International Journal of User-System Interaction 15(4) 2022, 101-113

Distributed search engine in digital libraries

Mirel Cosulschi[1], Valentin Ispas[2], Marian Cristian Mihăescu[2]

[1]University of Craiova, Department of Computer Science
Al. I. Cuza Street, 13, Craiova, Romania

[2]University of Craiova, Department of Computers and Information Technology
Blvd. Decebal, 107, Craiova, Romania

Abstract. This paper aims to describe a meta-search engine in several digital libraries, providing a simplified solution for searching scientific articles and documents from multiple sources. The application aims to facilitate searching and accessing scientific articles by providing users with a powerful and interactive search engine. The intuitive application interface offers advanced features that allow users to quickly and efficiently obtain the information needed, explore article details, and download relevant materials for later use in academic or research activities. Building a system that consolidates data from various sources has been proven to be a task that requires both software engineering and human-computer interaction skills. As the number and diversity of digital libraries that provide an API is increasing, we foresee that the development of search engines will receive increasing attention.

Keywords: Metasearch; information resource discovery; distributed collection; collection fusion; distributed information retrieval.

DOI: 10.37789/ijusi.2022.15.4.2

1. Introduction

This work aims to develop an advanced search engine in digital libraries that allows users to find relevant scientific articles on various topics of interest. The meta-search engine provides relevant details about each paper, including publication date, associated fields of study, authors, and number of citations. In addition, for some materials, users can download the articles as PDF documents.

A well-designed and implemented metasearch engine can provide many benefits, including improved resource access, search efficiency, optimized user experience, and effective content management. By exploring and analyzing the issues mentioned above, this work aims to contribute to

developing search systems through digital libraries and improving access and efficient use of digital resources.

By identifying the best practices and challenges associated with the domain of distributed search, the study can contribute to developing the academic and professional community to optimize the search experience and facilitate access to relevant information.

A graphical user interface was developed using the React library to facilitate easy interaction between the user and the application, which provides developers with an efficient solution for implementing interfaces. This interface provides an intuitive and easy-to-use interaction, allowing users to enter keywords to search for scientific articles in various external digital libraries.

The developed application supports several digital libraries, allowing users to search various information sources. These digital libraries include Semantic Scholar, DBLP, Elsevier, and Google Scholar.

The remainder of this paper is organized as follows. Section 2 presents related work, while Section 3 describes the data sources for the metasearch engine in more detail. In Section 4, we describe the application architecture. Finally, in Section 5, we provide the conclusions of our research and describe possible future work.

2. Related work

Ortega (2014) describes the main characteristics of six academic search engines in his book, highlighting their advantages and drawbacks and discussing their impact on research measurement and evaluation.

Meng & Yu (2010) describe the technology of large-scale metasearch engines, which are search systems that support unified access to multiple existing search engines. This work offers insights into the concept of metasearch engines, their operational principles, challenges, and strategies for result merging. Metasearch engines have the potential to overcome the limitations of major search engines, such as the coverage of the Deep Web. It serves as a valuable resource for understanding the role of metasearch engines in the broader context of information retrieval and database systems.

The paper O'Hara (2012) evaluates the impact of implementing the Summon discovery tool at the University of Manitoba Libraries, focusing on pre- and post-implementation analysis, user experience, and the overall

success of the search process. The insights gained from this study contribute to the literature on planning and implementing resource discovery tools in academic library settings.

The paper by Chaurasia (2011) introduces a meta-search engine that incorporates a prioritization mechanism to improve the relevance and order of search results. The prioritizer is a crucial component to enhance the user experience, and the paper likely provides insights into the operational framework and evaluation of the proposed system.

3. Data sources for digital search engine

3.1. Semantic Scholar

Semantic Scholar was developed by the Allen Institute for Artificial Intelligence, a research organization founded by Microsoft co-founder Paul Allen. Semantic Scholar is a search engine that combines machine learning, natural language processing, and machine vision to identify connections between various articles in computer science and biomedical journals (Kinney, Anastasiades et al. (2023)).

The primary purpose of Semantic Scholar is to help researchers, academics, and students find relevant and influential scientific papers. It covers various disciplines, including computer science, biomedical sciences, neuroscience, physics, etc. Content indexed by Semantic Scholar includes research articles from journals, conference proceedings, preprints, and other scholarly sources.

What sets Semantic Scholar apart from other search engines is its use of AI and NLP technologies to analyze and understand the content of academic papers. It uses machine learning algorithms to extract essential information from articles, such as quotes, figures, tables and references. This enables advanced search capabilities and allows users to access more comprehensive and contextually relevant results.

Semantic Scholar also uses artificial intelligence-based algorithms to identify and highlight essential information in research papers. For example, it can automatically extract and display key phrases, influential quotes, and important figures or tables. This feature helps researchers quickly understand a paper's main ideas and conclusions without reading it entirely (Kinney, Anastasiades et al. (2023)).

The Semantic Scholar REST API allows us to find and explore scientific publication data about authors, papers, citations, venues, etc[4].

To retrieve an author ID, the API can be queried with the following request:

https://api.semanticscholar.org/graph/v1/author/search?query=paolo+merialdo

The response to the above request could be:


```
{
  "total": 1,
  "offset": 0,
  "data": [
    {
      "authorId": "1796590",
      "name": "P. Merialdo"
    }
  ]
}
```


Additionally, if we want to get information about all the works of a particular author, we can use a query with the following structure:

https://api.semanticscholar.org/graph/v1/author/search?query=paolo+merialdo&fields=name,aliases,url,papers.title,papers.year&limit=5

The response for the previous request could be:


```
{
  "total": 1,
  "offset": 0,
  "data": [
    {
      "authorId": "1796590",
      "url": "https://www.semanticscholar.org/author/1796590",
      "name": "P. Merialdo",
      "aliases": [
        "P Merialdo",
        "P. Merialdo",
        "Paolo Merialdo"
      ],
      "papers": [
        {
          "paperId": "de0fa8cc07481293c941507a689e5931b22d93ff",
          "title": "Experiences and Lessons Learned from the SIGMOD
    Entity Resolution Programming Contests",
          "year": 2023
        },
        {
```


[4] https://www.semanticscholar.org/product/api

```
            "paperId": "0217aa3acc27d07a2ea2aa53618459943a8d97e2",
            "title": "Effective Explanations for Entity Resolution
      Models",
            "year": 2022
          },
…
```


3.2 DBLP

DBLP is a computer science bibliography website[5]. The beginning of DBLP is linked to the Universität Trier in Germany. It began as a small collection of HTML files in 1993 (Ley (2009)) to become a computerized bibliographic database with bibliographic information about 3,290,176 authors, 6,256 conferences, and 1,838 journals[6]. Since 2018, DBLP has become part of the Schloss Dagstuhl – Leibniz-Zentrum für Informatik (LZI) family.

All major computer science journals are tracked. The proceedings of many conferences are also followed.

There are three primary search services provided by DBLP: one for publications, one for persons (authors/editors), and one for venues (journals/conferences/etc). The base search API URLs of these services are the following[7]:

https://dblp.org/search/publ/api - for publication queries;
https://dblp.org/search/author/api - for author queries;
https://dblp.org/search/venue/api - for venue queries.

For example, to get information about the same author as in the case of Semantic Scholar, the structure of the request will be:

https://dblp.org/search/publ/api?q=author%3APaolo_Merialdo%3A&format=xml

The corresponding answer to the previous request could be:


```
<result>
  <query id="309630">:facetid:author:\"Paolo_Merialdo\"</query>
  <status code="200">OK</status>
  <time unit="msecs">1.20</time>
  <completions total="1" computed="1" sent="1">
    <c            sc="109"           dc="109"           oc="109"
     id="17963668">:facetid:author:Paolo_Merialdo</c>
  </completions>
```


[5] https://en.wikipedia.org/wiki/DBLP
[6] https://dblp.org/statistics/recordsindblp.html
[7] https://dblp.org/faq/How+to+use+the+dblp+search+API.html

```
<hits total="109" computed="100" sent="30" first="0">
...
  <hit score="1" id="164350">
    <info>
      <authors>
        <author pid="221/7529">Federico Piai</author>
        <author pid="a/PAtzeni">Paolo Atzeni</author>
        <author pid="m/PaoloMerialdo">Paolo Merialdo</author>
        <author                  pid="s/DiveshSrivastava">Divesh
   Srivastava</author>
      </authors>
      <title>
          Fine-grained semantic type discovery for heterogeneous
   sources using clustering.
      </title>
      <venue>VLDB J.</venue>
      <volume>32</volume>
      <number>2</number>
      <pages>305-324</pages>
      <year>2023</year>
      <type>Journal Articles</type>
      <access>open</access>
      <key>journals/vldb/PiaiAMS23</key>
      <doi>10.1007/S00778-022-00743-3</doi>
      <ee>https://doi.org/10.1007/s00778-022-00743-3</ee>
      <url>https://dblp.org/rec/journals/vldb/PiaiAMS23</url>
    </info>
    <url>URL#164350</url>
  </hit>
  ...
```

3.3 Elsevier

Elsevier is a Dutch academic publishing company focusing on science, technology, and medicine. It is headquartered in Amsterdam, Netherlands[8]. Among their products, journals such as The Lancet, Cell, and the ScienceDirect collection of electronic journals are worth mentioning.

Elsevier's products and services include Scopus and Scival: Scopus is an online multidisciplinary citation database of research literature, while Scival is a Web-based analytics tool to visualize research performance, benchmark against peers using several metrics, and review co-author networks[9].

For users who want to search and retrieve data from Elsevier products

[8] https://www.elsevier.com/about
[9] https://www.elsevier.com/solutions

programmatically, the publishing company provides APIs for interested researchers. More information can be consulted on the Elsevier Developers website. Access is granted via an API key. Among the various services available through APIs, users can access citation data, metadata, and abstracts from academic journals as indexed by Scopus[10].

Elsevier provides API access to subscribing institutions, allowing users to search and retrieve data programmatically from research databases such as Scopus, Embase, and Engineering Village.

For example, a query in the Elsevier databases for an author looks like this:

```
https://api.elsevier.com/content/search/author?query=authlast(Mer
ialdo)%20and%20authfirst(Paolo)&apiKey=7f59af901d2d86f78a1fd
60c1bf9426a
```

3.4 Google Scholar

Google Scholar is a freely web-based search engine designed to search scholarly literature, including articles, conference papers, theses, preprints, and other academic publications. Google developed it, and it was first released in November 2004.

The primary purpose of Google Scholar is to provide researchers, academics, and students with a comprehensive and convenient way to search for scholarly information. It indexes various disciplines, including science, engineering, and medicine. The content indexed by Google Scholar is sourced from multiple publishers, academic institutions, and online repositories (Alfonzo (2016)).

Through its citation tracking capability, Google Scholar can show how often a given article has been cited by other researchers, allowing users to gauge the impact and influence of a publication within the academic community.

Google does not provide an official API for Google Scholar. Some third-party solutions support profile, author, citation, and organic results.

[10] https://dev.elsevier.com/

4. Application description

4.1. Application Architecture

The application aims to implement a distributed search engine in digital libraries and aggregate the results into a single response page for the user. The search engine allows users to select the articles they wish to download in BibTeX format.

The application interface, developed in React.js, is essential in facilitating the interaction between the user and the application. It provides an intuitive platform for the user, offering the ability to search various digital libraries with ease.

Node.js is used for communication between the front end and the web application server. It handles calls to external APIs of digital libraries such as Semantic Scholar, DBLP, Elsevier, and Google Scholar. The main function of the application server is to call all external APIs and return an aggregated list of articles to the user interface.

In figure 1 is depicted the application architecture.

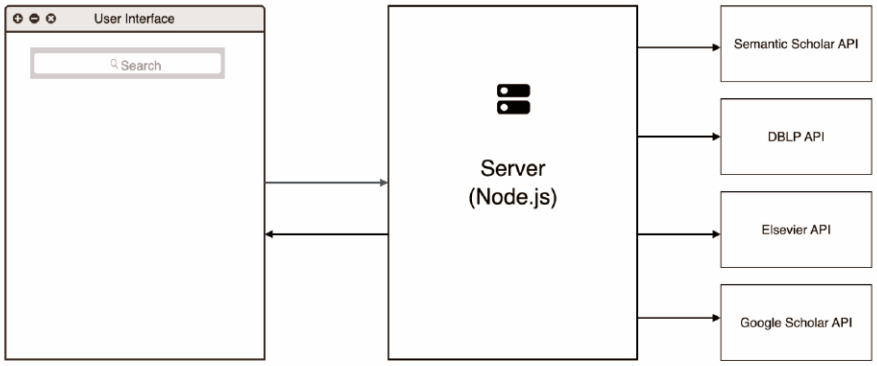

Figure 1. The application architecture.

4.2 Application User Interface

The user interface, an essential element of a web application, was developed using React.js to build a distributed search engine for scientific articles in digital libraries. This interface provides the user with an intuitive way to

interact with the search engine and efficiently find relevant information.

The application interface allows users to communicate with the application server, which makes calls to the digital library APIs and then aggregates all results. The user can then select, by ticking, the articles they wish to download in BibTeX format.

Before starting a search, the user can choose which libraries to search by checking that library next to the keyword input field (see Figure 2).

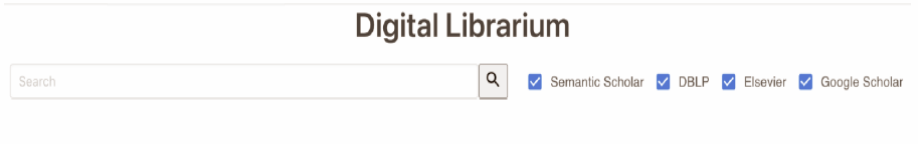

Figure 2: Choosing the data source(s).

After the user initiates the request, the keywords entered in the search box are retrieved and sent to the application server via an HTTP request. This application contains the information needed to search for articles in digital libraries.

Through the HTTP protocol, the server receives the keywords and can initiate the search process, using them as parameters for querying digital libraries.

Thus, by sending the keywords through the HTTP request, the communication between the user interface and the server is facilitated, and the latter can search for articles in the digital libraries according to the user's requirements.

App, *Homepage*, *SearchBar*, *CardList* and *Card* are *React* components created for application development. The functional components were used in the implementation of the application.

The architecture of React components in the user interface is described in Figure 3.

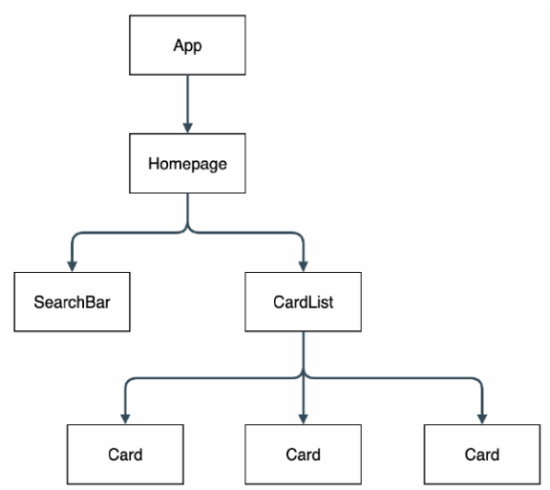

Figure 3. The architecture of the client side of the application.

App is the main component and the gateway to the user interface. This has a child component named *Homepage*.

The *Homepage* component retrieves the text the user enters from the *SearchBar* component and sends it in a request to the web server using the *axios* library, an HTTP client for Node.js.

The *SearchBar* component is essential in taking the keywords the user enters and transmitting them to the *Homepage* component. This functionality is achieved through a form containing a search box, where the user enters the desired text.

The text the user enters is collected and transmitted to the top component, *Homepage*, via the *onSearch* function.

Through this architecture, the *SearchBar* component captures and passes the keywords the user enters to the upper component, allowing the application to perform specific searches and provide relevant results to the user.

When the user interface receives the response from the server, which contains the articles found in the search based on the keywords entered by the user, it will display relevant information about each scientific paper found. This information includes the authors of the work, the field of activity, and the date or year of publication.

Within the interface, there will also be a "PDF" button indicating that the paper has an open-access PDF file for users to open or read. This functionality allows users to save and access a document containing the entire article or the

abstract, along with relevant images and other important aspects of the paper.

Through this mechanism, the user interface facilitates the presentation of relevant details about each article found and allows users to access and explore the entire content of the papers through the available PDF files. This functionality improves the user experience by allowing them to delve deeper into the content of the papers and access additional information for research or study.

If the user clicks on the title of an article, he will be redirected to the specific digital library from where that paper was taken. The user can access additional information about the article or explore other similar scientific works through this redirection. This functionality allows the user to engage more deeply with the content and explore more resources within the digital library.

Figure 4. The result of a search.

5. Conclusions

The developed application aims to implement a meta-search engine in several digital libraries, providing a simplified solution for searching scientific articles and papers in multiple sources. This search engine aggregates results from various libraries into a single search, providing users with an efficient way to find relevant information.

This application allows users to search for scientific papers using specific

keywords and get results from multiple sources simultaneously. This eliminates the need to search each library manually, saving the user time and effort.

Although Google Scholar provides implicit aggregation functionalities of search results from multiple data sources, it fails to provide explicit and custom selection of data sources. From this perspective, the proposed system represents a prototype application of a professional distributed search engine with specific selection and export functionalities.

Among the possibilities for future development are the following: increasing the number of data sources, implementation of an option for the user to select the field to search by (title, author, etc.) if digital libraries allow it, grouping of scientific papers according to the domain of study or according to the authors of the papers.

Finally, a formal comparative analysis with similar general-purpose applications (i.e., Google Scholar, Semantic Scholar, etc.) in terms of HCI methodology and UI/UX evaluation by user testing would clarify the advantages and disadvantages of the proposed application and clearly point out future research and development directions (Eberts (1994), Krug (2000), Norman (2013)).

References

Alfonzo, P. (2016) Teaching Google Scholar: A Practical Guide for Librarians, Rowman & Littlefield.

Chaurasia, B. K., Gupta, S. K., and Soni, R. (2011) Meta Search Engine Based on Prioritizor, 2011 *International Conference on Computational Intelligence and Communication Networks*, pp. 512-514, doi: 10.1109/CICN.2011.109.

Elsevier API Interface Specification, https://dev.elsevier.com/api_docs.html

Elsevier API Query Tool–the Data Fetcher. Getting Started Guide, ver 7.4.3, 2023, https://dev.elsevier.com/data-fetcher-resources/DataFetcherManual_7_4_3.pdf

Eberts, R. E. (1994) *User interface design*, Englewood Cliffs, Prentice Hall.

Ghafari, M., Mortaza, S., Touraj, I. (2012) A federated search approach to facilitate systematic literature review in software engineering, *Int. J. Softw. Eng. Appl.* 3(2), 13–24.

Hannousse, A. (2021) Searching relevant papers for software engineering secondary studies: Semantic Scholar coverage and identification role. *IET Soft.* 15, 126–146.

Kinney, R., Anastasiades, C. et al (2023) The Semantic Scholar Open Data Platform, *arXiv Preprint*, https://arxiv.org/abs/2301.10140.

Krug, S. (2000) *Don't Make Me Think: A Common Sense Approach to Web Usability*, New Riders, Berkeley.

Ley, M. (2009) DBLP - some lessons learned, *PVLDB* 2(2), 1493–1500.

Meng, W., Yu, C. T. (2010) *Advanced Metasearch Engine Technology*, Synthesis Lectures on Data Management, Morgan & Claypool Publishers.

Norman, D. A. (2013) *The Design of Everyday Things*, New York.

O'Hara, L., Nicholls, P., Keiller, K. (2012) Search Success at the University of Manitoba Libraries Pre- and Post-Summon Implementation, Planning and Implementing Resource Discovery Tools in Academic Libraries, 10.4018/978-1-4666-1821-3.ch015, 268-287.

Ortega, J. L. (2014) *Academic Search Engines*. *A Quantitative Outlook*, 1st Edition, Chandos Publishing