SOFTWARE TOOL ARTICLE

AQUA: an Advanced QUery Architecture for the SPARC Portal

[version 1; peer review: 1 approved with reservations, 1 not approved]

Niloofar Shahidi ¹, Xuanzhi Lin ², Yuda Munarko ¹, Laila Rasmy³, Tram Ngo⁴

¹Auckland Bioengineering Institute, The University of Auckland, Auckland, 1010, New Zealand

²Case Western Reserve University, Cleveland, OH, 44106, USA

³School of Biomedical Informatics, University of Texas Health Science Center at Houston, Houston, TX, 77030, USA

⁴California Medical Innovations Institute Inc, San Diego, CA, 92121, USA

v1 First published: 16 Sep 2021, 10:930
<https://doi.org/10.12688/f1000research.73018.1>

Latest published: 16 Sep 2021, 10:930
<https://doi.org/10.12688/f1000research.73018.1>

Abstract

The Stimulating Peripheral Activity to Relieve Conditions (SPARC) program integrates biological and neural information to create anatomical and functional maps of the peripheral nervous system. The SPARC Portal hosts a dynamic storage for the datasets, models, and resources to help the researchers find and produce data. Currently, the SPARC Portal provides a primary search tool, which lacks some features to improve the search experience. To purposefully retrieve the required information from the stored datasets and resources, we have developed an Advanced QUery Architecture (AQUA) for the SPARC Portal. Near-real-time auto-completion of the queries, close-matches suggestions, and multiple filters to narrow or sort the results are the major features of AQUA with the goal to enhance the usability of the SPARC search engine. AQUA is available from: <https://github.com/SPARC-FAIR-Codeathon/aqua>

Keywords

AQUA, SPARC, biological query, natural language processing, NIFS Ontology, text mining, Codeathon

This article is included in the **Hackathons** collection.

Open Peer Review

Approval Status

	1	2
version 1		
16 Sep 2021	view	view

1. **Vijay Rajagopal** , University of Melbourne, Melbourne, Australia

2. **Maryann E. Martone** , University of California, San Diego, San Diego, USA

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Niloofar Shahidi (nsha457@aucklanduni.ac.nz)

Author roles: **Shahidi N:** Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Lin X:** Software; **Munarko Y:** Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Software, Validation, Visualization; **Rasmy L:** Conceptualization, Data Curation, Formal Analysis, Methodology, Software, Validation; **Ngo T:** Conceptualization, Data Curation, Investigation, Methodology, Project Administration, Software, Supervision, Visualization

Competing interests: No competing interests were disclosed.

Grant information: The author(s) declared that no grants were involved in supporting this work.

Copyright: © 2021 Shahidi N *et al.* This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Shahidi N, Lin X, Munarko Y *et al.* **AQUA: an Advanced QUery Architecture for the SPARC Portal [version 1; peer review: 1 approved with reservations, 1 not approved]** F1000Research 2021, 10:930
<https://doi.org/10.12688/f1000research.73018.1>

First published: 16 Sep 2021, 10:930 <https://doi.org/10.12688/f1000research.73018.1>

F1000 Research Statement of Endorsement

David Nickerson confirms that the author has an appropriate level of expertise to conduct this research, and confirms that the submission is of an acceptable scientific standard. David Nickerson declares he is NF's primary supervisor and one of the organisers of the 2021 SPARC FAIR Codeathon. Affiliation: Auckland Bioengineering Institute, University of Auckland.

Introduction

The Stimulating Peripheral Activity to Relieve Conditions (SPARC) program is a platform to assist neuroscientists in developing new medical devices.¹ It aims to leverage our understanding of nerve-organ interactions in biological entities and advance existing medical tools. It hosts over a hundred datasets, projects, and resources that are increasing in number, and in the future, there will be a need for a robust tool to explore the expanding content. A targeted data retrieval from the SPARC Portal can boost the researcher-portal interaction experience and help users find the data they seek. However, the search features of the SPARC Portal are limited.

Currently, the search engine of the SPARC Portal does not account for close-matches or misspelt words. The primitive display of the returned results does not emphasise the matched texts and does not allow users to filter or sort the searched data. This prevents users from easily finding their required resources, and once found, users cannot properly narrow or sort the returned data. Moreover, the current description given for each returned result might not necessarily contain the matched keywords which leads to confusion. We have developed an application that we believe will enhance the SPARC Portal search by addressing the above-mentioned issues to reach a FAIR (Findable, Accessible, Interoperable and Reusable) repository to benefit researchers globally.

Advanced QUery Architecture (AQUA) is an application that aims at improving the search capabilities of the SPARC Portal. In particular, it makes the search engine smarter at reading and understanding queries. It also enhances the result display feature of the SPARC Portal by making it more user-friendly and providing users with more sophisticated result filtering and sorting options. Our end goal is to improve the visibility of the SPARC datasets exponentially. This, in turn, will benefit the SPARC community as a whole since their datasets will be more discoverable for reuse and subsequent collaborations.

AQUA was initiated and accomplished during the 2021 SPARC FAIR Codeathon held in July, for a time frame of two weeks. In AQUA, we have incorporated Artificial Intelligence tools to process and refine the queries on the SPARC Portal and implement predictive typing to give feasible suggestions. Thereafter, AQUA auto-corrects the queries to match the existing data on the SPARC portal and the Neuroscience Information Framework Standard (NIFS) Ontology. This will return the most probable datasets that match the search keywords and a list of related new keywords. To enhance the current results display, we have added some functional features to first more precisely filter and sort the results, second emphasise the matched texts for easier skimming, and third, in the case of no available matching results, allow the users to enter their email addresses and get notified when their requested dataset is published.

In this paper, we first review the implementation of AQUA and how its main sectors correlate with the user and the SPARC portal. Next, we provide more details on the sub-sections of each sector and their implemented tools and packages. We mention the added features to the AQUA User Interface (UI) and discuss how it differs from the existing SPARC Portal. Finally, we describe how AQUA can change the search tool on the SPARC Portal and denote the possible future developments to AQUA.

Methods

Implementation

This section discusses the improvement of the search tool on the SPARC Portal. Figure 1 demonstrates how the AQUA UI (also referred to as frontend) and the AQUA server-side data-access layer (also referred to as backend) bridge between the user and the SPARC Knowledge Base. AQUA UI receives the user's queries, formulates them in JSON, and transfers to the AQUA backend module. AQUA backend searches for the formulated queries in the SPARC Knowledge Base. Once the matching datasets/resources are detected, the AQUA backend returns the ranked results to the AQUA UI. Thereafter, the AQUA UI displays the results according to the user's preference of ranking/filtering. The AQUA UI is implemented using the HTML-CSS-JS trio and the main tools utilised for the AQUA backend are Python, Docker, SQLite,² and SciGraph.

Figure 2 depicts the pipeline of AQUA in three major sections:

- **Query refinement:**

1. Auto-completion: Based on the term, our tool automatically completes the queries if it partially/completely matches any keywords. It then sends the selected keyword to AQUA backend.

Figure 1. AQUA workflow.

2. Suggestions: If no exact matches are found, it finds close-matches and suggests them to the users by popping up the phrase: “Showing results for ...”. If the users select to search for their initial query, AQUA will send the raw and uncorrected query to the AQUA backend.
- **Results filtering:**
 1. Sort by: When the results for the query are displayed, user will have the option of sorting them based on the *Relevance*, *Date published*, and *Alphabetical order*.
 2. Filter by: The results can also be filtered based on *Keyword*, *Author*, *Category*, and *Publication date*.
 3. Matched text emphasised: The searched keywords will be emphasised in the returned results.
 - **“Notify me”:** At the end, if no results are returned by the AQUA backend, our tool asks the user if they want to get notified when a related resource is published or not. For a given email address, the tool checks for its validity and then stores it using SQLite. Thereafter, it will check for any updated/uploaded related resource on the SPARC Portal everyday at 2AM EDT. In case of the requested resource availability, it sends a notification email to the registered user.

The AQUA platform integrates Python libraries, data mining tools, a SQL database engine, and Document Object Model (DOM) API to mimic an environment similar to the SPARC Portal with an improved search functionality in multiple ways.

AQUA backend

The AQUA backend includes querying the SPARC Knowledge Base for information, delivering data to the frontend, and processing any logic that the AQUA UI requires. The SPARC Knowledge Base comprises of two references: **SPARC dataset metadata** and **NIFS ontology**. Metadata is the “Data about data”, *i.e.*, additional information provided about datasets. The SPARC dataset metadata includes information such as title, description, techniques, as well as the number of the files, formats, licenses, etc. (**SPARC dataset metadata**), and the NIFS ontology is a set of community ontologies used by SPARC to annotate data and models.

The AQUA backend focuses on two main features: **Query refinement** and **Email notification**. Below, we give a brief introduction to these added features.

- **Query refinement:**

When the initial query term is inserted it goes through two paths: auto-completion (yellow box in Figure 3) and suggestions (purple box in Figure 3).

Figure 2. An overview of the AQUA pipeline. The grey and yellow boxes correspond to the “Query refinement” and “Notify me” modules of the AQUA backend, respectively. The green box corresponds to the “Results filtering” function of the AQUA frontend on displaying the results. The purple boxes illustrate the filters and sorting options.

Figure 3. Query refinement by Auto-completion/Suggestions. The purple box corresponds to the path into returned suggestions and the yellow box corresponds to the auto-completion path. The procedure is demonstrated by an example of inserting a misspelt initial query (*braistem*) into the module.

1. Auto-completion:

The AQUA query refinement module auto-completes the queries after the third inserted letter while the user is typing. The idea of auto-completion is to prevent typos occurring and to give a better user experience in the SPARC Portal. We have created an n-gram model for auto-completion and utilised a Python library *fast-autocomplete*. In spelling correction task, an n-gram is a contiguous sequence of n letters from a given sample of text. An n-gram model is utilised to compare strings and compute the similarity between two words, by counting the number of similar n-grams they share. This technique is language independent. The more similar n-grams between two words exist the more similar they are.³

The Elasticsearch's auto-complete suggester is not fast enough and does not do everything that we need. Consequently, we have utilised the fast-autocomplete library in Python which provides us with a much faster process (reducing the average latency from 900 ms to 30 ms). Elasticsearch's auto-complete suggester does not handle any sort of combination of the words in query terms. For example fast-autocomplete can handle "*brainstem neuron in rat*" when the words "*brainstem*", "*neuron*", "*in*", "*rat*" are separately fed into it, while Elasticsearch's auto-complete needs that whole sentence to be fed to it to show it in auto-complete results.

2. Suggestions:

Simultaneously, AQUA utilises SciGraph for auto-correction and suggestion. SciGraph represents ontologies and ontology-encoded knowledge in a Neo4j graph. However, we found that solely using SciGraph is not sufficient because SciGraph returns alternative queries/suggestions without correcting the initial query. For example, if there is a typo or removed space between the words of a query (*scriptio continua*), SciGraph returns either no results or irrelevant results from the ElasticSearch. Therefore, we have added a new auto-correction feature to segment queries with missing spaces and fix error spelling by creating a pipeline to **SymSpellPy**. SymSpellPy is a Python port of **SymSpell** for spelling correction, fuzzy search and approximate string matching. This improves the performance before sending the request to the ElasticSearch. The auto-correction result is combined with the suggestion results and then executed as the final query search terms. This is demonstrated within the purple box in **Figure 3**.

a. Word segmentation:

Word segmentation divides a string into words by inserting missing spaces at the appropriate positions.

b. Spelling correction:

Supports spelling correction (word splitting/merging) of multi-word input strings in three cases⁴:

1) Extra space inserted into a correct word which leads to two incorrect terms; 2) Removed space between two correct words which leads to one incorrect term; 3) Multiple independent input terms with/without spelling errors.

To read more on AQUA query refinement visit: <https://github.com/SPARC-FAIR-Codeathon/aqua/blob/main/Documentation/QueryRefinement.md>.

• Email notification

The primary purpose of this module is to notify users whenever a new dataset is published matching their search terms. However, users can still use the same function to receive a summary table including basic information and links to all datasets currently matching their keywords. Additionally, as the “Notify me” module saves the requests in a database, this information can be further accessed and analysed to improve the content (**Figure 4**).

Figure 4. The pipeline of the AQUA “Notify me” module.

We can summarize the “Notify me” actions as follow:

1. Adds email requests with keywords;
2. Scans for existing search hits and sends email;
3. Moves the pending requests to a waiting list that is scanned daily;
4. Moves the fulfilled requests to an archive;
5. Any failed requests (that already have matching hits) will remain on the waiting list for one month, during which the “Notify me” module will try to send the email daily. Afterwards, if the email still fails, it will be moved to the archive with a “failed” status.

To read more visit: <https://github.com/SPARC-FAIR-Codeathon/aqua/blob/main/Documentation/NotifyMe.md>.

AQUA UI

AQUA UI receives the user’s queries, formulates them, and transfers to the AQUA backend module. When the response from the AQUA backend is received, the AQUA UI interprets it and displays the content on the screen. Like the SPARC Portal web application, the AQUA UI is implemented using **VueJS** and **NuxtJS**. Nuxt is an upper-level framework that is built over Vue.js to design and create highly advanced web applications.⁵ The AQUA UI displays the customised list of results with the emphasised searched keywords.

Operation

To start the application follow the steps in **Installation**.

How to use the features added by AQUA to the SPARC Portal search engine?

The application works like other similar search engines with a user interface mimicking the SPARC Portal environment.

1. Predictive search typing:

AQUA provides auto-completion for user’s queries as they type. This feature is powered by SciGraph and training data from the SPARC Knowledge Base. AQUA only shows auto-completion after users type three letters or more to avoid too many results being returned, slowing down the application.

2. Advanced search options:

By expanding the “Advanced search” tab under the search box, users can select whether AQUA searches for **Exact match** for their query or **Any of the words**. The default is **Any of the words** match.

3. Advanced sorting:

The existing SPARC Portal allows sorting based on dataset titles (alphabetically) and by published date. AQUA adds a “Relevance” sorting criterion that returns results based on how relevant the results are to their search query. This is set as the default sorting option.

4. Advanced filtering:

The existing SPARC Portal only allows for filtering based on “Dataset status”, which is either Published or Embargoed. AQUA adds more sophisticated filtering options. Users can filter datasets by one or several keywords, authors, and categories. Hit “Enter” after each “Keyword”, “Author”, or “Category” in their respective box to register it. After the entries are registered, click “Apply” to filter dataset results.

5. Email notifications for new matched datasets:

Users can opt in to receive emails about new datasets that match their search query. We believe this is a much needed option for users to stay updated about their search and SPARC datasets. Simply click on “Create alerts” under the search box and enter an email. AQUA will trigger an email send when newly added dataset(s) that match the search query are published by SPARC. This is a one-time-only email subscription.

6. Emphasise matched texts in result display:

When a dataset is returned, any matched text in the dataset title and description will be emphasised for easy and convenient lookup.

Use case

We conducted experiments to compare the performance of the AQUA query refinement module by either deploying SciGraph or fast-autocomplete. We analysed the operation in auto-completing the queries in terms of performance and execution time. We compared these two criteria in two scenarios: correct queries, and queries with one typo. Our experiment revealed that fast-autocomplete returns more completions than SciGraph in both cases of inserting correct queries and queries with typo. Also, fast-autocomplete returned the results 24 times faster in correct queries and 11 times faster in queries with typos.

We tested the performance of the AQUA spelling correction module and compared the results with the SPARC’s Elasticsearch. To do this, we randomly selected 22 sets of queries from the SPARC dataset, each containing fifty keywords or phrases. The queries were then modified to include different types of typos (deletion, insertion, replacement). We calculated the Mean Average Precision (MAP) of AQUA and the SPARC’s Elasticsearch in spelling correction. Results showed that as the number of terms in a query increases, the performance of AQUA noticeably surpasses the SPARC’s Elasticsearch (Table 1). Same steps were taken on querying the name of author/authors as keywords for 9 test collections.

Table 1. Mean Average Precision (MAP) of AQUA and the SPARC’s Elasticsearch over 22 test collections consisting of biological keywords as queries.

Typo	1 term		2 terms		3 terms	
	AQUA	ES	AQUA	ES	AQUA	ES
0 typo	0.714785	0.711452	0.569673	0.569673	0.680431	0.677097
1 del	0.635935	0.677184	0.555371	0.505849	0.668609	0.653644
1 insert	0.704785	0.742356	0.56559	0.572663	0.680431	0.661312
1 replace	0.644126	0.772202	0.548968	0.568364	0.680431	0.646185
no space	NaN	NaN	0.568006	0.987667	0.667097	0.816667
no space 1 typo	NaN	NaN	0.559696	0.995918	0.670508	0.056122
no space 2 typo	NaN	NaN	0.484005	0.056667	0.644305	0.010204
no space 3 typo	NaN	NaN	0.446296	0.184211	0.589903	0.003472
3 typo	NaN	NaN	0.540761	0.481212	0.646919	0.621238

Table 2. Mean Average Precision (MAP) of AQUA and the SPARC’s Elasticsearch over 9 test collections consisting of authors as queries.

Typo	1 term		2 terms	
	AQUA	ES	AQUA	ES
0 typo	0.863212	0.897673	0.926911	0.952778
1 del	0.613025	0.675974	0.818579	0.797889
1 insert	0.843871	0.914193	0.926944	0.96
1 replace	0.822374	0.867786	0.913039	0.913265
no space	NaN	NaN	0.926911	0.1245

Table 2 shows that AQUA performs better in correcting misspellings that appear in a two-term “author” query. A significant performance difference is AQUA’s ability to fix “author” as a query that loses space where AQUA’s MAP is 0.92 while the SPARC’s Elasticsearch’s MAP is only 0.12.

The experiment results and description are available [here](#). The code for running the experiments and the data are also available on: <https://github.com/SPARC-FAIR-Codeathon/aqua/tree/main/experiment>.

Conclusions and next steps

This paper demonstrated how the SPARC Portal could be more FAIR by improving its search feature through AQUA. Since the first contact between researchers and a repository of datasets/models/resources is through the website’s search engine, we enhanced the search system’s functionality and the user interface. In AQUA, we deployed multiple tools and packages to make querying the data more precise, convenient, and effective.

We propose to add a *view type* to the existing SPARC Portal to enhance the users’ experience with the website. The SPARC Portal’s existing view type is “List”. AQUA proposes to add a “Gallery” view option in the future. Also, we plan to add a new discovering feature to the SPARC Portal to find resources by querying snapshots of simulations. This can be done by segmenting the simulation results into smaller time intervals or any chunk of data. Currently, the AQUA “Notify me” feature is a one-time-only email notification. Options to be alerted more than once can also be added in the future. AQUA can also enhance the SPARC search engine further by improving user’s next query. This will be done by developing a session-based search based on user’s search or clickthrough history on the Portal. The feature will create a personalized experience for users and thus enhance their overall experience with the SPARC Portal.

Software availability

Source code available from: <https://github.com/SPARC-FAIR-Codeathon/aqua/blob/main/LICENSE>

Archived source code as at time of publication: <https://doi.org/10.5281/zenodo.5352470>.⁶

License: MIT

The AQUA application can be installed and run by cloning the [main Github repository](#) and following the command line instructions. Instructions on how to clone a Github repository can be found [here](#).

Acknowledgements

We would like to extend our special thanks to the NIH Common Fund’s SPARC Program and to the organisers of the 2021 SPARC FAIR Codeathon for their support during the planning and development of this project.

References

1. The Sparc Data and Resource Center: 2021. [Reference Source](#)
2. Bhosale ST, Patil T, Patil P: **SQLite: Light Database System**. *Int J Computer Sci Mobile Computing*. April 2015; **4**(4): 882–885.
3. Ahmed F, De Luca EW, Nurnberger N: **Revised n-gram based automatic spelling correction tool to improve retrieval effectiveness**. *Polibits*. December 2009; **40**(40): 39–48.
4. symspellpy api. [Reference Source](#)
5. Nuxt.js and Vue.js: **Reasons why they differ and when do they combine**. 2021. [Reference Source](#)
6. Shahidi N, Ngo T, Irasmy Y, et al.: **Nilooofar-Sh/aqua: First release of AQUA (v1.0.0)**. *Zenodo*. 2021. [Publisher Full Text](#)

Open Peer Review

Current Peer Review Status:

Version 1

Reviewer Report 26 July 2022

<https://doi.org/10.5256/f1000research.76636.r141807>

© 2022 Martone M. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Maryann E. Martone

Department of Neurosciences, Center for Research in Biological Systems, University of California, San Diego, San Diego, California, USA

The authors describe a query tool they developed for the SPARC Portal during a code-a-thon held in July 2021. The authors correctly identified several shortcomings of the portal search at that time, and created a service that would address them. However, there are a few issues that limit the utility of this article:

1. The main issue is that, as often happens, the SPARC search interface has evolved since that time, so many of the contentions are no longer true. SPARC has moved to a faceted search interface that goes well beyond the filters employed at that time and here, e.g., anatomical structure, technique, species, sex. The new search functionality uses an Algolia index which could support many of the features that you have developed (although I don't think it is completely open, so that is a minus). So I think for this to be useful beyond describing the technology used, you would have to compare AQUA to the current SPARC interface/services rather than the state in 2021. While I know that it may be impractical to redo all the tests etc with updated SPARC, I would like the author to at least acknowledge the update and address these issues in the discussion.
2. The authors lay out several features, e.g., autocomplete, spell checking and e-mail notification. These clearly would be useful, but I don't think that the authors provided evidence in the form of user testing that they *are* useful, that is, that they give better search results for an average SPARC user. They refer readers to the results in GitHub, but I'd like to see some concrete examples and user feedback. I know there is a Docker image available, but that is not practical for those with domain expertise to test. What are the authors plans to make a version of their interface available through SPARC? e.g., in the Tool and Resource section? I didn't see it listed there.
3. The authors don't discuss the generalizability of their approach. Would their code have use beyond the SPARC portal?

Minor issues:

1. When the authors refer to "keywords", are they specifically referring to the metadata field

marked "keywords"? What about the other standard metadata that a SPARC acquires and tags in the JSON metadata file?

2. Adding a reference to the SPARC SDS specification would provide readers with a better idea of the metadata available. Bandrowski et al., 2021¹.
3. NIFS should be NIFSTD. A reference for the NIFSTD ontology is Bug et al. 2008².
4. The table/figure legends are not adequate. Terms are introduced, e.g., SciGraph in Fig 1, that are not explained either in the legend or in the text. Why are some values bolded in the tables? A reference for SciGraph is Surles-Zeigler et al., 2022³ (currently accepted for publication in *Frontiers in Neuroinformatics*).

References

1. Bandrowski A, Grethe J, Pilko A, Gillespie T, et al.: SPARC Data Structure: Rationale and Design of a FAIR Standard for Biomedical Research Data. *bioRxiv*. 2021. [Publisher Full Text](#)
2. Bug WJ, Ascoli GA, Grethe JS, Gupta A, et al.: The NIFSTD and BIRNLex vocabularies: building comprehensive ontologies for neuroscience. *Neuroinformatics*. 2008; **6** (3): 175-94 [PubMed Abstract](#) | [Publisher Full Text](#)
3. Surles-Zeigler M, Sincomb T, Gillespie T, de Bono B, et al.: Extending and using anatomical vocabularies in the Stimulating Peripheral Activity to Relieve Conditions (SPARC) program. *bioRxiv*. 2021. [Publisher Full Text](#)

Is the rationale for developing the new software tool clearly explained?

Partly

Is the description of the software tool technically sound?

Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

No

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

No

Competing Interests: I am one of the PIs of the SPARC Data and Resource Center. This work concerns the SPARC project.

Reviewer Expertise: Neuroinformatics. I also am a PI in the SPARC Data and Resource Center so

know the SPARC project very well.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 01 November 2021

<https://doi.org/10.5256/f1000research.76636.r94519>

© 2021 Rajagopal V. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Vijay Rajagopal

Department of Biomedical Engineering, University of Melbourne, Melbourne, VIC, Australia

The authors present a search and retrieve tool for the SPARC knowledge database. Overall, the contribution is important and in line with making research data FAIR. The article is also written well but is missing some key components that would make testing and adoption of this tool in SPARC easy to do.

- Installation already assumes installation of yarn, and I was unable to easily install it. Please include details of popular alternate installation methods (like docker) within the instruction manual.
- Related to the above, what are the minimum required libraries that one needs to build and install to their software to reproduce their results?
- Tables 1 and 2 need to be more informative. What is the formula for mean average precision? Ideally, they should provide other metrics as well. Perhaps even the distribution of precision for the test collections.
- The authors suggest that AQUA surpasses Elasticsearch as number of queries increases in Table 1. Looking at column "3 terms", however, I see that ES has a MAP of ~0.8 vs AQUA's ~0.6 when there is no space. Therefore the claim that AQUA is superior is to Elasticsearch not clear to me.
- In Table 1, what do the NaNs mean in the column "1 term"? I can see that "no space" does not apply in this case. If this is a typo, it should be resolved. In these cases is a NaN an error produced by both ES (Elasticsearch) and AQUA?
- In Table 2, the "2 term" column shows that the MAP is not really that different between ES and AQUA, except for "no space". The table does not reflect the significant improvements by using AQUA. I suggest including more comparison metrics to make the case for the performance of AQUA.

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

No

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

No

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: computational physiology, mechanobiology, systems biology, image analysis, bioengineering, heart, breast

I confirm that I have read this submission and believe that I have an appropriate level of expertise to state that I do not consider it to be of an acceptable scientific standard, for reasons outlined above.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research