Issues in Science and Technology Librarianship

Creating a Scholarly API Cookbook: Supporting Library Users with Programmatic Access to Information

Vincent F. Scalfani
Chemistry and Informatics Librarian
University Libraries
The University of Alabama
Tuscaloosa, AL
vfscalfani@ua.edu

Kevin W. Walker
Director of Branch Library & Research Data Services
University Libraries
The University of Alabama
Tuscaloosa, AL
kwwalker@ua.edu

Lance Simpson
Assessment Librarian
University Libraries
The University of Alabama
Tuscaloosa, AL
lsimpson@ua.edu

Avery M. Fernandez
Data and Informatics Student Assistant
University Libraries
The University of Alabama
Tuscaloosa, AL
amfernandez7@crimson.ua.edu

Vishank D. Patel
Data and Informatics Student Assistant (former)
University Libraries
The University of Alabama
Tuscaloosa, AL
vdpatel@crimson.ua.edu

Anastasia Ramig
Data and Informatics Student Assistant (former)
University Libraries
The University of Alabama
Tuscaloosa, AL
aramig@crimson.ua.edu

Cyrus Gomes
Data and Informatics Student Assistant
University Libraries
The University of Alabama
Tuscaloosa, AL
chgomes@crimson.ua.edu

Michael T. Moen
Data and Informatics Student Assistant
University Libraries
The University of Alabama
Tuscaloosa, AL
mtmoen@crimson.ua.edu

Adam M. Nguyen
Data and Informatics Student Assistant
University Libraries
The University of Alabama
Tuscaloosa, AL
amnguyen4@crimson.ua.edu

Abstract

Scholarly web-based application programming interfaces (APIs) allow users to interact with information and data programmatically. Interacting with information programmatically allows users to create advanced information query workflows and quickly access machine-readable data for downstream computations. With the growing availability of scholarly APIs from open and commercial library databases, supporting access to information via an API has become a key support area for research data services in libraries. This article describes our efforts with supporting API access through the development of an online Scholarly API Cookbook. The Cookbook contains code recipes (i.e., tutorials) for getting started with 10 different scholarly APIs, including for example, Scopus, World Bank, and PubMed. API tutorials are available in Python, Bash, Matlab, and Mathematica. A tutorial for interacting with library catalog data programmatically via Z39.50 is also included, as traditional library catalog metadata is rarely available via an API. In addition to describing the Scholarly API Cookbook content, we discuss our experiences building a student research data services programming team, challenges we encountered, and ideas to improve the Cookbook. The University of Alabama Libraries Scholarly API Cookbook is freely available and hosted on GitHub. All code within the API Cookbook is licensed with the permissive MIT license, and as a result, users are free to reuse and adapt the code in their teaching and research.

Keywords: Application Programming Interface (API), Programmatic access, Z39.50, Machine-readable, Tutorial, Jupyter

Recommended Citation:

Scalfani, V. F., Walker, K. W., Simpson, L., Fernandez, A. M., Patel, V. D., Ramig, A., Gomes, C., Moen, M. T., & Nguyen, A. M. (2023). Creating a scholarly API cookbook: Supporting library users with programmatic access to information. *Issues in Science and Technology Librarianship, 104*. https://doi.org/10.29173/istl2766

Introduction

Researchers are increasingly in need of machine-readable data; that is, data in a readily accessible format for downstream and custom analysis. Access to machine-readable data facilitates a broad range of academic research, including, for example, bibliometric citation studies, text mining, and computational material property prediction. While some machine-readable data are available as bulk downloads (e.g., supplementary datasets), the ability to programmatically search for and harvest machine-readable data allows researchers greater flexibility in compiling custom datasets, streamlining computational workflows, and increasing the reproducibility of research.

Machine-readable data and programmatic searching are available from numerous open and subscription-based scholarly databases and are often accessible via a web-based Application Programming Interface (API) (MIT Libraries, 2023). With a web-based API, users generally write custom programmatic scripts to interact with the database, in place of interacting with the database through a traditional graphical web user

interface. Supporting the acquisition of machine-readable data is a growing opportunity for research libraries, as well as an area of growing interest within the context of Research Data Services (RDS) (Tenopir et al., 2015, 2016).

While the development of service support for API-based data harvesting can provide the means for a library's expansion further into RDS, scant availability of documentation and training represents a significant barrier. What documentation exists is typically written for experienced developers. Use cases and starter scripts are rarely included in documentation, which can make it difficult for novice researchers to get started. This article describes our exploratory efforts and experiences with navigating this hurtle to help promote the use of scholarly APIs across one academic library system's user base.

To address the existing support vacuum for novice API users, we developed and published an open-source Scholarly API Cookbook that contains use-cases and programming scripts for interacting with ten different Scholarly APIs. Content is presented in the form of an online Jupyter Book, and the majority of tutorials are available in four different programming languages including: Python, Bash shell, Matlab, and Mathematica. The Cookbook serves as an open resource for researchers, but also as an internal tool supporting teaching and research consultations related to data retrieval via programmatic means.

Literature Review

There are several reports in the literature of librarianship identifying services under the umbrella of RDS. Tenopir et al. surveyed North American libraries and identified specific areas of RDS that libraries are currently supporting, planning to support, or have left unaddressed. Some of the areas of support identified included consulting on data management plans, creating web guides for data resources, collaborating with researchers on a project, and providing support for finding datasets. It was found that the provision of reference support for finding/citing data was the most widely supported service across those institutions surveyed (Tenopir et al., 2015). This finding aligns with the recently published *Medical Library Association Data Services Competencies*, which defines the data literate librarian as one who "finds, interprets, and manages data according to ethical principles" (Federer et al., 2020, p. 305). Bearing this in mind, support for finding datasets, and more broadly, helping researchers acquire and compile their own datasets, comprises a key area of library support within the realm of RDS.

Yoon and Donaldson surveyed approximately 200 academic and public libraries from the American Library Directory, regarding their support of data curation services. Respondents most often indicated that they supported services such as data management planning (14%), as well as data organization and description (13%) (Yoon & Donaldson, 2019). However, 10% of responses also indicated that they supported data acquisition as part of their services (Yoon & Donaldson, 2019). Swygart-Hobaugh et al. (2022) completed a content analysis of about 150 recent dissertations from Georgia State University with the goal of identifying areas of data services in need. Interestingly, the use of primary data was high for the physical sciences and mathematics (86%),

education (77%), and humanities fields (75%). The use of secondary existing datasets was most often seen in the health (50%) and social sciences (59%). These results led Swygart-Hobaugh et al. to target their secondary data service efforts toward specific disciplines where use is high, and prioritize RDS data topics, including "survey design and administration, use of data collection tools such as the Qualtrics survey platform, qualitative interview methodologies, and web scraping and other primary data collection methods" (Swygart-Hobaugh et al., 2022, p. 900), for the fields mostly engaged in these types of research activities.

While the aforementioned research highlights growing RDS support for library users who are engaged in finding and acquiring data, this area of the literature remains underdeveloped. For example, there have been few, if any, published articles discussing how scholarly APIs are being deployed within academic libraries in support of user discovery and acquisition of research data. With that said, several academic libraries have compiled research guides related to APIs, including MIT Libraries, Boston College Libraries, UCLA Library, and UC San Diego (Boston College Libraries, 2023; MIT Libraries, 2023; UC San Diego Library, 2023; UCLA Library, 2023). A more limited number of libraries have begun to offer workshops and teaching materials related to scholarly APIs. For example, the University of Virginia offered a workshop in 2019 on the U.S. Census Data API using R; the University of Texas Libraries offered an introductory workshop in 2020 on APIs; and, more recently, the Boston College Libraries Digital Scholarship Group published an online module for how to interact with the Federal Election Commission API in Google Sheets (*BCDS | DS Learn: REST API, 2022*; Goodale, 2020; Huck, 2019).

Reports discussing instruction and tutorials related to general web and scholarly APIs are available outside the library science literature, spanning disciplines such as computer science, business, data science, chemistry, and bioinformatics. For example, web APIs were taught in the computer science curriculum at Harding University (McCown, 2010). As part of an undergraduate course focused on web information retrieval, students were tasked with using the Yahoo web API to build a search engine (McCown, 2010).

There have been reports of web API curricula being used within the business disciplines. Chen et. al (2009) reported their experiences with introducing open Web APIs into a graduate course in management information systems (MIS) at the University of Arizona. Here, students used company web APIs, including those made available through Amazon, eBay, and Google, to collect data and create a web-mining application. Overall, it was reported that students had a positive experience using the web APIs and, on a scale of 1 (extremely difficult) to 5 (extremely easy), all surveyed students ranked the use of web APIs as a 3 or 4 (Chen et al., 2009). Another business discipline example was reported by Olsen and Moser (2013); they taught web APIs at both the undergraduate and graduate levels at Arizona State University within the Department of Information Systems. Students first experimented with web APIs, such as Yelp and Twilio (communications tool) in a web browser, before branching into more advanced API interactions with programming languages or third-party utilities (e.g., Google Sheets). Through offering this course, Olsen and Moser (2013) found that teaching web APIs created student excitement. Further, they argue that developing an

understanding of web APIs adds valuable skills for both programmers and non-programmers—such as managers (Olsen & Moser, 2013).

Fergusson and Wild (2021) reported on their teaching strategies and experiences introducing web APIs to high school data science students. For example, they recommend that students first explore a website with both the standard graphical interface and manual modification of the URL (e.g., modify an identifier in the URL and see what the website returns). Following this preliminary exploration, students can then be introduced to the API as a platform for data retrieval and manipulation via some type of data science task, such as data visualization or transformation. APIs used in their teaching included: Pixabay (image sharing website), Star Wars API, YouTube Data API, and the Spotify API. Expanding upon this work, Fergusson and Pfannkuch (2022) reported on their use of the OMDb API (movie rating data) in a workshop with high school statistics teachers. The teachers used a custom web-based tool, which supported both graphical and code-based interaction with the API, to gather data and build predictive models (Fergusson & Pfannkuch, 2022).

Several reports of teaching, or creating web API tutorials, are reported in the chemical science literature. TeachOpenCADD is an online interactive collection of Python Jupyter Notebooks for learning computer-aided drug design. The collection includes a variety of tutorials which demonstrate how to acquire data programmatically from the ChEMBL, Protein Data Bank, KLIFS (Kinase database), and PubChem APIs (Sydow et al., 2019). Similarly, the Cheminformatics Online Chemistry Course, an open educational resource (OER) provided through LibreTexts, includes tutorials and student projects that focus on acquiring data through the PubChem API using Python, R, and Mathematica (Kim et al., 2021). Finally, Scalfani et al. (2020, 2021) described their experiences creating tutorials focused on (1) using Matlab to acquire data from the PubChem API, and (2) using the NCBI EDirect software to acquire data from PubChem and PubMed within a Linux terminal.

Methods

The purpose of this article is to provide details surrounding the development and deployment of an instructional tool for those learning to interact with, and harvest data from, scholarly APIs. While a case study methodology frames the construction of the narrative details of this work, additional technical details regarding the creation of the Scholarly API Cookbook (henceforth referred to simply as "the Cookbook") are provided. The word Cookbook, in relation to coding or programming, refers to a collection of code snippets that provide solutions to common programming problems—essentially, examples of code one might find useful in specific scenarios. In the case of the Cookbook described herein, one finds a collection of starter scripts and documentation for effectively interacting with a variety of scholarly APIs. It is hoped that through providing details of this work, others will discover a viable template for reproducing and expanding this work.

Operating System and General Software Environments

Ubuntu Linux 18.04 or later was used for all programming environments and software including API tutorial programming for Python, Mathematica, and Bash. Matlab code was initially developed on Windows 10, but then tested on Ubuntu. For the Python tutorials, a conda Python 3 environment was used. Where possible, the specific dependencies (e.g., curl for Bash tutorials) and versions of software used (e.g., Matlab R2022b), along with the date that the code was tested, are noted on the individual Cookbook tutorials.

Jupyter Book Document Formatting

Python and Mathematica tutorials were included in the Cookbook as Jupyter Notebooks. The Wolfram language kernel for Jupyter notebooks (Wolfram Research, n.d.) was used for the Mathematica notebooks. Matlab and Unix Shell Bash tutorials were manually typeset in the Python reStructuredText markup syntax for inclusion into the Cookbook.

Jupyter Book Building and Hosting

For local testing, a Conda environment was used with Jupyter Book (Executable Books Community, 2020). A typical environment recipe was as follows:

```
$ conda create --name book-env
$ conda activate book-env
$ conda install -c conda-forge jupyterlab matplotlib pandas sphinx pip
$ pip install -U jupyter-book
```

Specific versions used were Python (3.10.5), Jupyterlab (3.4.4), matplotlib (3.5.2), pandas (1.4.3), sphinx (4.5.0), pip (22.2.1), and jupyter-book (0.13.0). The jupyter-book build command was used to generate the HTML files. For hosting the book on GitHub, a GitHub workflow was created that builds and hosts the HTML files in a github-pages environment. This action is automated and new HTML files are generated when any edits or new tutorial files are committed to the main branch of the Cookbook GitHub repository.

Programming Team and API Tutorial Design

The programming team, referred to internally as the Data & Informatics Team, consisted of one librarian lead (VFS) and several undergraduate students with programming experience (AMF, VDP, AR, CG, MTM, and AMN). Students were hired as Data and Informatics Student Assistants, and part of their required goals was to contribute to the Scholarly API Cookbook. All new student team members received a general introduction to working with APIs responsibly, including, for example, respecting any time limits between API calls, data reuse considerations, and licensing restrictions.

The general workflow for creating a scholarly API tutorial was to first select a scholarly API and brainstorm some practical use-cases. These use-cases came from our own

interests, as well as inferred from reference requests over the past several years (e.g., bibliographic or property data requests). Once several use-cases were established, a student team member was assigned to create the scholarly API code in whatever programming language with which they were most comfortable. Team members were also encouraged to contribute their own use-cases as they experimented with the API; that is, our ideas provided the seed that were then often developed into something more substantial. One of the authors (VFS) met weekly with student team members for help, code-reviews, and feedback. When a scholarly API tutorial was completed, it was added to the Cookbook, and then ported (i.e., replicated) to another programming language by either the same student or another member of the team.

Results

The University of Alabama Scholarly API Cookbook is available on GitHub as a Jupyter Book (https://ualibweb.github.io/UALIB_ScholarlyAPI_Cookbook). The Cookbook is organized by API type and programming language (**Figure 1**). Users can reuse code from individual tutorials by copying code cell blocks or by downloading the complete source code from the associated GitHub repository.

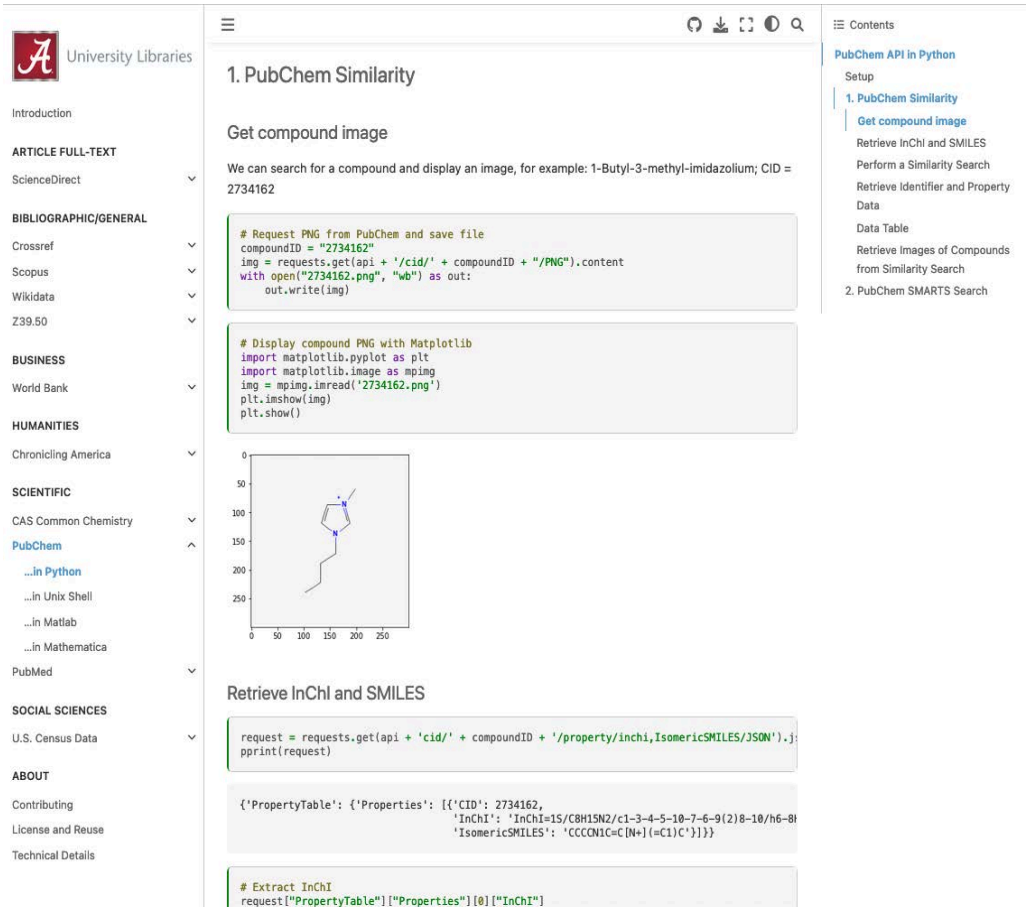

Figure 1. Screenshot of the Python PubChem API tutorial in the Cookbook.

At the time of this writing, the Cookbook contains tutorials for 10 different scholarly APIs spanning use-cases across several academic disciplines, including ScienceDirect (full-text), Crossref, Scopus, Wikidata, World Bank, Chronicling America, CAS

Common Chemistry, PubChem, PubMed, and U.S. Census Data. Each tutorial is available in Python, Bash, Matlab, and Mathematica, except Wikidata, which is currently only available in Mathematica. In total, the online Cookbook contains 37 open-access scholarly API tutorials. The contents of an individual tutorial generally include the tutorial author name(s), date the code was last tested, links and policies about the API, a few basic API calls, as well as 2-4 longer use cases. For example, the PubChem API demonstrates how to perform a programmatic chemical compound substructure search and retrieve compound images, while the U.S. Census Data API includes a use case to collect population data and plot the change in population over time across U.S. states.

The Cookbook is licensed with an open-source MIT license, which allows permissive reuse of the code and tutorials. Users can incorporate the code in their teaching and research. They can also contribute to the Cookbook on GitHub by submitting a pull request to fix bugs, add a new scholarly API recipe, or even suggest a new use case for us to add through the GitHub Issues tracker.

Discussion

Team Workflow and Challenges

Building a team of student programmers to tackle the development of code and tutorials for the Cookbook was instrumental to the successful launch of the Cookbook in a reasonable amount of time (i.e., about an academic year). While our broader RDS team includes at least four librarians with the requisite coding skills needed to contribute to a project such as this, other professional duties often hamper extensive, dedicated involvement in the actual coding process. Obligations such as meetings, teaching, and research make it challenging for a librarian to carve out the contiguous blocks of time needed to develop a resource such as the Cookbook in a timely manner.

As progress was made, helpful patterns began to emerge among our team. We found that assigning more experienced student team members to create new scholarly API tutorials, while assigning less experienced students to port existing scholarly API guides, was a successful strategy. Furthermore, while there was a significant time investment for our librarian project lead (VFS) on the front end—introducing students to APIs, helping setup their development environments, and providing feedback on code—students quickly became independent after completing one or two tutorials, delivering final draft tutorials that needed only minor edits before being added to the Cookbook. Here it is also worth noting that we found it helpful to introduce schedule overlap throughout the week whenever possible, so that team members could assist each other with code development.

Several challenges presented themselves as we began developing our API tutorials. The variety of APIs, and their related disciplinary fields, provided a significant amount of general content-related concepts with which student coders needed to familiarize themselves. In addition, inconsistencies in documentation, or the general lack thereof, prompted additional research and testing to complete the tutorials. Interestingly, these challenges led team members to consider the needs of the users who would eventually

use these coded recipes as a guide for learning how to use each API. This led to a deeper understanding of how the structure and clarity of a tutorial's code might impact the ability of others to learn from or deploy it.

Finally, while each new team member received instruction on the responsible use of APIs, licensing considerations, and data sharing, we did not explicitly frame these discussions in the context of data ethics principles (Data Ethics Framework Development Team, n.d.). In the future, we plan to expand the team member training with specific discussions regarding data ethics. This will be particularly important as team members engage in collaborative and sometimes confidential related programmatic services with faculty and students (see below).

Related Services

As the Data and Informatics Team grew, opportunities to expand the team's work beyond the development of the Cookbook began to emerge. This was a welcome development, as we initially had concerns that focusing on a single project (i.e., the Cookbook) for an extended period might prove too monotonous to elicit long-term enthusiasm and high-quality work. So, we began to branch out into other RDS activities, such as developing custom scripts for student/faculty researchers, teaching coding workshops, providing walk-in support for library users with coding questions, and exploring internal data-intensive projects (e.g., collections visualizations, programmatic cataloging workflows, etc.). At this point it became helpful to regularly engage team members in meaningful conversations about both their ongoing progress toward meeting project milestones, as well as the ways in which these projects were serving their own personal growth and edification. Importantly, it was in conversing with team members about their personal growth that helped reveal ways in which RDS projects might be tailored to the individual interests of team members.

The original intention was for the Cookbook to be a self-paced resource for users to independently learn about scholarly APIs. However, in developing this resource, our program has realized other key goals tied to the provision of both instruction and reference support for data-intensive research. For example, we have adapted content directly from the Scopus, PubChem, and the U.S. Census API Cookbook tutorials into our library instruction (UA Libraries Workshops, 2023). We have also used the Cookbook during research consultations as a reference tool for API use cases.

These types of research consultations have led to multiple collaborative pilot projects where our team members write and deliver custom API starter code to support data retrieval and analysis for researchers on our campus. With that said, the development of this service prompted us to develop the following criteria that govern these projects and the resulting code:

- If the project requires that we collect data via API and/or help with data analysis, we are collaborators on the project. Librarians/students that have contributed to the project are listed as co-authors on any resulting scholarship.

- If the project requires only the creation of a script, it is delivered with an MIT license attached and a request that the RDS team's contributions are acknowledged in any resulting scholarship.

This approach has a couple of benefits, particularly in cases where we deliver an MIT-licensed script and researchers collect their own data. The first is that we do not have to get involved with specific scholarly API data reuse conditions; that is, researchers are responsible for appropriate reuse of the data. Nevertheless, we still endeavor to follow the API scripting policies (e.g., time delays, data limits, etc.) and share the specific API reuse/licensing conditions with the researcher. The second benefit is that our MIT-licensed scripts give us the ability to adapt and re-use code with other researchers and/or to create additional Cookbook recipes. It is worth noting that while this informal policy has proven effective during our pilot phase, we are exploring a more formal memorandum of understanding (MOU) to govern these agreements.

Initial Impact

The UA Libraries Scholarly API Cookbook version one was completed during the fall 2022 academic semester. As the Cookbook is an open source project, some tutorials were published in a partially completed Cookbook prior to fall 2022, so for the purposes of assessing impact, we estimate the Cookbook to be active for about 18 months. During this time, we have adapted content from the Cookbook in five library scholarly API workshops and have delivered nine custom scholarly API related scripts for faculty or student research projects. The Scholarly API Cookbook GitHub repository currently has 5 stars (excluding our own stars); GitHub stars allow users to easily follow the repository, and as such, GitHub stars can be considered an altmetric. A Google search for "'Scholarly API Cookbook' site: .edu" turns up numerous hits of links to the University of Alabama Libraries Cookbook on LibGuides including, for example, George Mason University Libraries, UC San Diego Libraries, and University of Minnesota Libraries. In the near future, we are planning to assess current users of the Cookbook and seek feedback, which should help us improve the Cookbook and prioritize the creation of new API tutorial content.

Development Environment Experiences

The majority of programming work from team members was completed in an Ubuntu Linux environment. While the actual tutorials and programming languages are generally platform independent, we selected Linux as the main operating system used within the Team, in part because of preference, but also because of the open source nature of Linux and educational opportunity of learning a new operating system. Moreover, we have found installing software on Linux within a terminal and using a package manager to be easier and faster compared to traditional manual methods of installing software on Windows and Mac. While our internal library technology team performs the initial setup and deployment of the Linux workstations for the Data and Informatics Team, individual team members are largely responsible for installing and maintaining their custom development environments. Team members have commented that this is beneficial for their future careers, as familiarity with Linux is a useful skill in computer science work.

Jupyter Book Experiences

The Jupyter Book software proved to be an excellent platform for building and delivering the Scholarly API Cookbook. Building the book from a compilation of Jupyter Notebooks and reStructuredText markup tutorials was straightforward. The HTML generated supports the creation of an attractive and professional-looking web publication (Figure 1). Here it is worth mentioning the pros and cons of using a Jupyter Notebook, versus a markup document like reStructuredText, as a platform for delivering code tutorials. First, there is less effort involved in deploying Jupyter Notebooks for web-based instructional content, as no additional formatting of the tutorial is necessary for inclusion within a compiled Jupyter Book. From a user's perspective, reuse of content from Jupyter Notebooks should prove easier than with other platforms that would require a user to copy and paste code into a separate computational environment.

Content created using reStructuredText requires more effort to produce, as one must manually format that content with markup headings and code directives. However, reStructuredText also features helpful versioning control functionality that makes tracking changes within content more efficient than other platforms. For example, it is much easier to view document changes (e.g., Git diffs) when using reStructuredText, as opposed to a Jupyter Notebook, considering that a raw Jupyter Notebook, viewed through a text editor, delivers cell data in a JSON format. Furthermore, simply executing a cell changes elements within the JSON text (e.g., cell id, execution count). So, it can be unnecessarily difficult to view any substantial number of code changes within the Jupyter Notebook format.

Despite version control limitations, there remain many benefits to deploying Jupyter Notebooks as a platform for coding tutorials. Perhaps the most obvious benefit, code delivered through a Jupyter Notebook is much easier for the end user to deploy and reuse. In addition, Jupyter Notebooks can be used with Python and Bash code without any special configuration, and there are kernels available for using Mathematica and Matlab. As such, we plan to use Jupyter Notebooks, where possible, for future Cookbook API tutorials.

Another benefit of Jupyter Book relates to the platform's support for cloud computing. For example, Jupyter Book can be configured so that code shared within that publication is launched into an interactive cloud-based computing environment like Google Colab or Binder. This allows users to run the tutorial code without worry for software dependencies on the local machine, which reduces any hardware/software-related barriers one might encounter while using the tutorial.

While the cloud-based functionality of Jupyter Book was of interest to our development team, this feature was disabled in version one of the Cookbook because, to our knowledge, it would not be possible to configure these computing environments to run the Mathematica or Matlab code. We also discovered that some subscription scholarly APIs could not be effectively accessed through an online computing environment, as the associated IP (internet protocol) address fell outside of the university's approved IP range. With that all said, our team continues to explore options for deploying this

functionality in future versions of the Cookbook. In a future Cookbook release, we expect that the Matlab and Mathematica portions of the Cookbook can push code to Matlab Online and Wolfram Mathematica Online platforms in support of a cloud-based experience.

Lastly, as this project moves forward, the team plans to implement several other improvements to both the Cookbook's documentation as well as our development workflows. For example, we plan to develop an internal coding guide for the API tutorials, which will help ensure consistency of variable names, comments, and programming style. In addition, we plan to add content to each tutorial that more thoroughly documents software versions/environments, which will improve the reproducibility of coding recipes. We would also like to leverage test automation frameworks to develop a regimen of automated code testing wherever possible within the Cookbook. This would be more efficient than our current testing protocols, which involve the manual testing of all code locally before publishing tutorials to the online Cookbook. Importantly, an automated testing protocol would provide a means for establishing ongoing maintenance for the Cookbook, which would continue at regular intervals. We expect that in most cases we can check the expected code output within the automated testing, though this may prove challenging as it is expected that the scholarly API data schema, and underlying data, may continue to evolve over time. So we will have to be clever in designing our testing protocols to guard against false errors, which would likely lead us to redesign our tests.

Addition of New API Tutorials

There exists a wide variety of both open and subscription-based scholarly APIs available for acquiring information and data, such as: full-text articles, bibliographic data, U.S. government data, property data, images, books, newspaper articles, and court documents. Furthermore, we expect the availability of scholarly APIs will only increase—along with the need for related training materials to support the research of library users. Therefore, we plan to expand existing API offerings in the Cookbook, as well as port tutorials to the R and C programming languages. This work has already begun, and several new Python and R tutorials are in draft phase within a GitHub pull-request in the Cookbook repository. The new tutorials will be merged into the next version of the Cookbook, likely during summer of 2023. Here it is worth noting that while it may seem odd to port web API research scripts to a compiled language like C, computer science majors on our campus are introduced to the C language during their 100-level coursework. We anticipate that creating simple, practical examples of how to use C for library research will provide for some interesting instructional opportunities with our undergraduates.

Z39.50 for Library Catalog Data

In addition to expanding content for R and C within the Cookbook, we are also actively developing content that supports the use of Z39.50, which provides a means to address an existing gap within the API environment—a gap that impedes programmatic access to traditional library catalog metadata. This is a gap that came to our attention when it was discovered that Harvard Library is one of few, if not the only, library providing access to their catalog metadata via an API (Harvard Library, 2023). This prompted us

to explore the Z39.50 protocol, which was developed many years before web APIs were available.

The Z39.50 standard specifies communication rules between a client/server (e.g., a local computer and library database server), as well as defines a common query language (Lynch, 1997; Ward, 1994). Interestingly, Z39.50 access to library catalogs is still widely available across libraries, including our own library (Library of Congress, 2023). Z39.50 has traditionally been used to support data delivery for graphical-based applications like federated search (Ketchell et al., 1996), as well as for connecting bibliographic software to library catalogs (East, 2003). However, Z39.50 can also be used to search library catalogs programmatically with scripts. Some examples include automating multiple search queries, retrieving information related to the availability of resources, and returning MARC data, which can then be processed with downstream software (See examples in Z39.50 Cookbook Tutorial).

To demonstrate this use case, we added to the Cookbook a Z39.50 tutorial that uses Bash and Index Data's Yaz software—specifically the command line yaz-client program (Index Data, 2022) to query and retrieve data from our library catalog. This tutorial may prove less useful in time as more APIs become available for library catalogs, though, at least for the next several years, using yaz-client with a Bash script appears to be the best option for searching library catalog data via programmatic scripting. As an aside, we found the reverse Polish notation query language used by Z39.50 tricky to use at the time. However, being able to search library catalogs programmatically from a terminal was interesting enough to tamp down any frustrations; that is, compared to a traditional web interface for searching the library catalog, using a terminal interface presents an additional element of computing to the information discovery process.

Contributing to the Cookbook

As previously mentioned, the University of Alabama Libraries Scholarly API Cookbook is an open-source project with an MIT license. Anyone can reuse and adapt the individual content, or even fork the entire Cookbook. We highly encourage external contributions to the Cookbook. Such contributions might include feedback, bug reports, submitting a new or existing scholarly API tutorial, or even suggesting a new API or use case to add. In order to incentivize contributions, we are exploring ways to incorporate DOIs into individual tutorials so that authors can receive a defined citation for their efforts. Consequently, this will also provide a means for more stable link-outs from related library guides. In addition, as the Cookbook continues to grow, we are exploring adding an index to the Cookbook to improve discoverability. We also plan to add a highly visible contributor guide to the Cookbook, which would detail preferred formatting and required information for contributed work.

Conclusion

We suspect that library user interest in using scholarly APIs will continue to rise, and as a result, libraries will need to become familiar with supporting APIs as part of their research data services. To this end, we created the University of Alabama Libraries Scholarly API Cookbook to help researchers get started with writing code for

interacting with scholarly APIs programmatically. Development of the API Cookbook was a collaborative effort and included building a team of student developers. The Cookbook uses the Jupyter Book platform and currently contains tutorials for working with 10 different scholarly APIs as well as a tutorial on accessing library catalog data programmatically through Z39.50, as this data are not yet readily available via APIs. Tutorials are provided for each API in several different programming languages, including Python, Bash, Matlab, and Mathematica. We plan to continue maintaining and updating the Cookbook with new API tutorials, as well as additional programming languages such as R. The University of Alabama Libraries Scholarly API Cookbook is hosted on GitHub as a Jupyter Book. All code is open-source and MIT licensed. As a result, users are highly encouraged to reuse, comment, and contribute to the Cookbook.

Acknowledgments

We thank the administration of the University of Alabama Libraries for their support of this work, and the Libraries Area Computing Services team for their willingness to deploy Linux workstations in support of our program's code development and instructional offerings.

Data Availability Statement

All source code is available on GitHub:
https://github.com/ualibweb/UALIB_ScholarlyAPI_Cookbook

Definition of Terms

Research data services is a relatively new (past two decades), and still expanding, area of library practice, focused on data-intensive, technology-infused research.

A *starter script* is a snippet of programming code provided to novice programmers in support of their goals of learning a new programming language or tool.

Jupyter Notebook is a computational notebook for code and text, whereas Jupyter Book is a broader publication platform that allows one to create/publish more expansive projects. In the case of the project described herein, Jupyter Book provided a means for linking together multiple Jupyter Notebooks in support of creating a single "book" (i.e., the Scholarly API Cookbook).

Within the context of open-source code development, and GitHub in particular, a *pull request* is a user-initiated process by which new code is posted to the repository for review by project leaders before it is merged with the existing project code.

Within a Jupyter Notebook, code is presented/delivered through boxed sections of the document called *cells*.

JSON stands for JavaScript Object Notation, which is a commonly used open standard file format. Reading Python that is delivered in a JSON format can prove cumbersome.

Executing a cell means running the chunk of code found within that cell.

A *kernel* is defined as the portion of an operating system's base code, which controls most, if not all, interactions between that system and the hardware running it. To say that there is a Jupyter Notebook kernel for Matlab simply means there is a version of the notebook that can be deployed that makes it possible to deliver/run Matlab code to the end user (i.e., the person using the tutorial).

In cloud-based computing, there is no need for specialized software on the computer being used to access the tutorial content. This can prove beneficial for students accessing content through lab or library-based computers that prevent users from loading software not included in the base machine image.

To *port* code means to translate, or adapt, existing code for use with another programming language.

To *fork* code means to copy and build upon existing open-source code to develop a new software.

References

BCDS | DS Learn: REST API. (2022). Boston College Libraries Digital Scholarship Group. https://bcds.gitbook.io/learn/tutorials/rest-api

Boston College Libraries. (2023). *US data and statistics sources: APIs for scholarly resources.* https://libguides.bc.edu/c.php?g=43963&p=8347274

Chen, H., Li, X., Chau, M., Ho, Y.-J., & Tseng, C. (2009). Using open web APIs in teaching web mining. *IEEE Transactions on Education, 52*(4), 482–490. https://doi.org/10.1109/TE.2008.930509

Data Ethics Framework Development Team. (n.d.) *Federal data strategy, Data ethics framework.* General Services Administration. https://resources.data.gov/assets/documents/fds-data-ethics-framework.pdf

East, J. W. (2003). Z39.50 and personal bibliographic software. *Library Hi Tech, 21*(1), 34–43. https://doi.org/10.1108/07378830310467382

Executable Books Community. (2020). *Jupyter book (v0.10).* Zenodo. https://doi.org/10.5281/ZENODO.4539666

Federer, L., Foster, E. D., Glusker, A., Henderson, M., Read, K., & Zhao, S. (2020). The Medical Library Association data services competency: A framework for data science and open science skills development. *Journal of the Medical Library Association, 108*(2). https://doi.org/10.5195/jmla.2020.909

Fergusson, A., & Pfannkuch, M. (2022). Introducing high school statistics teachers to predictive modelling by exploring dynamic movie ratings data: A focus on task design. *Statistics Education Research Journal, 21*(2), 8. https://doi.org/10.52041/serj.v21i2.49

Fergusson, A., & Wild, C. J. (2021). On traversing the data landscape: Introducing APIs to data-science students. *Teaching Statistics, 43*(S1). https://doi.org/10.1111/test.12266

Goodale, I. (2020). *Working with APIs: An introduction to application programming interfaces.* https://guides.lib.utexas.edu/c.php?g=897091

Harvard Library. (2023). *Harvard Library APIs & datasets.* https://library.harvard.edu/services-tools/harvard-library-apis-datasets

Huck, J. (2019). *Census data with R.* https://jennhuck.github.io/workshops/tidycensus.html

Index Data. (2022). *Index data Yaz.* https://www.indexdata.com/resources/software/yaz/

Ketchell, D. S., Freedman, M. M., Jordan, W. E., Lightfoot, E. M., Heyano, S., & Libbey, P. A. (1996). Willow: A uniform search interface. *Journal of the American Medical Informatics Association, 3*(1), 27–37. https://doi.org/10.1136/jamia.1996.96342647

Kim, S., Bucholtz, E. C., Briney, K., Cornell, A. P., Cuadros, J., Fulfer, K. D., Gupta, T., Hepler-Smith, E., Johnston, D. H., Lang, A. S. I. D., Larsen, D., Li, Y., McEwen, L. R., Morsch, L. A., Muzyka, J. L., & Belford, R. E. (2021). Teaching cheminformatics through a collaborative intercollegiate online chemistry course (OLCC). *Journal of Chemical Education, 98*(2), 416–425. https://doi.org/10.1021/acs.jchemed.0c01035

Library of Congress. (2023). *Gateway to library catalogs: Z39.50.* http://www.loc.gov/z3950/

Lynch, C. A. (1997). The Z39.50 information retrieval standard. *D-Lib Magazine, 3*(4). https://www.dlib.org/dlib/april97/04lynch.html

McCown, F. (2010). Teaching web information retrieval to undergraduates. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education,* 87–91. https://doi.org/10.1145/1734263.1734294

MIT Libraries. (2023). *Resources and tools for computational research.* https://libguides.mit.edu/comptools

Olsen, T., & Moser, K. (2013). Teaching web APIs in introductory and programming classes: Why and how. *Proceedings of the AIS SIG-ED IAIM 2013 Conference.* https://core.ac.uk/download/pdf/301362169.pdf

Scalfani, V. F., Ralph, S. C., Alshaikh, A. A., & Bara, J. E. (2020). Programmatic compilation of chemical data and literature from PubChem using MATLAB. *Chemical Engineering Education, 54*(4). https://doi.org/10.18260/2-1-370.660-115508

Scalfani, V. F. (2021). Using NCBI Entrez Direct (EDirect) for small molecule chemical information searching in a Unix terminal. *Journal of Chemical Education, 98*(12), 3904–3914. https://doi.org/10.1021/acs.jchemed.1c00904

Swygart-Hobaugh, M., Anderson, R., George, D., & Glogowski, J. (2022). Diving deep into dissertations: Analyzing graduate students' methodology and data practices to inform research data services and subject liaison librarian support. *College & Research Libraries, 83*(6). https://doi.org/10.5860/crl.83.6.887

Sydow, D., Morger, A., Driller, M., & Volkamer, A. (2019). TeachOpenCADD: A teaching platform for computer-aided drug design using open source packages and data. *Journal of Cheminformatics, 11*(1), 29. https://doi.org/10.1186/s13321-019-0351-x

Tenopir, C., Hughes, D., Allard, S., Frame, M., Birch, B., Baird, L., Sandusky, R., Langseth, M., & Lundeen, A. (2015). Research data services in academic libraries: Data intensive roles for the future? *Journal of EScience Librarianship, 4*(2), e1085. https://doi.org/10.7191/jeslib.2015.1085

Tenopir, C., Pollock, D., Allard, S., & Hughes, D. (2016). Research data services in European and North American libraries: Current offerings and plans for the future. *Proceedings of the Association for Information Science and Technology, 53*(1), 1–6. https://doi.org/10.1002/pra2.2016.14505301129

UA Libraries Workshops. (2023). https://github.com/ualibweb/UALIB_Workshops

UC San Diego Library. (2023). *Finding data & statistics: APIs – scholarly resources.* https://ucsd.libguides.com/data-statistics/apis

UCLA Library. (2023). *Scholarly APIs.* https://guides.library.ucla.edu/scholarly-apis

Ward, M. (1994). Expanding access to information with Z39.50. *American Libraries, 25*(7), 639–641.

Wolfram Research. (n.d.). *Wolfram Language kernel for Jupyter notebooks.* Retrieved February 17, 2023, from https://github.com/WolframResearch/WolframLanguageForJupyter

Yoon, A., & Donaldson, D. R. (2019). Library capacity for data curation services: A US national survey. *Library Hi Tech, 37*(4), 811–828. https://doi.org/10.1108/LHT-12-2018-0209