Student Workers as Library Programmers: A Case Study in Automated Overlap Analysis

Matthew W. Goddard
California Baptist University, mgoddard@calbaptist.edu

Follow this and additional works at: <https://docs.lib.purdue.edu/charleston>

Part of the [Library and Information Science Commons](#)

An indexed, print copy of the Proceedings is also available for purchase at:

<http://www.thepress.purdue.edu/series/charleston>.

You may also be interested in the new series, Charleston Insights in Library, Archival, and Information Sciences. Find out more at: <http://www.thepress.purdue.edu/series/charleston-insights-library-archival-and-information-sciences>.

Matthew W. Goddard, "Student Workers as Library Programmers: A Case Study in Automated Overlap Analysis" (2014). *Proceedings of the Charleston Library Conference*.
<http://dx.doi.org/10.5703/1288284315640>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

Student Workers as Library Programmers: A Case Study in Automated Overlap Analysis

Matthew W. Goddard, California Baptist University

Abstract

This paper describes a single solution to two very different problems. The first problem is that undergraduate students who aspire to careers in programming or software development need real-world work experiences that are not always readily available. The second problem is that in considering whether to acquire large e-book packages, libraries need to be able to answer the question, "How many of these do we already have?" Currently, most ILSs do not include a built-in feature to address the need for this kind of overlap analysis. In order to develop a simple, low-impact technical solution to this second problem, the library at California Baptist University also helped to address the first. We hired one upper division Electrical and Computer Engineering student to create a method to easily assess the redundancy of titles between large e-book packages and current holdings. The objective of this paper is twofold: to advocate for increased high-level use of student workers enrolled in computer science or computer engineering programs, and to share one simple, affordable way for libraries to assess the feasibility of large e-book packages.

In the earliest days of American higher education, undergraduate students were scarcely allowed to use their university's libraries, let alone work in them. Even when hiring of academic library student workers picked up steam in the 1910s and 1920s, there was grumbling from some quarters about what was seen as a tilt away from professional standards (White, 1985). By now, however, the widespread dependence of nearly all academic libraries on student workers is a long-standing fact of life. But these undergraduate students, particularly those enrolled in upper division courses with content relevance to library operations, often have so much more to offer than libraries are willing to accept. These students are also acutely aware that they will be entering a job market where relevant work or internship experience is viewed as more important than academic achievement across "all industries and hiring levels" (Fischer, 2013). By entrusting these students with projects that create value for the library and provide meaningful experience to the student, libraries can create an ideal win-win scenario.

Literature Review

The typical responsibilities of student workers are commonly known and fairly standard across academic libraries. Shelving, circulation, and the

scanning component of digitization are representative of the relatively low level of responsibility entrusted to library student workers. Tasks requiring manual labor are prominent, and help desk duties are often limited to answering the most basic questions. Even as the historical trend has been toward giving student workers more responsibility, libraries remain reluctant to assign tasks requiring a high level of responsibility (Gruen & Wooden, 2011). There is a smattering of examples of more advanced projects in the library literature. In the late 1980s, Texas A&M hired students fluent in Chinese to assist in cataloging Chinese language materials (Gomez & LaGrange, 1990). At Virginia Commonwealth University in the 1990s, art students were hired to catalog art exhibition catalogs (Guidarelli & Cary, 1999). More recently, University of Michigan and Coastal Carolina University have implemented peer reference services, recruiting undergraduates to help other undergraduates with their research (MacAdam and Nichols; Faix et al.). Finally, many institutions with LIS programs offer graduate assistantships in a wide range of areas (Silver and Cunningham).

In all of these examples, students were recruited for the particular skills they brought to the project. In the first two examples, it was necessary to hire undergraduate student workers with particular skills because those skills were not

sufficiently represented by the library staff. The project discussed in this paper focuses on student workers with career aspirations in programming and/or web development. For small libraries without the budget capacity for hiring library developers or programmers, these students are a rich and underutilized resource.

The Project

The Annie Gabriel Library is the sole library serving California Baptist University (CBU), a mid-sized private university in Riverside, California. Like every other academic library, we employ many student workers without whom our operations would quickly spin into chaos. They process, shelve, circulate and repair books, they scan archival materials, and they keep our printers full of paper. They do much else, but the preponderance of these responsibilities are similarly low on the scale of complexity and responsibility.

With significantly more e-books in our collections than print books, the library depends on licensing large packages of e-books from publishers and aggregators, ranging in size from several hundred to tens of thousands. With the rapid growth of our e-book collection, and continued acquisitions of print books, it becomes increasingly important for us to effectively evaluate the redundancy between the many and varied e-book packages available to us and our current holdings. Only with this analysis can we accurately calculate important values like the cost per unique title. So we decided to take advantage of the expertise represented in the CBU School of Engineering's Electrical and Computer Engineering Bachelor of Science degree program by hiring one upper division undergraduate student to tackle this project.

The results of his efforts was a simple utility (dubbed "Osiris" after its creator) to quickly and easily compare any list of books to a library's current bibliographic holdings. It is a short, relatively simple 343-line Perl script that checks the status of any given CSV file of titles against our local ILS holdings using a catalog search service included in the standard Symphony Web Services module provided by our ILS vendor, SirsiDynix. While it was designed with the primary purpose of

evaluating e-book packages as described above, it can also be used to evaluate large donations of print materials, as well as verifying that subscribed e-book packages are fully represented in the local catalog. After describing some general guidelines for small libraries considering hiring student workers to apply their programming or development expertise to library projects, this paper will conclude with a brief description of the utility itself.

The first recommendation is that the main project should be relatively low priority. Learning on the job is a significant aspect of the student's experience, so projects that require quick turnaround on tight deadlines are not ideal. The student and supervising librarian may decide to establish project milestones in advance, with particular dates as deadlines, but these dates should be merely provisional. The student's experience, and the project itself, will benefit most when the student has ample time to tinker.

Second, it is good to have additional, smaller and simpler projects or tasks for the student to pursue, even if they have nothing to do with programming. This practice benefits both the student and the supervising librarian. For the student, these kinds of tasks may provide a welcome change of pace from the main project and allow the student to continue working even when progress on the main project is halted for whatever reason. For the librarian, they provide some of the benefits of a traditional student assistant in getting more things done. Some examples of peripheral tasks completed during our project include verifying activation of e-journal access, creating library signage, updating library tutorial videos, and providing manual collection analysis.

Third, the supervising librarian should frequently be available for questions. The student will learn by doing, by reading, but also by interacting with the supervising librarian. Many questions will naturally arise over the course of a project, so being available to answer questions will prevent misunderstandings and wasted time. We tried to schedule the student's hours to overlap with the librarian's hours as much as possible, and both

worked in the same office. Thus, it was an ideal environment for open communication.

Fourth, the supervising librarian should regularly check in on progress. The sharpest, most assertive students will ask all of the questions they need in order to effectively complete their project. However, either because they are shy or cocksure, some students will not. Regularly checking in on the student's progress not only provides accountability, but also provides an opportunity to communicate more deeply about the project. When everyone has a good understanding of how the project is proceeding, ideas can be shared about the best ways of proceeding. In our project, regular communication provided opportunities to discuss problems and combine our knowledge to identify the best solutions.

Finally, hiring students who are near the beginning of their senior year is a good idea for two reasons. First, these students will generally be the most experienced in the kind of programming and development they'll be asked to perform, even if only from their coursework. Secondly, relatively short-term employment ensures that in the unlikely event the arrangement ends up not being the win-win both parties expected, it will at least be only for a predetermined period of time. We have hired seniors for three consecutive years; each year, the graduating senior helps pick his successor. Using this method we have had great success in finding students who are sharp and committed.

The End Product

"Do we have this book?" is surely among the most basic questions that can be asked of a library, not to mention the most common. It is trivial for most people to answer this question for themselves. However, the task of automating this process in order to answer it for thousands of books at once reveals just how complex it really can be. Does "have" mean hold in print, or do e-book licenses count? Does "this book" mean this specific edition, or will any edition do? For the purposes of identifying overlap, we wanted to know if we held any edition, irrespective of format. This requirement limited the value of International Standard Book

Numbers (ISBNs), which are assigned to specific editions in specific formats (many newly published books will have five different ISBNs: for hardcover, softcover, and three different e-book formats (EPUB, MOBI and PDF). So the much simpler question, "Is this ISBN indexed in our catalog?" is not sufficient to answer our broader question, "Do we have this book?"

Other identifiers naturally suggested themselves as alternatives, foremost among them OCLC numbers; however, OCLC numbers are very rarely used by vendors, and thus are unlikely to be included in e-book package listings. So in spite of their shortcomings, ISBNs remained the best option for a unique identifier to look up, but further steps were needed to catch those cases where there is a match between titles, but not editions or format.

Thus, the script uses a two-step process. First, it searches for the ISBNs included in the source data file. Since many vendors include multiple ISBNs for each title, the script first looks up one, then the other only if the first does not find a match. If either ISBN is found in the local catalog, the script prints the result to the output file and moves on to the next title.

If neither ISBN is found, the script moves on to the second step, searching by title and author. Essentially, it performs a traditional known-item search, among the simplest tasks for a human operator. However, because the exact expression of titles and authors can vary in subtle but impactful ways, this step is more complicated than the first. We made an effort to find an optimal balance between two competing values: precision (i.e., accuracy—the proportion of matches that are made that accurately represent a match of the same title) and recall (i.e., comprehensiveness—the proportion of actual matches that are successfully identified as such).

The problem of precision arises in particular when the source data does not include author names, and/or when titles are extremely short or general. For example, if all we know about a title is that its title is Biology and its author is Smith, it is extremely likely to find results (there are many books written by a Smith with "biology" in its

title), regardless of whether any of those results are actually the title in question.

The various complications that had to be considered related to recall included:

- Source data that includes edition information in the title field, which is not indexed in the local catalog.
- Author names that include initials (such as those specified in APA style) rather than spelling the name out.
- Author names that include middle names or titles that are not indexed in the local catalog.
- Other minor differences of spelling or punctuation that may impact the search.

While some of these cases have no elegant solution using the tools and limitations of the project, our overall approach was to strike an optimal balance wherein titles that are held are most likely to be identified as such, without creating unnecessary cases of titles that are not held being identified as held. To that end, we decided to exclude subtitles of books, to exclude stop words, to remove certain special characters that may affect the search, and to remove volume information that may be included in the title field. We also decided to avoid complications in author names by only including the author's last name.

To maximize the script's flexibility and reduce the workload necessary to run it, the operator is first asked to provide a number of inputs. These inputs provide information on the formatting of the source data so that that data does not need to be extensively manipulated prior to running the script. Specifically, the operator is prompted to provide the name of the source file, the name of the output file, the separator value (typically comma or pipe), and which columns contain the title, author, and ISBNs. There is also a prompt asking whether the author's first and last names are combined in one column or separated into two, and if the former, the order of those names ("First Last" or "Last, First").

After the script completes, it outputs the overall number and percentage of unique titles, the runtime, and any errors that may indicate titles that should be checked manually. It also creates an output file in CSV format that lists details of each title's status, number of hits, and the item type of the first hit. While it's not always perfectly accurate, this information has already proven invaluable for analyzing the viability of prospective e-book collections.

Conclusion

The future of library technology will be developed by individuals currently enrolled in IT/CS programs at our institutions. Employing these students now exposes them to the value and relevance of libraries, as well as our unique needs and challenges. While a consideration of the differences between the computer science/information technology paradigm and the library/information science paradigm is well beyond the scope of this paper, it might be noted that librarians have an opportunity to augment the dominant computer science paradigm in which they're being inculcated with the library perspective that places a great emphasis on privacy, ethical behavior, and traditional sources of information.

At the same time, bringing a computer science or information technology student to work in the library introduces a valuable source of fresh and innovative thinking that may sometimes be lacking in academic libraries. The result of this project was so valuable to the library, and the experience of it so valuable to the student worker, that we expect to continue to employ one senior from the Electrical & Computer Engineering Program to pursue similar projects in the future. It is hoped that other academic libraries with staffing limitations might learn from this experience. By creating a flexible environment that fosters creativity, by giving the student time to pursue his or her own ideas, and by helping projects along with suggestions when they hit a wall, libraries can create a win-win scenario that will pay dividends for both student and library, far into the future.

References

- Faix, A. I., Bates, M. H., Hartman, L. A., Hughes, J. H., Schacher, C. N., Elliot, B. J., Woods, A. D. (2010). Peer reference redefined: New uses for undergraduate students. *Reference Services Review*, 38(1), 90-107.
- Fischer, K. (March, 2013). The employment mismatch. *Chronicle of Higher Education*. Retrieved from <http://chronicle.com/article/The-Employment-Mismatch/137625>
- Gomez, J. & LaGrange, J. (1990). A Chinese challenge: Utilizing students for special cataloging projects. *Cataloging & Classification Quarterly*. 12(1), 87-102.
- Gruen, C. M. & Wooden, A. M. (2011). Student assistants 2.0: Utilizing your student assistant's capabilities. In Baudino, F. (Ed.), *Brick and click libraries: An academic library symposium, Northwest Missouri State University, Friday, November 4, 2011* (pp. 18-26). Maryville, MO: Northwest Missouri State University. Retrieved from <http://files.eric.ed.gov/fulltext/ED526899.pdf>
- Guidarelli, N. & Cary, K. (1999). Untapped resource: Art students cataloging art exhibition catalogs at Virginia Commonwealth University." *Cataloging & Classification Quarterly*, 26(4), 63-75. http://dx.doi.org/10.1300/J104v26n04_05
- MacAdam, B. & Nichols, D. P. (1989). Peer information counseling: An academic library program for minority students. *Journal of Academic Librarianship*, 15(4), 204-209.
- Silver, S. L., & Cunningham, V. P. (2008). *The impact of the USF Tampa Library graduate assistant program on career and professional development*. Retrieved from ERIC database. (ED500319).
- White, E. C. (1985). Student assistants in academic libraries: From reluctance to reliance. *The Journal of Academic Librarianship*, 11(2), 93-97.