Ticker: The Academic Business Librarianship Review, 6:2 (2021)
https://doi.org/10.3998/ticker.1944

Research Support through Programming: Developing an Automation-Assisted Literature Search

KEVIN A THOMAS
University of Pennsylvania, Philadelphia, PA
kevinat@wharton.upenn.edu

MARCELLA E. BARNHART
University of Pennsylvania, Philadelphia, PA
bmarcell@wharton.upenn.edu

Abstract This case study describes the work of librarians at the Lippincott Library of the Wharton School at the University of Pennsylvania in developing a novel approach to supporting research through programming. Approached by researchers for assistance with a large-scale literature search that also involved text extraction, we utilized both traditional bibliographic indexes (Web of Science) and citation management tools (EndNote) and less-traditional tools like the programming language Python and a PDF extraction program (LA-PDFText) to approach different sections of the project. This article outlines that process and briefly discusses the potential for developing new services in business libraries around programmatic research support.

Keywords *data services, research programming, electronic data processing, case studies*

Working with data has become an integral part of the business library environment, so much so that a recent double issue of the *Journal of Business and Finance Librarianship* focused exclusively on business and financial data (Bordelon, 2020). This holds true at the Lippincott Library of the Wharton School, the business library at the University of Pennsylvania. We serve researchers across all business disciplines but have been data-focused for many years due to the emphasis on financial research from our faculty and students. However, over the past few years, there has been an uptick in requests for assistance with data projects outside of those focused on financial, company or market information. These fell into categories that Pinfield, Cox and Rutter (2017) described as "datafied scholarship," using machine-assisted techniques to work with complex data sets. This article provides a case study of one of these projects, detailing how specific programming skills and software tools can be used to creatively support non-traditional data capture and collection early in the research lifecycle.

Maxwell, Norton and Hu (2018) argued that libraries are missing critical strategic opportunities by framing the work that is being done around data in more traditional terms of archiving and access. In their view, librarians should expand into direct, active work with data—data analysis—rather than focusing solely on data curation as the future of data services. Cox et al. (2019) developed a "maturity model" of research data services in academic libraries based on survey data from international institutions. Their model delineates four levels of commitment to such services, from none (Level 0) to transformation (Level 3). At the transformative level, the librarians providing data services have acquired new skills related to data analysis to support these services and organizational structures may change to acknowledge new work patterns and relationships, such as embedding librarians within research teams. Based on the results of their survey, they concluded that transformation is not yet occurring at a large scale in academic libraries. While the project that we describe does not rise to the level of transformation, we believe that it does illustrate a high level of data analysis support and collaboration.

Process

In the fall of 2019, a Wharton faculty member approached the Lippincott Library for assistance with a large-scale literature search. Initially, the research team framed the project as a systematic review, but it became clear upon more discussion that they were not interested in evidence synthesis. Instead, the research team wanted to identify all experimental designs used in researching their topic of inquiry. Their ultimate goal was to replicate and extend the tasks employed in prior research. As we explored options for capturing their topic using traditional literature searching techniques, it became clear that those would be overly broad and would involve winnowing out many false drops from the retrieved set. For example, a Boolean search intended to capture the desired breadth of content led us to build a preliminary Scopus query that identified roughly two million papers; this was far more material than the research team was prepared to review and did not even cover all relevant timeframes and databases. In its place, we developed an iterative process that leverages both manual and automated steps to incrementally collect and prune a selection of relevant papers, delivering the required text content to the research team.

Search & Download

The research team provided a set of 54 seed papers from which to begin our search process. They had discovered these papers independently and considered them exemplary of the type of content they hoped to identify. We used the Web of Science (WoS) to identify bibliographic records for these papers, based largely on our intention to use the WoS features of saving and sharing information via EndNote Web to work with the research team. Sharing records would allow them to explore different ways of identifying other works using cited and citing record options, as well as by using the Related Records feature. Because of the many variations in terminology related to their research area, using citations to stand in for concepts would be useful in identifying additional articles. In addition, the integration between WoS and classic EndNote (stand-alone software version) allowed us to easily import WoS records into EndNote, and from there we used the automated Find Full Text feature in EndNote to download the full-text PDFs.

Extract & Review

Extracting files requires pointing software to each file according to its unique file path and file name. Because EndNote saves all downloaded PDFs within the related EndNote Library's directory, we knew the beginning of the file path. From that point, things become more complicated. EndNote nests each file within subdirectories. It also gives each file a unique file name. The files we downloaded during development did not follow an entirely regular pattern (e.g.: ~/001/001.PDF; ~/002/002.PDF; …; ~/nnn/nnn.PDF). Instead, they appeared to reflect a naming or storage structure internal to WoS that was not readily predictable. We therefore developed a small Python script that uses the os.walk method to locate each PDF stored at any level within a specified directory and return a list of each file path and file name. This process, shared on the Lippincott Library blog (Thomas, 2020), readily generalizes to research applications that use other file structures and file types.

```
import os
       def list_files(filepath, filetype):
       paths = []
       for root, dirs, files in os.walk(filepath):
             for file in files:
                    if file.lower().endswith(filetype.lower()):
                          paths.append(os.path.join(root, file))
                              return(paths)
```

Figure 1. Code snippet for finding files

With full-text PDFs downloaded and located, extracting the entire text of each paper could have taken just a few lines of code. Our researchers, however, wanted to isolate content about experimental designs, generally found under the heading Methods or a similar label. They also wanted to exclude content outside the relevant section because the additional content could have increased review time or misdirected a human or algorithmic reviewer about what approach the research in that paper implemented. Our script addresses these requirements by feeding each document to an executable program named LA-PDFText, referring to its layout-aware (LA) text extraction (Ramakrishnan et al., 2012); see Bast & Korzen (2017) for a comparison of PDF extraction tools.

PDFs can contain metadata, but they do not reliably contain metadata that explicitly identifies text content by section. In response to this circumstance, LA-PDFText examines both a paper's text (e.g.: the word "Methods") and elements of its layout, such as type size and column width, to estimate what text falls within what section. Users can configure the identification parameters (e.g., associate 12-point type with body text) and select the output (e.g., only text identified as within the Methods section). According to Ramakrishnan et al. (2012), section identification becomes more reliable when tailored to a single publication with consistent formatting. The authors report an average of 89% recall, measured as True Positives divided by the sum of True Positives and False Negatives, in testing. Working with assorted publications and identifying only the Methods, we optimized multiple configurations so that the researchers could choose their preferred balance of precision and recall. Our range of settings achieved average recall rates of about 67% to 87% within our sample of 54 seed papers. Because the research team intended to review potentially tens of thousands of articles, even a review time reduction for 67% of all papers suggested a substantial potential labor savings.

Future Extensions

LA-PDFText saves extracted text in the form of a text file, a straightforward document format that many software applications can read. Although the research team for this project chose to take responsibility for screening papers and extracting concepts from the Methods text, the text output lends itself to further programmatic filtering and analysis that could leverage various techniques. These present additional library service opportunities.

For example, we began developing Boolean screening filters for the project described in this article. Our initial filter attempted to catch files for which LA-PDFText failed to identify a Methods section. It compared each file's character-length to a specified threshold, listing suspiciously short files for manual review. We also explored applying text pre-processing procedures, such as standardizing case and lemmatization, and developed logic that would check for given words, potentially terms that the researchers thought should disqualify a paper from further consideration. In programmatically loading each text file into Python and comparing it to Boolean criteria, developers have a straightforward yet flexible tool for applying rules-based document filtering.

Researchers working in the systematic review space have taken other automated approaches to facilitate literature review and selection. Shakeel et al. (2020) documented several techniques, grouping them into four primary categories: visual text mining (VTM), text classification, information retrieval, and citation links. VTM breaks papers into constituent words and creates a graphical display depicting the relative similarity of the papers' language or, in some cases, citation networks. Text classification techniques employ an initial set of manually-selected articles to train an automated model to include or exclude papers based on similarities in terminology. Information retrieval draws on the text of an initial paper and query to automate the process of expanding the user's search terms and syntax, scoring results by relevance. The final technique gathers citation links from a provided set of seed papers, iteratively retrieving the cited papers and gathering their citation links; limited user-assessment helps model a scoring system for weighting the resulting papers' relevance.

Researchers who want to draw out themes from a set of papers they have selected might explore the text mining technique called topic modeling. Topic modeling forms topics by grouping co-occurring terms from across all documents. Someone familiar with the subject matter may then associate these groups of terms with conceptual topic labels. Topics relate back to each paper by measuring the degree to which each paper relates to each topic; see Barde & Bainwad (2017) for an overview of topic modeling techniques and tools.

Conclusion

The importance of keeping detailed documentation on a project like this cannot be overstated. Wang (2013) notes that data projects often involve a longer service lifecycle as researchers revisit and revise their approaches, also requiring different types of communication throughout the process; we certainly found this to be true. While the process outlined above smooths the rough edges for simplicity and coherence, there were many fits and starts. The researchers would disappear for several months because of competing priorities. They would re-emerge not quite remembering where we had left off, and each time we restarted the project, we spent considerable time re-educating the team about why certain decisions had been made. Keeping meeting notes, categorized email threads, versioned code files and procedure outlines, and shared working folders helped the library team remain organized and consistent. While the process was frustrating at times, it encouraged us to develop and maintain documentation that we have adapted to other projects with related but distinct needs.

Ultimately, the Lippincott Library knit together existing tools and custom programming into a documented literature search and text extraction process that scales with researchers' requirements. Our process leverages Web of Science and EndNote for exploration and full-text downloads, employs Python and LA-PDFText for section-specific text extraction, and offers ready integration with additional text analysis and literature review automation techniques. Although we developed it for and adapted it to one research team's needs, the approach remains flexible enough to use with projects that employ different selection criteria or require other document text sections. The scenario also highlights the potential value of supporting faculty research through a combination of more-traditional librarianship tools and novel—even custom-programmed—technological solutions.

Ticker: The Academic Business Librarianship Review, 6:2 (2021)
https://doi.org/10.3998/ticker.1944

References

Barde, B.V. and Bainwad, A.M. (2017). An overview of topic modeling methods and tools. *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, 745-750 https://doi.org/10.1109/ICCONS.2017.8250563

Bast, H. & Korzen, C. (2017). A benchmark and evaluation for text extraction from PDF. *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 99-108. https://doi.org/10.1109/JCDL.2017.7991564

Bordelon, B. (Ed.). (2020). Special issue on business and financial data [Special issue]. *Journal of Business and Finance Librarianship, 25*(3-4), 103-378.

Cox, A.M., Kennan, M.A., Lyon, L., Pinfield, S. & Sbaffi, L. (2019). Maturing research data services and transformation in academic libraries. *Journal of Documentation*, *75*(6), 1432-1462. https://doi.org/10.1108/JD-12-2018-0211

Maxwell, D., Norton, H. & Wu, J. (2018). The data science opportunity: Crafting a holistic strategy. *Journal of Library Administration, 58*(2), 111-127. https://doi.org/10.1080/01930826.2017.1412704

Pinfield, S., Cox, A.M. & Rutter, S. (2017). *Mapping the future of academic libraries: A report for SCONUL*. https://sconul.ac.uk/sites/default/files/documents/SCONUL%20Report%20Mapping%20the%20Future%20of%20Academic%20Libraries.pdf

Ramakrishnan, C., Patnia, A., Hovy, E., & Burns, G. (2012). Layout-aware text extraction from full-text PDF of scientific articles. *Source Code for Biology and Medicine*, *7*, Article 7. https://doi.org/10.1186/1751-0473-7-7

Shakeel, Y., Krüger, J., Von Nostitz-Wallwitz, I., Saake, G., & Leich, T. (2020). Automated selection and quality assessment of primary studies: A systematic literature review. *ACM Journal of Data and Information Quality*, *12*(1), Article 4. https://doi.org/10.1145/3356901

Thomas, K. (2020, February 5). Python pointer: Find files with os.walk(). *Datapoints: A blog from the Lippincott Library of the Wharton School of Business*. https://lippincottlibrary.wordpress.com/2020/02/05/python-find-files-with-os-walk/

Wang, M. (2013). Supporting the research process through expanded library data services. *Program: electronic library and information services*, *47*(3), 282-303. https://doi.org/10.1108/PROG-04-2012-0010