The Future of Engineering Education
2024 Annual Conference & Exposition
Oregon Convention Center ASEE
Portland, OR . June 23 - 26, 2024
Paper ID #41329

Leveraging Peer-Authored Tutorials to Cultivate Programming Skills and Promote Open Educational Resources: A Multi-Classroom Case Study

Dr. Dirk Joel-Luchini Colbry, Michigan State University

Dr. Dirk Colbry is a faculty member in the Department of Computational Mathematics, Science and Engineering (CMSE) at Michigan State University. Dr. Colbry earned his Ph.D. in Computer Science and his principle areas of research include scientific image understanding, large scale computing and education.

Leveraging Peer-Authored Tutorials to Cultivate Programming Skills and Promote Open Educational Resources: A Multi-Classroom Case Study

Abstract

The rapid evolution of the computational and technological landscape poses a significant challenge for educators in computer and data science. Keeping pace with the ever-changing tools and technologies is an ongoing struggle, and the ability for students to become self-reliant learners, adapting to new tools, is crucial. This paper introduces a pedagogical approach that leverages student-authored tutorials to cultivate programming skills and promote open educational resources (OER). The approach has been implemented across diverse classroom settings, including a summer research program, multiple years of an undergraduate data science capstone course, and a graduate special topics course on generative AI (Artificial Intelligence).

Motivation

The dynamic nature of the computation and technology landscape necessitates a shift in teaching strategies. Traditional methods of having students master specific tools are rapidly becoming obsolete, prompting the need to refocus learning goals on teaching students how to adapt and learn new tools. The "protégé effect," emphasizing that students learn best by teaching, forms the basis of the approach outlined in this paper. This student-centric strategy aims to address the challenge of keeping up with evolving tools by having students collaboratively identify and write tutorials for useful of-the-moment tools.

The key learning objective of this approach is to help students understand the importance of exploring and using current and emerging tools as part of their lifelong education. The specific tools can vary a lot depending on individual classroom learning goals, resulting in a wide range of student-authored tutorials. Some examples from the author's classes include:

- Setting up ChatGPT to help write code in Jupyter notebooks.
- Building and deploying your own Shiny App.
- Accessing the US census API in Python.
- Downloading and installing Seaborn to make more robust figures.

Students are tasked with creating in-depth tutorials designed to help their peers learn to use the software tools effectively. Creating successful tutorials requires that student authors both understand the tools and effectively communicate their functionality to peers. These assignments culminate in the collaborative curation of a git repository that serves as a valuable resource for current and future students [1]. Importantly, these new tutorials are shared under a Creative Commons license and provided as Open Educational Resources (OER), allowing free access by learners worldwide [2].

This paper describes the structure of the tutorial development assignment and the steps involved, sharing insights and case studies on how to implement this approach successfully in different classroom settings. The methodology for assessing the assignment's effectiveness is discussed. Additionally, this paper addresses the transferability of this approach to a broad range of programming and data science courses, highlighting its adaptability and the benefits of contributing to the OER ecosystem.

The outcomes of this multi-classroom case study offer valuable insights for educators seeking to enhance their students' tool fluency, self-directed learning capabilities, and collaboration skills while also contributing to OER. By emphasizing the importance of figuring out tools and the creation of comprehensive tutorials for peers, this pedagogical approach not only equips students with essential technical skills but also fosters a culture of mutual support and knowledge sharing in the classroom, contributing to a broader educational community.

Methodology

This section introduces the concept of a **Student-Crafted Hub for Open Learning and Academic Resources (SCHOLAR)**. The motivation behind SCHOLAR is to encourage students to engage in self-directed learning and transfer their knowledge by writing tutorials that will teach others. These tutorials become a reusable resource that the student authors can keep as a future reference, while also building a repository of tutorials that are useful to the instructor and future students. In fields like computer and data science, where tools and libraries are constantly changing, the ability to adapt approaches and learn new tools quickly and efficiently is an essential skill. The SCHOLAR approach stresses the importance of students filtering through vast information, learning how to use tools for specific problems, and developing skills for navigating the ever-evolving technology landscape. Students also learn how to write coherent tutorials that are complete and thorough enough for their peers.

The SCHOLAR assignment employs an open-source platform, utilizing git version control and Jupyter notebooks for organizing the tutorials [3]. Jupyter notebooks are an open-source file format designed specifically to encourage communication. Jupyter notebooks effectively present ideas using formatted text, LaTeX equations, images, video, and executable code. They are an ideal tool for communicating complex ideas, and thus Jupyter notebooks are extremely suitable as a base file type for any type of tutorial.

Git is a version control system that is used by some of the world's biggest open-source software projects and is specifically designed to enable multiple authors to work together on any text-based file format. There are many online repository management systems that can help git users share their files; the most common one is github [4], but there are many others such as gitlab [5] and bitbucket [6]. For the Jupyter notebook tutorial repositories described in this paper, students use our university-hosted gitlab instance. Each student in the class automatically has an account in this gitlab that is linked to their student ID, which makes tracking and grading assignments much simpler.

The SCHOLAR methodology involves a five-step cycle, each assigned as a course task:

1. **Generate a new Tutorial and issue a pull request.** Students pick a software library or tool that they need to use in their projects and write a comprehensive tutorial for the library/tool. This tutorial should include the steps needed to install the software and get it working using a simple example. When they are done, students use the git concept of a "Merge" or "Pull request" to submit it to the main repository.
2. **Review and merge the new tutorial into the main branch.** Students review each other's work in the second step. The reviewer must follow the directions outlined in the tutorial and successfully get it working. The reviewer must also ensure that the changes meet the coding standards set up in the class, which typically cover simple things like file naming conventions and ensuring that only files needed for the tutorial are included in the

repository. The instructors or the students can then merge the solution into the main repository.

3. **Review existing tutorials and Submit an Issue.** Since software is constantly being updated and changed, the tutorials authored by previous classes need to be regularly reviewed and updated to ensure they remain valid. This assignment helps with this data curation process. Almost all online git repositories have a mechanism for users to submit "issues," which is just a way to identify problems that need to be fixed in a repository. In this step of the SCHOLAR assignment, students find a mistake or recommended improvement in a tutorial and submit the "issue" to the repository. The issue should say where the mistake is and give sufficient details for someone else to be able to correct the mistake.
4. **Review current issues, fix the issue, and submit the fix as a pull request.** In this step, students go through the issues that have been submitted and pick one or more that they can fix. Students then proceed to fix the issues in a "branch" and push that branch to the main repository as another pull or merge request. The comment for the pull request should clearly indicate which issue is being fixed by identifying it using a hash tag and issue number.
5. **Merge a pull request into the main branch.** Finally, either the instructor or the students should review the merge requests and accept or reject them based on how well the issue was resolved. This is essentially a repeat of the tasks in step 2, but substitutes fixing the existing tutorials instead of adding newly authored ones.

This cyclical approach (see Figure 1) allows students to continuously contribute to and improve the SCHOLAR repository, reinforcing their skills in learning, collaborating, and adapting to new tools. The ordering of the steps can be easily adjusted to meet the varying skillsets and learning goals of different class settings. For example, students with less technical experience might start by reviewing existing tutorials, submitting issues, and learning to navigate the git repository before authoring and submitting their own tutorials.

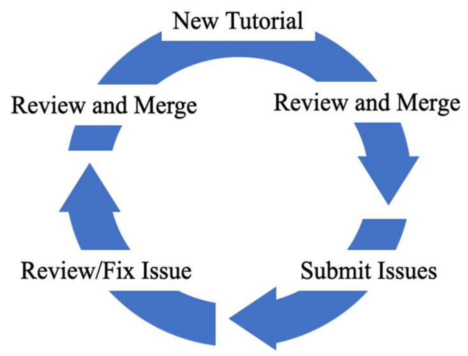

Figure 1: Five-Step Assignment Cyclic Cycle: New Tutorial, Review and Merge, Submit Issues, Review/Fix Issue, Review and Merge

The next three sections describe case studies that illustrate the effectiveness of the SCHOLAR assignment in different classrooms. These case studies include an undergraduate capstone course, a bridge program for incoming students in a Master of Data Science (MSDS) program, and a graduate special topics course in generative AI (Artificial Intelligence). The case studies demonstrate how engaging students in generating and evaluating tutorials authored or updated by their peers enhances their learning and increases students' adaptability to new tools and techniques.

Case Study: MSDS Bridge Curriculum

In the summer of 2022, a team of six incoming graduate students spent the summer preparing to start a Master of Science in Data Science (MSDS) program. This MSDS program was newly launched at Michigan State University in the fall of 2022, and these six students were funded as part of an NSF workforce development project (NSF #2123260) to spend the summer preparing for success as graduate students. In addition to studying math and programming, these students helped curate some self-guided tutorials for the other incoming MSDS students who would arrive in the fall.

This MSDS program recruited students from a wide range of backgrounds, and there was considerable concern among the faculty that not all students would arrive with a common understanding of core topics and skills necessary for success in the new MSDS program. The six students who arrived early had a variety of disciplinary backgrounds as undergraduates and were not yet experts in data science. These six students worked with faculty over the summer to brainstorm key topics and skills that incoming students would need to know for the first year of classes. The students then reviewed and tried various online learning resources in these areas and compiled the ones that they found most useful for themselves and their incoming peers. These tutorials were added to a shared repository and provided to all incoming MSDS students the week before classes started when they convened for orientation and onboarding activities. These included a workshop designed and led by the six summer students, who taught their peers how to download the repository of resources and get it working on their individual laptops.

This summer program was an early inspiration for what became the SCHOLAR approach. The summer program did not involve formal classes and assignments, and the six students curated a repository of existing resources instead of authoring entirely new tutorials. However, the materials and resources that the students collated highlighted many of the computational and communications tools that were important for the MSDS program. The summer program also gave the faculty an opportunity to help students develop and manage a shared repository and highlighted many of the practical and logistical issues that needed to be resolved as the SCHOLAR approach evolved [7].

https://gitlab.msu.edu/CMSE/data_science_bridge_curriculum

Case Study: Undergraduate Data Science Capstone

Starting in the spring of 2023, the SCHOLAR approach was integrated into the data science capstone course at Michigan State University (MSU). Data science is a new undergraduate major at MSU, with the first senior-level capstone design course offered in spring 2022. This course typically serves 60 students divided into 12 project teams working with community partners on a variety of real-life data science challenges. During the first year, it became apparent that students both needed tutorials and resources for learning various computational and analytical tools, and needed practice developing clear written communications. In the second year (2023), the SCHOLAR method was introduced to help address both learning objectives.

The capstone is a high engagement, 4-credit course that meets three times a week for an hour and twenty minutes per class, with students expected to spend substantial out-of-class time working together on their group projects. One class day a week (typically Friday) is set aside for the SCHOLAR assignments. The tutorials collated during summer 2022 for the MSDS students were used as a starting point for the first capstone course, and during class on Fridays each student was expected to work with the repository. The SCHOLAR approach was adjusted for the capstone course as follows:

1. Review a tutorial and submit an issue.
2. Review the issues, pick one, resolve it, and submit the solution as a pull request.
3. As a Team, write a new tutorial and submit it as a pull request.

In the first year of using the SCHOLAR approach as part of the capstone course, the instructors managed all the merge requests. This turned out to be time-consuming, as every request had to be carefully reviewed to determine whether the student team's proposed solution actually resolved the issue (or whether their new tutorial worked). In 2024, the second time the SCHOLAR assignments were used, the instructors experimented with a different approach. A team of six student volunteers was identified to work with the instructors to review and issue the merge requests. These students took on this extra responsibility as an "Honors Option" for the class (an option offered at Michigan State allows students to earn Honors credit in a regular, non-Honors course). The students are given extra instructions on how to work with git and the "git management team" shared what they learned about git with their project teams. Once all the assignments have been submitted and merged, the "git management team" reviewed and organized the tutorials to make the entire repository easier to use and understand, and to help prepare it for use by future students [8].

https://gitlab.msu.edu/CMSE/datatools_tutorial_demo

Case Study: Graduate Special Topics Course

In the fall of 2023, the Computational Mathematics, Science and Engineering program at Michigan State University offered a graduate-level Special Topics course on using Generative AI in scientific discovery. This course was open to graduate students from across the university and enrolled 12 students from 6 majors. Four faculty volunteered as the instructional team and the course was modeled after two previous "emerging technologies" classes taught in the same department. The idea behind all of these "emerging technologies" special topics courses is that they specifically introduce something that is not only new to the students, but may also be new to the instructors. For example, previous "emerging technologies" courses covered NextGen GPU programming, FPGA programming for scientific computing, and utilizing a unique large scale location dataset. In all these cases, the instructors were not the experts but acted more as guides to graduate student learning.

To foster co-created knowledge and a shared learning process, previous versions of this course had students add notes to a wiki during the semester. This shared recording space allowed students to reflect on and build off each other's knowledge. Expanding on this constructivist approach, the SCHOLAR model was introduced in the "emerging technologies" course for fall of 2023, focusing on generative AI. Instead of a wiki, each student built a tutorial about some aspect of generative AI and/or its use in scholarly research. Examples included:

- Gradio Library Tutorial.
- Prediction Protein Structure from Amino Acid Sequence using AlphaFold – Tutorial.
- How to use ChatGPT in Jupyter Notebook.
- TSNE Tutorial: Visualizing & Exploring High-Dimensional Data in 2D/3D.

Students shared their tutorials with each other in a git repository using the five-step SCHOLAR approach previously described. Each tutorial was written and published freely as an open educational resource, which not only allowed the students to use them in class but to also share outside the classroom to enhance their science and research. Since learning to use git was not a priority for the course, we adjusted the SCHOLAR steps such that students turned in their tutorials via the course management system and the instructor added them to the git repository. The full repository was made available to the students as a reference (with basic instructions for accessing the contents, for students unfamiliar with git), and this repository can be used as a starting point when the course is taught again [9].

https://gitlab.msu.edu/CMSE/Gen_AI_Tutorials

Lessons Learned

In this section we will talk about some of the lessons learned when implementing the SCHOLAR approach in various classrooms. By navigating these lessons learned, educators and students alike can optimize the SCHOLAR assignment for effective learning, collaboration, and the creation of valuable open educational resources.

Navigating the Learning Landscape: One of the fundamental aspects of the SCHOLAR assignment is recognizing that students do not have to create tutorials entirely from scratch. The internet is filled with useful resources (as well as unhelpful ones) such as videos, blogs, and Stack Overflow comments that students can leverage to unravel complex topics. The SCHOLAR assignments aim to teach students how to navigate this information deluge effectively and encourage students to select and organize useful pieces in a coherent manner to create learning resources for themselves and their peers. An essential skill cultivated through this process is understanding when and how to cite sources in digital environments; at a minimum, students are required to provide links to the original sources.

Teaching git and Collaboration Etiquette: The SCHOLAR assignments emerge as effective vehicles for teaching git and its associated etiquette in the classroom. Git, a powerful yet complex tool, can be used on three major levels: as a consumer, as a personal backup system, and as a collaborative tool between teams. While many individuals are initially introduced to git as consumers, forking or cloning repositories to use others' files, the SCHOLAR assignment encourages students to progress to using git for their software projects. The assignment emphasizes the development and use of key skills like tracking changes, commenting on modifications, and adopting git as a large-scale collaboration tool. The student-generated SCHOLAR repository, which includes contributions from multiple classes, facilitates the teaching of best practices for working on extensive collaborative projects.

While the SCHOLAR approach can be advantageous when learning git is one of the course objectives, that is not always the case. The complexity and power of git means that it can be frustrating or even distracting for students when learning this collaboration tool is not a key component of the course (as with the generative AI course described above). Considering this, the SCHOLAR assignment allows flexibility for students to submit their tutorials in alternative ways, with the instructors ultimately responsible for synchronizing the repository. This adaptability ensures that the focus of the SCHOLAR approach remains on the overall learning goals, while acknowledging the varied skill levels of students in different classes.

Automated Grading and Tracking Student Progress with Git: Git's complexity also proves advantageous in automating aspects of grading and tracking student progress. Git repositories automatically log students' actions, enabling the development of scripts that facilitate the evaluation process. For instance, scripts can identify when students complete specific assignments by analyzing submitted issues or merge requests. While these scripts may not evaluate the quality of submissions, they significantly expedite the identification of students encountering difficulties, streamlining communication and support for instructors.

Challenges with Jupyter Notebooks and Git Integration: Integrating Jupyter notebooks with git repositories introduces challenges due to the unique combination of source- and program-generated information in Jupyter files. The inherent issue arises when running a Jupyter file adds output cells, triggering git to perceive significant changes incorrectly. To mitigate this, a simple yet crucial step is introduced – clearing all output cells before committing any changes to Jupyter notebook files. Although this additional step may seem tedious and students occasionally forget it, it plays a vital role in minimizing noise and bloat within git repositories. This is a well-known challenge in the Jupyter community.

Discussion and Future Work

The outcomes of this multi-classroom case study emphasize the value of the SCHOLAR assignment in enhancing students' tool fluency, self-directed learning capabilities, and collaboration skills. The approach not only equips students with essential technical skills but also fosters a culture of mutual support and knowledge sharing. By contributing to the OER ecosystem, students become active participants in the broader educational community, ensuring the sustainability and adaptability of their learning resources. This paper serves as a guide for educators seeking innovative ways to empower students in navigating the ever-changing landscape of computational tools and technologies.

Although the open nature of the assignment encouraged student creativity, as the repository becomes bigger it becomes harder to organize. A future improvement to the SCHOLAR methodology will be the addition of template files. We are also working to integrate specific features such as titles, summaries, and keywords. These fields will hopefully allow us to develop software to automatically build a SCHOLAR tutorial index with the option of exporting the entire repository as a website.

The SCHOLAR lessons and examples will continue to be refined and shared as a git repository for any interested instructors. As the SCHOLAR type of lessons are adopted, it will be possible to share the resources across institutions building a robust framework and resource for learning.

References

[1] "Git." Accessed: Aug. 22, 2016. [Online]. Available: https://git-scm.com/

[2] S. Downes, "Models for Sustainable Open Educational Resources," *Interdisciplinary Journal of E-Learning and Learning Objects*, vol. 3, no. 1, pp. 29–44, Jan. 2007.

[3] T. Kluyver *et al.*, "Jupyter Notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds., IOS Press, 2016, pp. 87–90.

[4] "GitHub · Build software better, together." [Online]. Available: https://github.com/

[5] "GitLab.com | Open Source Git Management Software." [Online]. Available: https://www.gitlab.com/

[6] "bitbucket.com." [Online]. Available: https://www.gitlab.com/

[7] "CMSE / Data_Science_Bridge_Curriculum · GitLab," GitLab. Accessed: Feb. 07, 2024. [Online]. Available: https://gitlab.msu.edu/CMSE/data_science_bridge_curriculum

[8] "CMSE / DataTools_Tutorial_Demo · GitLab," GitLab. Accessed: Feb. 07, 2024. [Online]. Available: https://gitlab.msu.edu/CMSE/datatools_tutorial_demo

[9] "CMSE / Gen AI Tutorials · GitLab," GitLab. Accessed: Feb. 07, 2024. [Online]. Available: https://gitlab.msu.edu/CMSE/Gen_AI_Tutorials