Adaptive Feedback on Web Help-Seeking Behaviors for Novice Programmers

Novice programmers who are writing code encounter frequent challenges, which they often attempt to address by searching online to learn new concepts or debug their code [3]. However, novice programmers rarely receive explicit instruction on how to effectively resolve programming challenges with online search. Seeking help through online search is a critical self-regulatory skill for students, both in class and after graduation, but professional programmers agree that it can be difficult to learn [8]. Some prior work has investigated the challenges that *professional developers* face when searching and how to support them. However, there is little research about how programmers learn to use online search, how to teach search effectively, or how adaptive tools can support this process. Further, prior work has been limited to collecting user search queries and developer surveys, without using the programmer's code and errors to understand their context and the reason for their search.

My research goal is to 1) understand the strategies that undergraduate novice programmers use to search for help online when writing and debugging code, and 2) explore how to design adaptive learning environments that support students in learning effective online search strategies.

Building on Prior Work: Collecting data from novices' programming environments has become popular in Computer Science (CS) Education due to their ability to provide granular views into the programming process via incremental code snapshots. These code snapshots have been leveraged to extract features such as compiler error encounter rate and error resolution time, which are used to understand novice debugging behaviors. In addition, code snapshots can be used in the design of interventions that dynamically provide feedback during the programming process based on current and past code contexts. It has been shown that intelligent tutoring systems (such as augmented programming environments) that provide timely automatic feedback can positively affect learning [5]. Work has been done on augmenting learning environments with tools such as web browsers that filter search results to be more understandable by novices [2,7]. However, many of these tools focus on improving search *results*: **no tools have been developed with a focus on improving code search *behavior* and imparting generalizable long-term search skills.**

RQ1: *What barriers do students face and what strategies do they employ when using web-based search systems to write and debug code?* In Year 1, I will conduct exploratory lab and classroom studies to identify common barriers that novices encounter and strategies that they use for online search during programming problems. **My goal is to discover what unique problems novices encounter when using online search, inform pedagogy on how online search should be taught, and learn what features to consider when designing a tool to support better search behavior**. In small scale lab studies during the first semester, novice programming students will be asked to solve programming problems appropriate to their skill level while having access to online search. I will collect think-aloud protocols as students work, pre- and post-surveys, logs of students' search behavior, and fine-grained code snapshot logs. Classroom studies of NCSU's introductory computing (CS1) course during the second semester utilize augmented programming environments and provide a large volume of search and code snapshot data, which will allow us to test if the findings from the lab studies are generalizable to the classroom. Using this data, we can evaluate how search behavior from beginner programmers may differ from or align with prior work on professional developer behavior, such as in query style, query reformulation rate, and search session length. Building on prior work on using code snapshots to track compiler error resolution [1], we will explore how to track students' success in using online search to resolve errors. This will allow us to identify which students are struggling, where they are struggling, and what search strategies are effective.

RQ2: *How can we give students accurate and timely feedback on how to improve the effectiveness of their search queries?* In Year 2**, I will perform design-based research of a learning environment that will provide automated support to students to improve their search skills.** While the ultimate design of the system will be shaped by student and teacher needs identified in Year 1, the system will support developing students' skills in the key phases of the search process. The key phases of search are derived

from theories of help-seeking (e.g. [4]): recognizing the need for help, gathering relevant information, formulating an effective query, identifying an effective source of help, and integrating the help into code. For example, consider a student who is stuck on an error and whose previous search attempts were not successful. The system could then suggest an effective help-seeking strategy that was identified in RQ1, such as query reformulation. The system will offer adaptive help, responding to a student's current code and error message. I will achieve this by building on foundational work by the HINTS lab at NCSU on using program code to create adaptive feedback systems [6]. **My current membership in the HINTS lab puts me in a unique position to design this system.**

RQ3: *What effect does timely automated feedback on search queries have on novice programming behaviors and their ability to use web-based search to resolve future problems?* After multiple rounds of development in Year 2, **in Year 3, I will deploy the system and perform classroom studies to measure the impact this tool has on novices' search and programming behavior.** These classroom studies will initially take place over the course of a semester in NCSU's CS1 course. The effectiveness of the intervention will be measured by the change in student programming performance and retention of help-seeking behaviors over time (as evidenced by features of successful help-seeking identified in RQ1, such as query styles and error resolution rate). In addition to quantitative metrics, qualitative feedback by students at the end of the semester will also inform the system's overall impact.

Intellectual Merit: This project will provide novel insight on the strategies that undergraduate novice programmers use to search for help online. By extending prior work on programmer search behaviors through the novel methods of observing program and error state, we can gain granular information about the contexts in which novice programmers seek help. In exploring effective search strategies (RQ1), we can inform the design of better pedagogy for teaching online search. The feedback methods designed in RQ2 and evaluated in RQ3 will inform future designs of intelligent tutors for help-seeking. This project will provide a base for future work on understanding programming help-seeking behavior, and the methods discovered could be extended to learner populations beyond novices.

Broader Impact: In many CS programs, searching for code help online is a skill that is not explicitly taught, yet is expected of students in upper-division courses and after graduation. Explicit pedagogy around web search will normalize help-seeking behavior and reduce impostor syndrome. Automated feedback systems allow for the refinement of these skills even if an instructor is unavailable. This project will inform future online search pedagogy and the design of learning environments that reinforce these skills. By addressing these two issues, this project will help **make CS more accessible**.

[1] *Jadud. "Methods and Tools for Exploring Novice Compilation Behaviour" ICER'06* ***[2]*** *Lu et al. " " Information Retrieval Journal '17* ***[3]*** *Muller, et al. Exploring Novice Programmers' Homework Practices: Initial Observations of Information Seeking Behaviors SIGCSE '20* ***[4]*** *Nelson-LeGall. "Help-Seeking: An Understudied Problem-Solving Skill in Children" Developmental Review '81* ***[5]*** *Nesbit, et al. "Intelligent Tutoring Systems and Learning Outcomes: A Meta-Analysis" Journal of Educational Psychology '14* ***[6]*** *Price, et al. "iSnap: Towards Intelligent Tutoring in Novice Programming Environments" SIGCSE '17* ***[7]*** *Venigalla et al. "StackDoc - A Stack Overflow Plug-in for Novice Programmers that Integrates Q&A with API Examples" ICALT'19* ***[8]*** *Xia, et al "What do developers search for on the web?" Empir Software Eng '17*