise Hivert. Too fast, too soon? Sweden backs away from screens in schools. *Le Monde*, May 2023.

[14] Helena Isaksson Persson. Has computational thinking a place in the curriculum? In *Pupils' Attitudes Towards Technology*, pages 213–230, 2022.

[15] Cruz Izu, Claudio Mirolo, Amber Settle, Linda Mannila, and Gabriele Stupuriene. Exploring Bebras tasks content and performance: a multinational study. *Informatics in Education*, 16(1):39–59, 2017.

[16] Barbara Kushan. Preparing programming teachers. *ACM SIGCSE Bulletin*, 26(1):248–252, 1994.

[17] Michael J Lee. Exploring differences in minority students' attitudes towards computing after a one-day coding workshop. In *ACM ITiCSE*, pages 409–415, 2019.

[18] Michael J Lee. Providing near-peer mentorship to increase underrepresented minority youth participation in computing. In *Handbook of Research on Equity in Computer Science in P-16 Education*, pages 1–13. IGI Global, 2021.

[19] Veronica Lin and Orit Shaer. Beyond the lab: Using technology toys to engage South African youth in computational thinking. In *CHI Extended Abstracts*, pages 655–661. ACM, 2016.

[20] Jamie C Macbeth, Michael J Lee, Jung Soo Kim, and Tony B Zhang. Children who code. *The Routledge Companion to Digital Media and Children*, pages 67–74, 2020.

[21] Matsie Magdeline Mokonyane-Motha and Thelma Thelma De Jager. The significance of training student-teacher lecturers in pedagogical robotic and coding skills. *International Journal of Social Science Research and Review*, 6(12):107–118, 2023.

[22] J. Patten, P.J. Guo, and K.A. Toh. Fostering computational thinking in young children. In *SIGKDD*, pages 63–872. ACM, 2013.

[23] Marcos Román-González, Juan-Carlos Pérez-González, and Carmen Jiménez-Fernández. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, 72:678–691, 2017.

[24] Ismaila Temitayo Sanusi and Fitsum Deriba. What do we know about computing education in Africa? A systematic review of computing education research literature. *arXiv preprint arXiv:2406.11849*, 2024.

[25] Ruiqi Shen and Michael J Lee. Learners' perspectives on learning programming from interactive computer tutors in a MOOC. In *VL/HCC*, pages 1–5. IEEE, 2020.

[26] Ruiqi Shen, Donghee Wohn, and Michael J Lee. Programming learners' perceptions of interactive computer tutors and human teachers. *International Journal of Emerging Technologies in Learning (iJET)*, 15(9):123–142, 2020.

[27] Ruiqi Shen, Donghee Yvette Wohn, and Michael J Lee. Comparison of learning programming between interactive computer tutors and human teachers. In *CompEd*, pages 2–8. ACM, 2019.

[28] Bernhard Standl. Solving everyday challenges in a computational way of thinking. In *ISSEP*, pages 180–191. Springer, 2017.

[29] Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

[30] Aman Yadav, Hai Hong, and Chris Stephenson. Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, 60:565–568, 2016.

[31] Soner Yildirim. Effects of an educational computing course on preservice and in-service teachers: A discussion and analysis of attitudes and use. *Journal of Research on Computing in Education*, 32(4):479–495, 2000.

[32] María Zapata-Cáceres, Estefanía Martín-Barroso, and Marcos Román-González. Computational Thinking Test for beginners: Design and content validation. In *EDUCON*, pages 1905–1914. IEEE, 2020.

[33] LeChen Zhang, Jalal Nouri, and Lennart Rolandsson. Progression of computational thinking skills in Swedish compulsory schools with block-based programming. In *Australasian Computing Education Conference*, pages 66–75, 2020.

Jguardrail: A Framework for Identifying Possible Errors in Student Java Code*

Ian Finlayson and Stephen Davies
Computer Science Department
The University of Mary Washington
Fredericksburg, VA 22401
ifinlay@umw.edu, sdavies@umw.edu

Abstract

This paper introduces Jguardrail, a tool for identifying potential programming errors in Java programs, especially for beginning programming students. We identified several programming patterns which often lead to bugs in student programs, that are not flagged as warnings by the Java compiler. Jguardrail is a static analysis tool, written with the ANTLR parser framework, which recognizes these patterns and provides warning messages to the programmer. By providing an additional layer of warning reporting, above what the compiler itself provides, Jguardrail aims to help students avoid common programming pitfalls. This paper discusses the patterns Jguardrail provides warnings for, its usage in a CS2 course, and a comparison to other tools.

1 Introduction

This paper presents a tool for identifying common programming errors in Java programs. Over many years of teaching using the Java programming language, we identified several patterns that result in bugs that beginning programming students struggle to find and correct. This tool, named Jguardrail [6], makes a first pass over programs looking for these mistakes and reports them as warnings, effectively adding an extra layer of warning reporting on top of what the Java compiler offers.

Java is the predominant programming language used in computer science education at the college level [16]. Java has many features to recommend it, and is widely used in industry, but in our opinion the Java compiler does not do a great job of identifying warnings, even compared to compilers like GCC. There are many programming constructs that will certainly, or almost certainly, result in erroneous code which the Java compiler does not flag, even with extra warnings turned on. Jguardrail attempts to give programmers, especially beginners, more diagnostic warnings in such instances.

There has been much prior work relating to the most common mistakes made by novice programmers. Jackson et. al. [10] published a study in which they identify the most common compiler errors encountered by beginning Java students, using the compiler errors that the Java compiler itself gives.

Brown and Altadmri [2] investigated to what degree instructor perceptions of the common mistakes students make are accurate. Some of the mistakes in their study are syntax errors the Java compiler will always catch. Others are mistakes that the compiler does not catch, including many that Jguardrail provides warnings for.

There have been many published articles [1], [12], [13], [15] studying novice programmer's level of understanding of error messages, or works that seek to make such error messages more comprehensible to beginners. Much of this research suggests that these messages may be written for an audience of more experienced developers rather than novices, and that there are improvements to be made in gearing error and warning messages to novices.

Jguardrail differs from these works in that it seeks to actually augment the messages given by the compiler. We have identified mistakes that are not identified by the Java compiler and built a tool which does identify them.

This paper is structured as follows: Section 2 enumerates the patterns that Jguardrail detects as being possible mistakes and issues warnings for. Section 3 discusses the implementation of Jguardrail. Section 4 provides the results of using this tool with students in a CS 2 class. Section 5 compares its warnings to those identified by the Java compiler itself, as well as IDE and linting tools. Section 6 discusses future work and section 7 draws conclusions.

2 Patterns Flagged by Jguardrail

This section will describe the coding patterns flagged by Jguardrail as being either likely sources of bugs or poor programming style.

2.1 String Equality

In Java, there is almost never a reason to use the == operator to compare a string to a literal value. To that end, Jguardrail flags code like this:

```
if (choice == "Quit")
```

The programmer certainly meant to use the `.equals` method instead. This mistake can be difficult to debug since it will seem like string comparisons simply aren't working as the programmer believes they should.

2.2 Empty Control Structures

In this pattern, there is a control structure that has a single semi-colon for its body, as in the following code:

```
while (index < 100);
{
    array[index] = 0;
    index++;
}
```

Here the semi-colon on the end of the line with the while loop forms the body of the loop, meaning the loop is likely to be infinite, as in this case.

2.3 Missing Braces on Control Structures

Jguardrail also catches cases where the curly braces for a control structure are not written. Most consider omitting the braces to be poor style. For instance the Google Java Style Guide [4] dictates their use even if blocks contain a single statement. They can also lead to mistakes like this:

```
if (numbers[i] > numbers[i + 1])
    int temp = numbers[i];
    numbers[i] = numbers[i + 1];
    numbers[i + 1] = temp;
```

Here the code will not behave as the author probably intends, since only the first indented line is actually part of the if statement. Jguardrail gives warnings when braces are omitted to encourage good style and avoid such bugs.

2.4 Constant Integer Division

Jguardrail flags code where division is applied between two integers literals which do not divide evenly, as in the following code example:

```java
public static double triangleArea(int base, int height) {
    return (1 / 2) * base * height;
}
```

The issue here is that Java sees 1 and 2 as integers and so applies integer division, truncating the result to 0. This can be especially confusing to students coming from languages like Python that convert numbers to floating-point values in cases like this.

2.5 Missing this

If a programmer is attempting to set an instance variable to a parameter value, as in a constructor or setter method, and forget the `this.`, then we create a useless statement such as the following:

```java
public Student(String name) {
    name = name;
}
```

Jguardrail issues a warning, as it is likely that the intention was to set an instance variable with the same name instead.

2.6 Variable Shadowing

A similar mistake can be seen when a programmer accidentally shadows an instance variable, such as in the following code:

```java
public void setName(String newName) {
    String name = newName;
}
```

Here, instead of setting the existing instance variable `name` to `newName`, we have introduced a new local variable with the same name (which is called shadowing). Jguardrail flags the declaration of variables which shadow instance variables like this, as it can cause difficult to debug issues where variable updates seem not to take effect.

2.7 Void Constructors

We also issue warnings when a "constructor" has accidentally been marked as `void`, as in the following code snippet:

```java
public void Student() {
    this.name = "";
    this.grades = new ArrayList<>();
}
```

Since this has a return type, Java sees it not as a constructor, but rather as a method which just happens to be named the same thing as the class it is in. So the constructor will not run when objects are created, leading to a confusing debugging session. Jguardrail detects this for default constructors or those with parameters.

2.8 Uninitialized Variables

Related to the previous issue, many students make mistakes in their programs caused by not initializing instance variables. If class-type instance variables are not initialized, they begin as null references. Students often struggle with `NullPointerExceptions` caused by this issue. Jguardrail checks for instance variables that are not initialized (whether inline or in constructors) and issues warnings for them. This warning is perhaps a little opinionated since Java does guarantee initial values for all variables (unlike C++).

2.9 Mis-capitalized toString

In Java we can control how objects are displayed by overriding a method called `toString`, but this only works when the method is spelled and capitalized correctly. If the student calls the method `tostring`, for instance, Objects will be printed out as just the class name, an `@` character and the memory address the object is stored at. Jguardrail checks for methods like this and issues a warning that the programmer likely meant `toString` instead.

2.10 Missing Breaks in Switch

Finally Jguardrail looks for switch statements where there are missing break statements in the cases, as in the following code:

```
switch (option) {
    case "add":
        System.out.println("Adding");
    case "quit":
        done = true;
}
```

While there are legitimate uses of having switch cases "fall through", such as handling both upper-case and lower-case menu options, or "Duff's Device"[5], most of the time students do this by mistake. Interestingly, this is the only one of these which is caught by the `javac` compiler, and even then only with the `-Xlint:all` flag.

3 Implementation

Jguardrail essentially is a compiler front-end which parses Java code, performs some analyses and then (unlike an actual compiler) stops there. It only produces warnings and leaves actual compilation up to the existing Java compiler.

Jguardrail is implemented using the ANTLR [14] parser framework. ANTLR is a parser generator tool which also comes with example grammars for existing languages such as Java, which we used. This provides us with a syntax tree of the program being analyzed, after lexical and syntax analysis are performed. One of the great things about ANTLR is that the grammar is decoupled from actions taken after parsing. It uses the Visitor Pattern [9] to allow code to walk the parse tree after it is produced. We utilized this to write the analyses as visitors that walk over the tree after the code is parsed, searching for the patterns we identified.

Jguardrail does a separate tree-walk for each of the patterns that it catches, using a system in which new ones can be easily added to the system. Each of these overrides the needed `visit` methods ANTLR produces. For example, to catch division between two integer constants, we visit tree nodes which perform division. From there we simply check if both operands are integer constants and, if so, whether they divide evenly or not.

Some of the checks were slightly more complicated to perform. For instance to look for uninitialized instance variables, we have to make two passes. In the first we note all of the instance variables that a class has. If it's initialized inline, we skip over it, but if not, we add it to a list. In the second pass, we scan through all the constructors making sure that each one initializes all of the instance variables it needs to. Two passes are needed as constructors do not necessarily come after all instance variables. Moreover this analysis must be done using a stack as Java allows for nested classes. So we in fact keep a stack of lists of uninitialized variables to ensure that it works correctly in this case as well.

Each of these analyses records any warnings that should be issued about the code, which are displayed at the end of the program. We sort them by line number and pattern the output after that given by the `javac` compiler.

4 Evaluation

This tool was used in two sections of a CS 2 class during the Spring 2024 semester. The class uses the Java programming language and introduces concepts of object-oriented design and analysis. This class was taught using a command-line interface in which students write programs in the Vim text editor and compile using the `javac` command, on a shared server operated by the department.

With the goals of helping students avoid errors, and of learning how often students actually write the patterns Jguardrail checks for, we set up our system so that Jguardrail was automatically invoked when the `javac` command was run. This was done by creating a shell script in `/usr/local/bin` called `javac` which calls Jguardrail followed by the real `javac`, which is in the `/usr/bin` directory. Because `/usr/local/bin` comes first in a default `$PATH` variable, students got the Jguardrail warnings by default.

We also decided to write our script in such a way that if Jguardrail *did* find any warnings, it does not compile the code at all. This way, warnings must be fixed before the code can be run. This behavior is similar to the `-Werror` flag that the GCC compiler provides.

We instrumented Jguardrail so that each time a warning was issued, we saved a record of it so that we can collect usage statistics. This machine is used by students in many classes, but here we will look specifically at the 43 students in the two sections of the CS 2 class.

The student who triggered the fewest warnings triggered 19 of them. The student who triggered the most triggered 597. The mean was 124.3 and the median value was 93. This was across a semester's worth of work.

Table 1 contains data on how many times each of the warnings was triggered by the students in the CS 2 class. The warnings are listed in the same order as they are described in Section 2.

Table 1: Warnings issued by Jguardrail over one semester of a CS 2 class

Pattern	**Times Issued**
String Equality	125
Empty Structures	25
Missing Braces	218
Integer Division	0
Missing this	12
Local Shadowing	312
Void Constructor	65
Uninitialized Vars	4580
tostring	0
Missing Break	7

As can be seen, students are much more likely to trigger some of these warnings than others. The integer division and mis-capitalized `toString` method were never triggered at all. The integer division one was actually triggered by a more advanced student using the server, but not in the CS 2 class population.

The use of uninitialized instance variables is a clear outlier. It is possible that Jguardrail is a little over-zealous in its enforcement of the practice of initializing all instance variables. However it certainly leads to bugs in student programs and we believe being a little extra rigorous is not a bad thing at this level. Additionally Java already ensures that all local variables are initialized before being used, so enforcing this for all variables provides a consistent rule for students.

We believe that the use of Jguardrail in this class did make a difference in the amount of bugs in student code. Even setting aside the use of uninitialized instance variables, many of the other warnings were issued a significant number of times and can cause difficult bugs if not addressed. It also made helping students who encounter these issues easier on the instructor. For example in the case of shadowing a local variable, we can explain the warning to them and how to fix it, rather than need to help them debug why their variables are not being updated as they expect.

The tool did cause some issues for students, however. Code in the textbook used for the class occasionally violates the rules we enforce, which can be frustrating for a student who may not understand why code given to them by the text may fail to compile. Jguardrail is rather an "opinionated" tool in that it flags things as errors which may be considered subjective style decisions. It raises the question of what is considered "good code" which different instructors will certainly have different opinions on. However we do believe it served to help students avoid bugs in their programs.

5 Comparison to Other Tools

In this section we compare Jguardrail to other tools which report warnings on Java code. We compared the OpenJDK `javac` compiler itself, as well as the Eclipse[8], NetBeans[7] and IntelliJ[11] IDEs. Interestingly these IDEs each provide more warnings than the compiler itself provides. We also tried the Sonar Lint[17] linting tool plugin for IntelliJ, which provides more warnings on top of IntelliJ itself. We also tried the SpotBugs[18] plugin for IntelliJ, but it did not provide any more warnings than IntelliJ itself did so it is not present in the results.

Table 2 shows which of the checks that Jguardrail performs are caught by these tools. Again, these checks are listed in the same order as they are described in Section 2 of this paper.

Table 2: Comparison of tools which offer warnings for Java programs

Pattern	javac	Eclipse	NetBeans	IntelliJ	Sonar
String Equality			✓	✓	–
Empty Structures			✓	✓	–
Missing Braces					
Integer Division				✓	–
Missing this		✓	✓	✓	–
Local Shadowing		✓[1]	✓		✓
Void Constructor		✓		✓[2]	–
Uninitialized Vars				✓	–
tostring					✓
Missing Break	✓[3]				✓

[1] Eclipse gives a warning for the code we used to test local variables shadowing instance variables, but the warning was that the local variable was never used. This warning does not clearly indicate the underlying mistake.

[2] IntelliJ provides a warning for our test which says that the method (with the same name as the class) is never called. This indicates something is wrong, but is not especially clear as to the mistake being made.

[3] The javac compiler only provides warnings for missing break statements with the `-Xlint:all` flag passed to the compiler.

The – entries for Sonar are because that tool was used as a plugin for IntelliJ. So the cases where IntelliJ already provided warnings were still provided with the Sonar lint tool installed.

The IntelliJ IDE, especially with the Sonar lint tool installed, catches almost all of the cases Jguardrail catches. The Checkstyle tool [3] was also tried and performs similarly to Sonar. However, in our experience most students ignore the warnings given by IDEs such as IntelliJ for a few reasons. One, they are fairly inconspicuous and many students don't even really notice them. Two, these IDEs allow you to run the program if it compiles, whether or not warnings are given. Students will usually choose to run and test their code rather than read the warnings. Finally IDEs and style checkers use warnings to suggest refactoring programs in ways students may find confusing. For example, IntelliJ suggests that switch statements be replaced with switch expressions which students may not have seen.

Jguardrail is designed to be a code analysis framework specifically for beginning students. It does this by focussing on mistakes that we have observed beginning students making, making the warning messages clearer than some other tools provide, and making the warnings more visible. In our usage of the tool, we went so far as to dis-allow running of the program until warnings are resolved.

Jguardrail is also a command-line tool. In an environment, such as ours, where we encourage students to use the command-line to write and compile programs, Jguardrail provides many more warnings than the `javac` compiler itself.

6 Future Work

There are several areas in which Jguardrail can be expanded. Of course more warnings can be added to the tool. We would like to issue warnings for code which is improperly indented. That way if the code structure the programmer sees does not match the structure the compiler sees, the student will be given a warning. This will be a little trickier to implement using ANTLR than the other analyses, but is possible.

We will also be making the tool more flexible by providing a means for users to determine which of the patterns should be considered errors (disallowing compilation), warnings (allowing compilation), or disabled entirely. This will allow instructors to determine which patterns they want to be highlighted as potential mistakes for their students.

We are also beginning work on an IDE plugin for Jguardrail, initially for IntelliJ. The goal here is not only to provide warnings for the things Jguardrail provides warnings for, but also to allow the option of making fixing warnings mandatory before programs can be run.

7 Conclusion

Jguardrail is a framework for identifying programming patterns which will, or are very likely to, lead to incorrect Java programs. It currently provides warnings for ten such patterns, which are not caught by the Java compiler itself. This tool can be extended to catch other such patterns.

The study we conducted shows that most of these are actually encountered in real student code at the CS 2 level. Because Jguardrail catches these mistakes at compile time, students should be in a better position to fix the underlying mistake instead of needing to debug the symptoms of the mistake. We can also use this data to inform our teaching, highlighting the issues that we see occurring the most for our students.

Overall we believe that the needs of novice programmers are different from those of experienced professionals. We have by and large moved away from using programming languages geared towards beginners (such as Pascal or Scheme) in favor of those used in industry. However, that doesn't mean we need to use the same tools around those languages. We believe that it is possible to develop tooling that does a better job of catching and reporting mistakes that beginning students most commonly face, and Jguardrail is a step in that direction.

References

[1] Brett A. Becker et. al. "Effective compiler error message enhancement for novice programming students". In: *Computer Science Education* 26.2-3 (2016), pp. 148–175.

[2] Neil C.C. Brown and Amjad Altadmri. "Investigating novice programming mistakes: educator beliefs vs. student data". In: *Proceedings of the Tenth Annual Conference on International Computing Education Research*. ICER '14. 2014, pp. 43–50.

[3] CheckStyle. *CheckStyle*. `https://checkstyle.sourceforge.io/`.

[4] Google Corporation. *Google Java Style Guide*. `https://google.github.io/styleguide/javaguide.html`.

[5] Tom Duff. *Explanation, please!* `https://www.lysator.liu.se/c/duffs-device.html`.

[6] Ian Finlayson. *Jguardrail Code Repository*. `https://github.com/IanFinlayson/jguardrail`.

[7] Apache Software Foundation. *NetBeans Homepage*. `https://netbeans.apache.org/front/main/index.html`.

[8] Eclipse Foundation. *Eclipse Homepage*. `https://eclipseide.org/`.

[9] Erich Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1st ed. Addison-Wesley Professional, 1994.

[10] James Jackson, Michael Cobb, and Curtis Carver. "Identifying top Java errors for novice programmers". In: *Proceedings frontiers in education 35th annual conference*. IEEE. 2005, T4C–T4C.

[11] JetBrains. *IntelliJ Homepage*. `https://www.jetbrains.com/idea/`.

[12] Yoshitaka Kojima, Yoshitaka Arahori, and Katsuhiko Gondow. "Investigating the difficulty of commercial-level compiler warning messages for novice programmers". In: *International Conference on Computer Supported Education*. Vol. 2. SCITEPRESS. 2015, pp. 483–490.

[13] Marie-Hélène Nienaltowski, Michela Pedroni, and Bertrand Meyer. "Compiler error messages: what can help novices?" In: *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*. SIGCSE '08. 2008, pp. 168–172.

[14] Terence Parr. *The definitive ANTLR 4 reference*. The Pragmatic Bookshelf, 2013, pp. 1–326.

[15] James Prather et al. "On Novices' Interaction with Compiler Error Messages: A Human Factors Approach". In: *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ICER '17. 2017, pp. 74–82.

[16] Robert M. Siegfried et al. "Trends Of Commonly Used Programming Languages in CS1 And CS2 Learning". In: *2021 16th International Conference on Computer Science and Education (ICCSE)*. 2021, pp. 407–412. DOI: 10.1109/ICCSE51940.2021.9569444.

[17] Sonar. *Sonar Lint Homepage*. https://www.sonarsource.com/products/sonarlint/.

[18] SpotBugs. *SpotBugs Homepage*. https://spotbugs.github.io/.

Teaching Bioinformatics Students to Lead Reproducible Research*

Miranda Malouf Darby
Hood College
Frederick, MD 20854
darby@hood.edu

Abstract

Reproducibility, the extent to which the results of an experiment will lead to the same conclusion each time the experiment is performed, and robustness, the extent to which an experiment is generalizable to other situations, are essential indicators of the reliability of a study. Unfortunately, many studies that have sought to replicate published work reveal that most results cannot be reproduced. A great deal has been written about "reproducibility crisis" over the course of the last decade, yet the problem persists. Ongoing efforts to develop best practices for experimental design, statistical analysis, and the handling of data and other computational resources have not resolved the problem. Best practices are often not followed. It is not sufficient to train bioinformatics students to understand and implement these practices. Many bioinformatics studies involve scientists with a variety of specialties who all contribute data for the bioinformatician to analyze. Therefore, in order to solve the reproducibility crisis, bioinformaticians need communicate well with their colleagues who have different training backgrounds and convince them to follow best practices. This paper describes a course that teaches the best practices for research methodology and data handling, the ability to assess the quality of research results and data produced by others, and the leadership skills to influence others to perform research well.

1 Introduction

For a decade, researchers have bemoaned the "reproducibility crisis" revealed when efforts to repeat published studies did not lead to the published results [18], [3]. Reproducibility, the extent to which an experiment will lead to similar results each time it is performed, is an essential indicator of the reliability of an experiment. Reproducibility alone does not guarantee that the results are reliable, since problems in experimental design or analysis that are faithfully repeated could lead the same erroneous results. But, the ability to successfully replicate an experiment is a good first step. Experiments that are well designed and executed are also robust, meaning that the findings are generalizable to other situations [15]. Robustness relies on good experimental design regarding sample collection and processing, data acquisition, and appropriate analysis methodology in addition to good data management and analytical workflow development. Unfortunately, the majority of published research findings in a variety of life science fields cannot be replicated [3, 15, 18, 20, 27] and recent studies report that the same is true for machine learning-based science [1, 10].

The lowest standard of experimental replication is "computational replication", which does not repeat the entire experiment but only the analysis steps taken after the data is collected [6, 13]. In theory, this should be easy if all stages of the analysis are scripted. In 2016, a diverse group of stakeholders from academia, funding agencies, scholarly publishers, and industry worked together to develop principles to enhance the reusability of data and analyses [24]. The FAIR principles of Findability, Accessibility, Interoperability, and Reusability [7, 25] apply to all types of data and also the associated algorithms, tools, and workflows. However, subsequent studies indicate that the FAIR principles are not well adopted in practice. Workshops conducted in 2018 and 2019 at the National Institutes of Health attempted computational replication of five bioinformatics studies and failed because of missing data, software, and documentation [27]. Attempts at computational replication of Jupyter notebooks published with biomedical articles [20] and R scripts from the Harvard Dataverse repository [22] revealed similar problems of missing data, buggy code, and broken dependencies.

Bioinformatics is an interdisciplinary field that applies advanced statistical methodology and computational techniques to the analysis of biological data. That means that bioinformatics studies must not only be computationally reproducible, but also use appropriate statistical methodology and follow the best practices for biological experimental design. Unfortunately, efforts to address the use of statistical methods [**13**, 23] and the application of experimental design principles for robust research [8, 14] have not resolved the reproducibility crisis any more than the computational efforts.

The reproducibility crisis in bioinformatics is compounded by a lack of effective guidance. Increasing adoption of next-generation sequencing and other high-throughput techniques that generate massive datasets that are too large for standard analysis methods has led to a boom in bioinformatics. Rather than looking at one gene at a time, new techniques allow researchers to simultaneously query every gene in the genome, every RNA in the transcriptome, or all of the proteins or metabolites in a cell. However, the rapid evolution of new types of data generated by these techniques has outpaced the evolution of reliable techniques to analyze them. For many types of data, no "gold standard" analysis methods exist. The lack of established methods means that bioinformaticians must use their best judgement when deciding how to analyze an experiment, requiring them to fully understand the biological underpinnings of the data and associated metadata, appropriate use of advanced statistical techniques, and how to implement their analyses using code that is robust and error-free. Then, for the analysis to be reproducible, all data, metadata, and code need to be well annotated and able to be used by other researchers. Therefore, the reproducibility crisis will not be resolved until and unless experimental design considerations, statistical methodology, and computational guidelines become intrinsic parts of bioinformatics education.

However, as previous research demonstrates [1, 15, 20, 22, 26, 27], simply teaching skills for reproducible research will not resolve the crisis. Bioinformaticians must be leaders who ensure that the best practices for robust and reproducible research are correctly understood and implemented. Bioinformatics is inherently multidisciplinary, so bioinformatics studies involve teams of scientists with different training backgrounds who work together. Since the field of bioinformatics is relatively new and rapidly evolving, many bioinformaticians are young and early in their careers. They work in laboratories or research groups headed either by classically trained biologists or computer scientists, who understand their own field deeply but may not understand bioinformatics. For example, biologists who are used to smaller, more focused studies may not understand how many samples are needed to overcome the inherent variability of a high-throughput experiment or the need to statistically correct for the fact that tens-of-thousands of tests are being performed simultaneously [**13**]. They are also unlikely to know how to ensure that the massive data files are handled correctly, the code is error-free and robust, or that the analysis is reproducible. In contrast, computer scientists understand data handling and code testing, but likely not the biological variables at play or how they should be addressed in the analysis in order to produce biologically meaningful results. They may also be unaware of biological experimental design pitfalls or confounding variables that can affect the data [19].

Consequently, although students graduating with a masters' degree in bioinformatics are unlikely to immediately head their own research groups, they must take leading roles in ensuring that experiments are performed and analyzed correctly. Becoming effective leaders who can “lead up” to influence their superiors as well as peers and subordinates requires “soft skills” such as communication and leadership that are not typically included as learning objectives in bioinformatics curricula. To remedy this deficit, this paper describes a three-credit course on leading reproducible research. The course teaches the best practices for reproducible and robust research methodology as well as communication and leadership skills. While the course is designed for master’s level bioinformatics students, the concepts and activities can be adapted for undergraduate education.

2 Course Overview

The course on leading reproducible research is a core course in the Bioinformatics MS curriculum and has been offered by the author seven times over the last eight years. Along the way, the course has been continually updated to reflect the most current guidelines and best practices for research methodology and the stewardship of data and computational artifacts. The author completed a more extensive overhaul of the course in 2021 to adapt it for use as part of a graduate certificate in Project Management in the Life Sciences in addition to its inclusion in the Bioinformatics MS.

The course follows a “flipped classroom” format [2, 4] wherein most of the information is imparted to students outside of class through readings, videos, and weekly homework assignments. Class time is spent discussing and building on the ideas in small breakout groups and as a whole class, in addition to short lectures intended to refine student understanding and extend the concepts. Discussions are enriched by an ongoing activity that the author calls “Real World Problems”. The author provides index cards and a manila envelope for students to anonymously share concerns, challenges, and problems that they have encountered in the workplace. Depending on the concerns that are raised, these can be used as prompts for a focusing activity at the beginning of class, or can be pulled out later to enrich discussions of relevant topics in the course.

2.1 Instruction on Robust and Reproducible Research Practices

Like many of us in the hard sciences, the students are usually much more interested in learning technical considerations and “hard skills” for analysis than “soft skills” such as communication and leadership. Therefore, most of the focus of the first two-thirds of the course is on experimental design and analysis. However, the author takes advantage of openings in the class discussions to start students thinking about the connection between their scientific goals and good leadership.

The class starts with a discussion of respect, trust, and reliability. Students identify what makes research results reliable and trustworthy– namely that the results are robust, reproducible, and fairly reported. They also identify what makes them respect, trust, and rely on other people—honesty, integrity, hard work, and a willingness to mentor others. The first lecture introduces the reproducibility crisis and its consequences as well as the challenges that are inherent to bioinformatics research. We discuss the benefits and difficulties of working at the intersection of biology, computer science, and statistics. As a group, we talk about the cultural and communication differences between biologists, computer scientists, and statisticians and identify the key factors that influence their experimental design decisions. Finally, we consider different sources of error that can occur at each stage of an experiment that might be missed by each group. For example, a biologist who writes some code may not be aware of how to test their code and avoid silent errors, while a computational scientist may not recognize all of the potentially confounding variables that can affect the data.

Instruction on the best practices for reproducible research starts with computational reproducibility, the ability to fully reproduce the results of an experiment by using the same data, metadata and code provided by the original authors of a study [6, 13]. Students learn about the FAIR principles [24] and how to ensure that their data and analysis workflows are Findable, Accessible, Interoperable and Reusable. We also add an additional R by discussing how to make an analysis Robust in addition to Reusable. Robust workflows contain all essential software components and can be used on a variety of systems.

The next focus of the course is on experimental design considerations. Students learn about a variety of sampling methodologies, their benefits, and their drawbacks. They describe the relationships between sample size, effect size, sample-to sample variability, and significance thresholds relative to the power of an experiment to produce significant results. We discuss how and why the number of biological replicates improves the power of an experiment while technical replicates do not, and why technical replicates are still useful. Students identify many different types of confounding or “nuisance” variables that could influence the outcome variable that is being measured and learn how to deal with them through randomization, blocking, and regression analysis [19]. Students learn how perform exploratory data analysis and sanity checks to examine the integrity of the resulting data and how to remove outliers and build a statistical model. Finally, we discuss what should and should not be done in order to “save” an experiment in case no results are significant. While this discussion is based on best practices for statistical methodology as described in [**13**, 19], the topic also provides an opening to discuss research ethics and leadership. For example, students often describe unethical research practices that they have seen others perform or have been encouraged to perform themselves in order to improve the statistical significance of results. This sparks a practical discussion of ethical research practices and how students can "lead up" to their bosses and maintain their integrity while keeping their jobs.

2.2 Teaching Students to Assess the Quality of Work Performed by Others

Once students understand the principles of experimental design and computational reproducibility, they learn how to assess the quality of published work and publicly available data. The ability to determine whether published work is reliable is important for scientists of all kinds, but even more so for bioinformaticians because they often utilize published data as part of their own studies. Some students in our Bioinformatics MS program are fresh from their undergraduate studies while others are established researchers who already have PhDs, so this section of the course starts with optional articles on how to efficiently read a biological research article followed by instructions for how to review a manuscript for possible publication.

Students apply these instructions to review Anil Potti's infamous paper, "Genomic Signatures to Guide the Use of Chemotherapeutics" [17] as if it were a new manuscript submitted for publication. Although the paper contains many errors caused by a combination of carelessness and fraud, most students like the paper the first time that they read it. They change their minds as we discuss the results of the paper in class and they start to notice things like scatter plots that don't look linear or have an R^2 value, but the p value is suspiciously low. They also note that differences between groups are reported as significant although the error bars are huge and clearly overlap each other. The following assignment is for them to read through the exchanges between Potti and "forensic data scientists" Baggerly and Coombes who reported their inability to reproduce Potti's results [9]. Over the course of several letters to Nature, Baggerly and Coombes reveal a variety of careless mistakes and also some apparent fraud committed by the Nevins lab. In class, the author presents "Chemotherapeutic Signatures…" as if at a conference and has students interject with Baggerly and Coombes' arguments to point out the errors as we get to them. We conclude with a discussion of reasons why people cheat, the actions students can take if they become aware that their peers are behaving unethically, and how they may act in the future to prevent subordinates from committing fraud.

As a summative assessment for this section of the course, students apply what they learned by reviewing of one of a selection of retracted manuscripts obtained from the retraction watch database (Retractionwatch.org). Students identify the strengths and weaknesses of the work and recommend whether or not it should be published. After completing an initial analysis of the work, students share their thoughts in small groups. The author leverages the diversity of students in the class by grouping together students with different backgrounds and levels of experience in biology, computer science, and statistics to provide a variety of perspectives and insights. Finally, the students complete an in-depth written evaluation of the retracted article that they selected.

3 2.3 Teaching Students to be Effective Leaders

As previously mentioned, Bioinformatics students are often resistant to the idea of spending time learning about "soft skills". In addition, many do not see themselves as leaders and do not covet leadership positions. Their resistance usually fades over the course of the discussions described above. Students begin to realize how they can benefit from influencing others, and that leadership is not limited to directing subordinates. The author's aim is to capitalize on this by demonstrating that good leadership can take many forms depending on the situation and the individuals who are involved. Each week, students respond to writing prompts that force them to articulate how their perceptions of leadership and their own potential to be leaders change during this section of the course.

We start by examining leadership in the context of scientific research using an excellent book, "Managing Scientists" by Alice Sapienza [21]. Students read chapters on "The Condition of Being Different", "Understanding What Motivates You and What Motivates Others", "Understanding Your Leadership Style and That of Others", "Communicating Effectively" and "Dealing with Conflict" while completing homework assignments to guide and enrich their reading. In class, we highlight the characteristics and behaviors of effective and ineffective leaders as students share their experiences and discuss how to deal with different types of managers. We also differentiate between management and leadership – managers direct tasks while leaders influence people.

Our discussion of "The Condition of Being Different" highlights the impact of cultural differences based on age, race, gender, and also scientific discipline on the ways people interact and perceive the words and actions of others. Students share their experiences as we highlight how diverse perspectives can strengthen groups, then we relate those ideas to the multidisciplinary nature of bioinformatics research. In later chapters, students are often surprised by the outcome of exercises designed to reveal the extent to which they are motivated by the need for power, achievement, and affiliation in the workplace. Students also reflect on whether their own leadership styles tend to be task-focused or relationship-focused. Task-focused leadership styles work well for highly structured tasks while more open- ended tasks that require a lot of creativity benefit from relationship-focused leadership. Students explain why different modes of communication are best suited for different kinds of messages and describe how personal biases and perceptions can color the way that messages are received and interpreted. Finally, students demonstrate strategies for dealing with conflict using role play while responding to prompts taken from the semester-long “Real World Problems” activity.

The remainder of the course is spent exploring the characteristics and behaviors of good leaders by comparing and contrasting different philosophical perspectives on leadership. Students get a brief overview of Hellenic views on leadership by watching short videos explaining the views of Socrates, Plato and Aristotle. After discussing these views in class, students then participate in a lecture on Alexander the Great by identifying ways in which his behavior was influenced by Socrates', Plato's, and Aristotle's ideas. Students discuss the benefits of self-examination, surrounding themselves with peers whom they respect and from whom they can learn, showing empathy, exploring all things in moderation, and the lessons of Plato's Allegory of the Cave [16] among other ideas. Next, they identify passages in The Analects of Confucius [11] that illustrate concepts such as self-rectification, justice, benevolence, filial piety, long-term thinking, and the “golden rule”.

During a lecture expanding on these concepts, students discuss what these ideas mean in terms of how a leader should behave and compare them to those espoused by the Hellenics. Finally, students read either “The Tao Te Ching” [12] or the “Basic Writings” of Zhuangzi [28] and identify passages illustrating ideas such as seeing the good in bad things and the bad in good things, living in the moment while planning for the long term, finding balance between opposites, mindfulness, avoiding conflict, fostering harmony between individuals in a team, empowering others to succeed, and seeking results but not acclaim. Following a lecture expanding on these ideas, students compare and contrast them to the Hellenic and Confucian viewpoints, then identify concrete actions and behaviors that a Daoist leader would take in response to a variety of challenges. At the end of each class session, students apply the perspective being discussed to respond to prompts selected from the collection of “Real World Problems”.

4 Student Learning Objectives and Examples

Table 1 contains a listing of the student learning objectives for the course, an example assignment and readings related to each objective, and some of the insights that students are expected to attain:

5 Evaluation and Feedback from Students

The author has found this course to be simultaneously very challenging and rewarding to teach. The described course is one that students frequently argue against taking, claiming that they don't need it either because have no intension of being leaders or because they feel that they have nothing to learn. On the first day if class, many students arrive looking bored and reluctant to engage. However, most students become more enthusiastic by the end of the first session. Over the course of the semester, the class discussions become increasingly lively as the vast majority of students engage deeply with the material.

Student evaluations of the author's teaching are generally very strong with most scores (out of five) being in the high 4's or 5 in every category. However, a couple of students commented that the course was overly theoretical when it was first offered. Student engagement and performance on learning assessments has improved over the years as the author has increasingly found ways to help students understand general concepts in concrete terms by identifying actions that illustrate how they can be applied. Students particularly enjoy the "Real World Problems" activity and comment that they value the advice that they get in response to the concerns that they submit.

Perhaps more importantly, this course inspires many alumni to reach out in thanks. Alumni say that employers ask them during interviews how they will ensure that their research is meaningful and reproducible. Potential employers are very impressed when they can readily give a well-considered answer. Alumni have also written to the author to express how they applied the leadership concepts that they learned in order to overcome particular challenges that they faced.

5.1 Selected Comments from Student Evaluations

- "The instructor was contagiously enthusiastic and helped me take an interest in a subject that previously thought to be mundane. I am very glad I took this course."

- "I like way she related topics and gives real world examples."

- "In context of leadership I love the discussion topics she provided really interesting topics."

- "After attending her lectures the way I saw and think completely changed. After Reading different percepts I realize that we have many things to learn from our own experience and as well books... This course actually helped me to understand concepts of leadership and their applications in real world."

- "Dr. Darby has continuously been enthusiastic and encouraging. This class was one of the best classes that I have taken at Hood and I really learned a lot from the other people in the class due to the diverse age range. I always got the impression that Dr. Darby cared a lot about how we were outside of school and hoped that we learned instead of caring just about a grade. She gave great advice regarding how to stand up for yourself in various situations and encouraged us to speak in class because she was actually/actively interested in what I had to say. I am really grateful that I took this class and got to know Dr. Darby more during this semester."
- "Appreciated the Professor bringing in different ideas/topics to the class on the leadership"
- "Thank you very much! I enjoyed almost all the reading and I learned a lot about how I interact as a researcher with other researchers"
- "Even though this course was less focused on STEM subject matter, it gave me the information I needed to confront the STEM workforce with all of its unique challenges"

References

[1] S. Rachid Zaim A. A. Kim and V. Subbian. "Assessing reproducibility and veracity across machine learning techniques in biomedicine: A case study using TCGA data". In: *International Journal of Medical Informatics* 141 (Sept. 2020), p. 104148. DOI: 10.1016/j.ijmedinf.2020.104148.

[2] M. I. Baig and E. Yadegaridehkordi. "Flipped classroom in higher education: a systematic literature review and research challenges". In: *International Journal of Educational Technology in Higher Education* 20.1 (Nov. 2023), p. 61. DOI: 10.1186/s41239-023-00430-5.

[3] M. Baker. "First results from psychology's largest reproducibility test". In: *Nature* (Apr. 2015). DOI: 10.1038/nature.2015.17433.

[4] J. Bergmann and A. Sams. *Flip your classroom: reach every student in every class every day*. Eugene, OR: International Society for Technology in Education, 2012.

[5] V. Buffalo. *Bioinformatics Data Skills*. First edition. Sebastopol, CA: O'Reilly Media, 2015.

[6] B. Grüning et al. “Practical Computational Reproducibility in the Life Sciences”. In: *Cell Systems* 6.6 (June 2018), pp. 631–635. DOI: 10.1016/j.cels.2018.03.014.

[7] Jeffrey Brainard Jocelyn Kaiser. *Ready, set, share: Researchers brace for new data-sharing rules.* Accessed: Sep. 03, 2024. URL: https://www.science.org/content/article/ready-set-share-researchers-brace-new-data-sharing-rules.

[8] F. Ju and T. Zhang. “Experimental Design and Bioinformatics Analysis for the Application of Metagenomics in Environmental Sciences and Biotechnology”. In: *Environmental Science and Technology* 49.21 (Nov. 2015), pp. 12628–12640. DOI: 10.1021/acs.est.5b03719.

[9] J. Wang K. R. Coombes and K. A. Baggerly. “Microarrays: retracing steps”. In: *Nature Medicine* 13.11 (Nov. 2007), pp. 1276–1277. DOI: 10.1038/nm1107-1276b.

[10] S. Kapoor et al. “REFORMS: Consensus-based Recommendations for Machine-learning-based Science”. In: *Science Advances* 10.18 (2024), eadk3452. DOI: 10.1126/sciadv.adk3452.

[11] Q. Kong and B. Watson. *The Analects of Confucius*. Translations from the Asian Classics. New York: Columbia University Press, 2010.

[12] B. Watson Laozi, S. Addiss, and S. Lombardo. *Tao Te Ching*. Indianapolis: Hackett Publishing Co, 1993.

[13] P. Poulain M. Ziemann and A. Bora. “The five pillars of computational reproducibility: bioinformatics and beyond”. In: *Briefings in Bioinformatics* 24.6 (Nov. 2023), bbad375. DOI: 10.1093/bib/bbad375.

[14] R. Normand and I. Yanai. “An Introduction to High-Throughput Sequencing Experiments: Design and Bioinformatics Analysis”. In: *Deep Sequencing Data Analysis*. Ed. by N. Shomron. Totowa, NJ: Humana Press, 2013, pp. 1–26. DOI: 10.1007/978-1-62703-514-9_1.

[15] B. A. Nosek et al. “Replicability, Robustness, and Reproducibility in Psychological Science”. In: *Annual Review of Psychology* 73 (Jan. 2022), pp. 719–748. DOI: 10.1146/annurev-psych-020821-114157.

[16] Plato. *The Republic*. Classical Era.

[17] A. Potti et al. “Genomic signatures to guide the use of chemotherapeutics”. In: *Nature Medicine* 12.11 (Nov. 2006), pp. 1294–1300. DOI: 10.1038/nm1491.

[18] B. Pulverer. “Reproducibility blues”. In: *EMBO Journal* 34.22 (Nov. 2015), pp. 2721–2724. DOI: 10.15252/embj.201570090.

[19] G. P. Quinn and M. J. Keough. *Experimental Design and Data Analysis for Biologists*. Cambridge University Press, 2002. URL: `https://www.book2look.com/book/ge5gzy6TUg`.

[20] S. Samuel and D. Mietchen. "Computational reproducibility of Jupyter notebooks from biomedical publications". In: *GigaScience* 13 (Jan. 2024), giad113. DOI: `10.1093/gigascience/giad113`.

[21] A. M. Sapienza. *Managing Scientists: Leadership Strategies in Scientific Research*. 1st. Wiley, 2004. DOI: `10.1002/047166359X`.

[22] A. Trisovic et al. "A large-scale study on research code quality and execution". In: *Scientific Data* 9.1 (Feb. 2022), p. 60. DOI: `10.1038/s41597-022-01143-6`.

[23] M. Wang and Q. Long. "Addressing Common Misuses and Pitfalls of P values in Biomedical Research". In: *Cancer Research* 82.15 (Aug. 2022), pp. 2674–2677. DOI: `10.1158/0008-5472.CAN-21-2978`.

[24] M. D. Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific Data* 3.1 (Dec. 2016), p. 160018. DOI: `10.1038/sdata.2016.18`.

[25] M. D. Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific Data* 3.1 (Mar. 2016), p. 160018. DOI: `10.1038/sdata.2016.18`.

[26] J.-B. Poline Y.-M. Kim and G. Dumas. "Experimenting with reproducibility: a case study of robustness in bioinformatics". In: *GigaScience* 7.7 (July 2018). DOI: `10.1093/gigascience/giy077`.

[27] M. Zaringhalam and L. Federer. *Data and Code for Reproducible Research: Lessons Learned from the NLM Reproducibility Workshop*. May 2020. DOI: `10.5281/zenodo.3818329`.

[28] Zhuangzi and B. Watson. *Basic Writings*. Translations from the Asian Classics. New York: Columbia University Press, 1996.

LEARNING OBJECTIVE	EXAMPLE ASSIGNMENT	EXAMPLE INSIGHTS
Understand what makes research robust and reproducible	Student participation in lecture and discussion, reading on FAIR principles [18, 23]	Considerations for testing and sharing code, data, and metadata that are robust
Describe the principles of experimental design for biological research	Quiz on experimental design lecture, videos, and excerpts from book [17]	How to make statistical power calculations, identify and handle confounding variables of different types
Identify sources of error that are endemic to bioinformatics research	Participation in lecture and class discussion. Questions on reading from [**13**, 5]	How to handle data, test code for silent errors, apply statistical tests correctly
Demonstrate best practices for handling and preservation of data and computational resources (FAIR principles)	Students annotate workflow with commentary on how to correct likely failure points	Data acquisition, analysis steps should be scripted, figures generated in script, robust data formats used
Describe how to use exploratory data analysis to identify common errors in datasets	Participation in lecture and class discussion of common errors and the plots that reveal them, i.e. PCA	Starting data should be analyzed for mislabeling, missing values, off-by-one errors, inconsistent coding
Interpret and critique published literature	Analysis of retracted article by A. Potti [15] and chosen article from RetractionWatch.org	It is important to compare claims in article to the data that is shown, identify errors and overstatements
Communicate experimental procedures, results, and discussion orally and in writing	Participation in discussion and written analysis of retracted article by A. Potti [15] and RetractionWatch.org	Students articulate each of the claims made in the paper and discuss the supporting evidence for each claim.
Practice effective written and oral communication skills	Participation in discussion and lecture on modes of communication and how to overcome challenges. Chapter on effective communication [20]	Different modes of communication are better depending on the audience and type of information being conveyed.
Articulate challenges of planning and executing a project in a multidisciplinary environment	Participation in discussion and lecture on "cultural" differences between computer scientists, statisticians, and biologists. Reading from [5, 20]	Each group has different challenges, biases, goals, and vocabularies. Bioinformaticians become central communicators
Understand the nature of good leadership and identify key characteristics of good leaders	Many lectures, interactive discussions, and readings [**25**, 14, 20, 26] comparing modern western leadership styles to those of the ancient Hellenics, Confucians, and Daoists. Weekly written reflections.	Central themes include surrounding oneself with the best people, learning from them, treating others with respect, empathy, supporting others' needs without micromanaging.
Articulate the differences between leadership and management	Participation in discussion and lecture on leadership versus management. Readings [20]	Management relates to the completion of tasks and projects. Leadership is about people.
Appreciate the importance of ethics in research	Participation in discussion of retracted articles by A. Potti [15] and from RetractionWatch.org	Scientists' reputations are important assets that can't be easily rebuilt. Bad science hurts everyone.
Define what ethical leadership means to them	Many lectures, interactive discussions, and readings [**25**, 14, 20, 26] comparing modern western leadership styles to those of the ancient Hellenics, Confucians, and Daoists. Weekly written reflections.	It is essential to be honest, treat others well, protect public welfare, act with empathy, and acknowledge others' contributions
Assess where they are on their own continuum of leadership	Many lectures, interactive discussions, and readings [**25**, 14, 20, 26] comparing modern western leadership styles to those of the ancient Hellenics, Confucians, and Daoists. Weekly written reflections.	Many students who think that they lack leadership potential and the desire to lead end up realizing that they are already acting as good leaders. Students articulate leadership goals.

Figure 1: Student Learning Objectives.

Reviewers — 2024 CCSC Eastern Conference

Prof. Alvin Chris ....Furman University, Greenville, SC, United States
Dr. Anewalt Karen .... University of Mary Washington, Fredericksburg, VA, United States
Dr. Bany Muhammad Nooh .....Frostburg State University, Frostburg, MD, United States
Dr. Barland Ian ....Radford University, Blacksburg, VA, United States
Prof. Carter Karla ....Bellevue University, Bellevue, NE, United States
Dr. Childs Dawn ....Marymount University, Cypress, CA, United States
Dr. Conrad Sue ....Marymount University, Fairfax, VA, United States
Dr. D'Antonio Lawrence ..Ramapo College, Dobbs Ferry, NY, United States
Dr. Dimitoglou George ....Hood College, Frederick, MD, United States
Dr. DiTursi Dan ..Rensselaer Polytechnic Institute, Troy, NY, United States
Prof. Dougherty John ......Haverford College, Haverford, PA, United States
Dr. Finlayson Ian . The University of Mary Washington, Fredericksburg, VA, United States
Dr. Flinn Michael .Frostburg State University, Frostburg, MD, United States
Dr. Freedman Reva ..Northern Illinois University, DeKalb, IL, United States
Dr. Grinberg Grigoriy ......Montgomery College, Gaithersburg, MD, United States
Dr. Gupta Pranshu ....DeSales University, Center Valley, PA, United States
Dr. Highley Timothy ...La Salle University, Philadelphia, PA, United States
Dr. Hovemeyer David .... Johns Hopkins University, Baltimore, MD, United States
Dr. Lamprecht Ruth Mount St. Mary's University, Emmitsburg, MD, United States
Dr. Lee Ingyu .... Troy University, Troy, AL, United States
Dr. Li Jiang ....Hood College, Frederick, MD, United States
Prof. Lindoo Edward ....CCSC Treasurer, Stuart, FL, United States
Prof. Lopez Christian .... Lafayette College, Easton, PA, United States
Dr. McCloskey Robert W ..... University of Scranton, Scranton, PA, United States
Dr. McManus John ...Randolph-Macon College, Ashland, VA, United States
Dr. Ngo Linh .. West Chester University of Pennsylvania, West Chester, PA, United States
Dr. Nuakoh Emmanuel Borkor ....North Carolina AT State University, Shrewsbury, MA, United States
Dr. Nuangpookka Pipop ..Bay Atlantic University, Washington, DC, United States

Dr. Poger Sofya ......Felician University, Woodland Park, NJ, United States
Ms. Ravishankar Veena University of Mary Washington, Fredericksburg, VA, United States
Mr. Sanders Patrick SUNY University at Buffalo, Buffalo, NY, United States
Dr. Senbel Samah Ahmed ....Sacred Heart University, Fairfield, CT, United States
Dr. Stange Melissa Shenandoah University, Stephens City, VA, United States
Mr. Stedman Jacob Daniel .....Marymount University / Advanced Circuits, Osseo, MN, United States
Prof. Teresco James D. ....Siena College, Loudonville, NY, United States
Dr. Tu Junyi ....Salisbury University, Salisbury, MD, United States
Dr. Webb Alla ......Montgomery College, Gaithersburg, MD, United States
Prof. Wonnacott David Glenn .... Haverford College, Haverford, PA, United States
Prof. Wright John ....Juniata College, Huntingdon, PA, United States
Ms. Xu Wenjuan . Frostburg State University, Frostburg, MD, United States