Automation of the Digital Research Support System

Vaibhavi S.

Abstract

Purpose

Ashoka University has started curating research guides on its website, featuring topics submitted by its students to aid them in their research. However, with most of the library's books not being correctly classified into neat and proper subject areas on the Koha platform and in the absence of a service provider procured by the library that grants an Application Programming Interface (API) to retrieve all articles on a topic that the library provides access to, it took weeks for the librarians to create a single guide manually. Therefore, an open-source web application has been developed from scratch in the library to automate the curation of research guides — the logic and architecture of which is presented in this paper.

Methodology

Koha API and Google Books API have been integrated to retrieve data on physical books. As for the articles' data, OpenAlex API has been utilised in combination with a CSV (comma-separated values) file containing the database of all the journals subscribed to by the library. The data is then made accessible to the user through a web application.

Findings

The simple design of the program showcased difficulty in handling a few edge cases, due to which some books and articles weren't retrieved in spite of them being made available by the library. While for our purpose, these cases didn't pose a problem, we have suggested ways to modify the program and fix them.

Value

The paper explains the working of the open-source web application, so that any other library that does not wish to procure a paid service for curation of subject or research guides can use our application by simply replacing our database with their library's database.

Introduction

Background

Our library uses the Koha platform to display its collection of physical books. However, the platform has a limitation in that the search system in Koha currently only matches exact words entered by the user with the title, subject, and other fields in a book's data. However, a significant number of the books in the library do not have all the subjects they belong to correctly entered in their subject fields. Moreover, not all books on a specific subject have that subject's name as part of their title or other fields. Consequently, to retrieve all books on a specific topic, the librarians had to first search for all the books on that topic on the internet and then, for each of them, check if they are present in our collection.
As for the articles, while we use the Knimbus platform to display all subscribed and open-access articles in response to a search, we preferred to add the titles and links to the articles in the guides themselves for completeness, instead of directing users to another platform. Therefore, for a topic, the librarians had to first copy the titles of the articles and then click on each of them to copy their links to add them to the guides.
Given the large size of our collection, it took the librarians weeks to curate a single guide, which delayed delivery of resources to the researchers and was also not an optimal use of their time.
Our web application aims to automate this process of retrieving books and articles so as to deliver resources to the users faster and also optimise the use of the librarians' time.

System Architecture

Overview

To overcome Koha's search system's limitation, we decided to first get a list of relevant books on the requested topic through Google Books using its API. This is beneficial as Google Books uses many advanced techniques as part of its search system [2], like searching the content of books [3].

As for the retrieval of articles, the Knimbus platform procured by the library, currently does not provide an API to its users. Consequently, we could not directly get a list of articles we provide access to through the program. To solve this, OpenAlex was used as its API is openly accessible and does not require special authentication [4]. We first used its API to retrieve a list of articles on a topic and then added a step to filter out those articles to which the library provides access. For this filtering, a CSV sheet was prepared that has a list of all the journal titles and publication year ranges of their articles to which the library has subscribed.

The source code for the system has been made available as an open-source project on GitHub.

Backend Workflow

Workflow for retrieving the books

We first ran a program to retrieve and save the data of all the books in our library to the system which runs the web application. This is done so that every time the application needs to display books on a topic, it doesn't have to retrieve our collection data again. The program uses the Koha API (the documentation of which can be found at: https://api.koha-community.org/) to retrieve and store all the books' ISBNs, titles, and Koha links to the system in JSON (JavaScript Object Notation) format, in a file named 'data.json'. The JSON format was chosen as it is lightweight and easy for machines to parse and generate [1].

Through the web application, when a user requests available books on a topic, first, the ISBNs of all the books on that topic are received through the Google Books API by passing the topic as a search query parameter in the API call, and then parsing the response received to extract the ISBNs.
Then, our library's books' data is loaded by loading the 'data.json' file into the system. The program then matches each ISBN received through the Google Books API with our library's data to check which books on the topic are present in the library. For the books that are present, their titles and Koha links are displayed to the user.

Workflow for retrieving the articles

The retrieval of articles is done in two parts:

1. Open-Access Articles: First, a list of the open-access articles on the requested topic is received by setting the open-access status of the articles to true using filters and passing

the topic as a search query parameter in the API call.
The titles and links of these are extracted from the response received and added to a list named ‘result’. The number of articles received in the response can be changed by modifying the number of pages of results retrieved and the number of articles per page in the program.

2. Closed-access articles: Similar to retrieving open-access articles, a list of closed-access articles is received by changing the open-access status of the articles to false in the filters of the API call.
   Next, to filter out the articles that the library provides access to, each article's journal title and publication year are extracted and matched with the CSV sheet that has the library’s subscribed journal titles’ data. For the articles that we have subscribed to, their titles and links are retrieved and added to the list ‘result’.

In the end, all the titles and links stored in the list ‘result’ are shown to the user.

Discussion

Limitations

Some books in the library do not have an ISBN. Therefore, checking the availability of books in the library by only matching the ISBNs results in some books being left out.

In the retrieval of articles, in some cases, the journal titles do not have a single publication year range (link 1997-2025) for the subscribed articles, but have multiple ranges (like 1997-2000 and 2005-2025). In such cases, we have kept the larger range in the CSV sheet used to check article access.
Furthermore, some journal titles also have volume and issue numbers in addition to publication years that we have subscribed to, like Volume 81, number 3 (July 1995) to Volume 66, number 1 (fall 2014). In such cases, to simplify the ranges, we have considered the year ranges in which we have access to all articles of the journal. For example, in the provided example, we would consider the range 1996 to 2013.
Both these cases resulted in some subscribed articles being left out.

Future Improvements

To display even those books that do not have an ISBN, additional fields, like titles, could also be used to match the books retrieved from Google Books with the library's books.
Next, one way to solve the problem of multiple ranges could be adding the journal title multiple times to the CSV sheet, where each entry has one range of the title. The algorithm could then be modified to match all entries of the journal title in the sheet instead of just the one, as it does in the current version of the application.
Furthermore, the range could be specified in more detail by also considering the month in addition to the year, and checking access by extracting the full publication date (which includes the month) of the articles from the data received from the OpenAlex API.

We have not implemented these changes as for our purpose of curating guides, we are not looking for an exhaustive list of all books and articles on a topic but a specific number of results, like 20 books and 100 articles. Therefore, increasing the program complexity and consequently increasing the time taken to get the results is unnecessary in this case.

Conclusion

In this paper, we have presented a study of a web application solution that we created to address the issue of the immense amount of time required to curate research guides in the absence of a paid software solution procured by a library. The architecture of the program behind the application, along with the reasons behind its steps, was explained to aid individuals in freely modifying and customising the application for use in their own libraries in the future. The program did have limitations, which were discussed along with some ways to address those. Moreover, if we create more versions of the application, they will be made freely available on the GitHub page where the source code of our application has been uploaded.

References

1. https://www.json.org/json-en.html
2. https://ieeexplore.ieee.org/abstract/document/4377029
3. https://www.google.com/intl/en/googlebooks/about/index.html
4. https://www.sciencedirect.com/science/article/pii/S2215016124000554