ASH: A New Tool for Automated and Full-Text Search in Systematic Literature Reviews

Marek Sośnicki(✉) and Lech Madeyski

Department of Applied Informatics, Wroclaw University of Science and Technology, Wyb.Wyspianskiego 27, 50-370 Wroclaw, Poland

Abstract. Context: Although there are many tools for performing Systematic Literature Reviews (SLRs), none allows searching for articles using their full text across multiple digital libraries. **Goal**: This study aimed to show that searching the full text of articles is important for SLRs, and to provide a way to perform such searches in an automated and unified way. **Method**: The authors created a tool that allows users to download the full text of articles and perform a full-text search. **Results**: The tool, named ASH, provides a meta-search interface that allows users to obtain much higher search completeness, unifies the search process across all digital libraries, and can overcome the limitations of individual search engines. We use a practical example to identify the potential value of the tool and the limitations of some of the existing digital library search facilities. **Conclusions**: Our example confirms both that it is important to create such tools and how they can potentially improve the SLR search process. Although the tool does not support all stages of SLR, our example confirms its value for supporting the SLR search process.

Keywords: Systematic literature review · Systematic review · SLR · Knowledge synthesis · Automated search · Meta-search

1 Introduction

Systematic literature review (SLR) is a commonly and increasingly used process to systematically analyze all research related to some specific area. One of the most difficult parts of preparing SLR is the search and selection phase, where researchers must gather a base set of papers to be analyzed. It is necessary to obtain very high levels of search completeness—which means getting possibly all available literature on the specific topic of interest. Researchers have developed several approaches which aid in this task. The basic method is an automated search, which is involves running search queries across different search engines [4]. However, with the increasing amount of research, it gets harder to perform a complete search of potentially relevant computer science and software engineering articles. The SLR process recommends using multiple search engines, however, each search engine has slightly different features, which may bias the search process and increase the amount of work needed in order to perform a comprehensive search [1].

As SLRs are effort-intensive, researchers have developed different tools to support the process [12]. Such tools can assist in the SLR process by automating some phases or by providing a unified interface for navigation across different literature. Unfortunately, although the concept of meta-search tools is understood [17], there is currently no tool that would gather data from all important software engineering search engines and provide a unified interface to facilitate the search and selection phase [13].

Hence, the main contribution of this article is an Automated Search Helper (ASH)—a tool which automates the process of downloading articles and allows the user to perform an initial search and selection phase for the SLR process. ASH takes a list of paper identifiers, downloads their bibliographic information along with their full text, and saves the data in a unified format, allowing users to run searches on the collected data. Additionally, the tool's interface presents papers in a way that accelerates the decision making in the selection process for SLR. In this paper, we present the first version of the application, which focuses on the download and unification of the article data. This shows the potential of the tool, and is intended to encourage future users to use it in their SLR routine. Currently, ASH supports six main digital libraries that catalog Computer Science research.

This article is divided into the following sections: Sect. 2 presents the current state of the art, Sect. 3 explains the motivation to create such a tool along with an example problem, Sect. 4 shows the workflow of the tool, Sect. 5 explains how ASH was used to address the example problem, and Sect. 6 draws conclusions and proposes future improvements.

2 State-of-the-Art

SLRs were introduced to software engineering by Kitchencham et al. [6,10]. The SLR process consists of three main phases: planning, conducting, and reporting. In the planning phase, researchers identify the need for a review and prepare a protocol that contains research questions, search strategy, study selection criteria and procedures, quality assessment checklists, data extraction and synthesis strategy, and the project's timetable. In the second phase, researchers conduct the review according to the protocol. The first part of this phase is a search for all studies, next, there are one or more selection phases, e.g., initially, studies are excluded based on a title, abstract, and keywords and only the remaining ones have their full text analyzed. The next parts are the quality assessment of the selected studies and data extraction and analysis. The final phase is about reporting the review which is preparing a paper or technical report [2].

2.1 Search Process

One of the most difficult and critical parts of the review is the search for, and the selection of, primary studies. The search process must be defined in a way that provides the highest possible level completeness, and should also support the reproducibility of the research process. Researchers have developed many techniques that can help to achieve it. The first and most commonly used technique to gather research is an automated search. The main source of studies for the search are digital libraries. They allow researchers to find all articles related to the topic using their search engines. A well-prepared search string may allow users to obtain most of the research on a given topic, on the other hand, a poorly-constructed string may miss relevant papers or provide a large number of irrelevant articles. Preparing an effective search string is not an easy task and it requires extensive knowledge of the topic and often requires considerable iteration to get the best possible outcome [4]. Digital libraries differ in their query languages, so researchers usually have to prepare different strings for different libraries [7]. Moreover, some digital libraries do not allow for some exclusions to be expressed in a search string, or do not allow searches related to the full text of the paper [9] which can make it difficult to find all relevant studies [1].

2.2 Current Search Problems

Al Zubidy et al. [1] identified five main groups of problems: problems with search strings—building, manipulation, etc., problems with digital libraries front-end and back-end functionalities, lack of tool support, and problems related to researchers. Some of those problems can be solved by researchers, others can be solved with tools, but many can only be solved by digital library providers.

The main source of problems is the lack of support for performing SLRs provided by the digital libraries. Each digital library works differently, the search strings are built with slightly different syntax, results are returned in a different format. Each digital library allows for different search exclusions (e.g., limits to the scope of the search). Some libraries do not allow to search in full text, and sometimes the full text is missing or cannot be downloaded. Moreover, when the search is performed many duplicates and irrelevant results are returned—especially when the search is done on multiple databases [1]. Even if the above problems are mitigated, some issues can be efficiently solved with proper tools. Although many stages of performing SLR can be automated (sometimes only partially), the search and selection process often requires substantial manual effort. Manual search and selection are prone to human errors, which can reduce the validity of the process. However, with proper tooling, the risk of misclassification of articles can be reduced and the whole process can be accelerated.

2.3 Tools Supporting SLR Process

Many tools designed for SLRs were designed to support medical reviews [15], although some studies summarized existing tool support for SLRs in software engineering [1,12,13].

One of the most recent tools for SLR is Thoth [11] that tries to facilitate all phases of SLR. The developers' goal was to allow the user to perform the whole SLR process using their tool, but it focuses mainly on the selection phase. It allows user to perform automatic searches but only in Scopus. Other recent tools that support all phases of SLR are SLuRp [3] and Buhos [15]. Both have multiple useful features, but their main focus is to improve communication between reviewers working on the same SLR project.

Some researchers are content to use multipurpose tools like Excel or Jabref [1] in their SLRs. Such tools allow data storage (both of citation information and extracted data), but they were not explicitly designed to support the SLR process, thus, their interfaces and facility do not properly support the aggregation and organization of results from all SLR phases. Al. Zubidy [1] analyzed also other tools which are used for SLRs including, e.g., SLRTool [5], SESRA [14]. Each of them has multiple useful features, but none has all the features desired by reviewers [1].

There currently is no tool that would fully automate the whole search and selection process. Some tools try to apply some automation (e.g., SLuRp can perform semi-automated downloads of articles), but their main focus is on later stages of SLR—which do not depend on external factors such as digital library facilities.

3 Motivation and Significance

In this section, we present a motivational example based on a problem we encountered in our research. The problem occurred when we were undertaking an SLR about mutation testing in C++ programming language. The goal was to gather all papers related to mutation testing in which authors used C++—either to create a tool for mutation testing, or to review the quality of software in this language.

The problem appeared during the search phase. First, we prepared a list of articles which would be considered as a checklist for validation of the search [8]. Due to a large number of articles in this area, we decided that the search phrase would be performed as an automated search in most well-known Computer Science related digital libraries . The following query was used to find all relevant articles: `("mutation testing" OR "mutation analysis" OR "mutant analysis") AND "C++"`. If it was possible for a specific digital library, we limited the search to the Computer Science area. In Scopus, this query produces about 80 results. Unfortunately, it is the only major search engine that can perform a search of the phrase "C++", other libraries omit "++" which produces hundreds of results most of them relating to "C" instead of "C++". Moreover, even this large set of articles did not cover all the articles from the checklist. The reason for this was that some articles (e.g., [16]) provided information about the programming language used for mutation testing within the full article text, e.g., in sections about the evaluation of new approaches.

In this example, we had two options. Firstly, we could modify the queries to include searching the full text of a paper. Unfortunately, some sources (including Scopus) do not offer full-text search. Secondly, if full-text search was not available in a given digital library, we could use the following query: `"mutation testing" OR "mutation analysis" OR "mutant analysis"` and then search the full text of the identified articles manually. The second approach yields thousands of results, where more than 95% are irrelevant. Assessment of all the papers would require substantial manual effort. It would mean downloading and opening the text of the full article, searching for usages of "C++" phrase, analyzing the context of its use, and finally accepting or rejecting the article. Having such large numbers of papers, it would be easy to misclassify some of them which would undermine the goal of maximising search completeness. Additionally, there is no consistent standard across digital libraries—as mentioned above, different queries would be used for different libraries which is not a desirable situation [1].

4 Tool

To solve the problem stated in previous section a tool (Automated Search Helper) was created. ASH downloads articles from across multiple digital libraries, extracts the text from them, transform them into a common format and removes duplicates among them. Moreover, ASH provides a way to search inside full text of the downloaded articles. Results of queries are displayed as web pages, served by the web server included in the tool, they allow user to quickly analyze the query hits.

ASH is still under development, but current version is already functional and can be efficiently used in the SLR process. The tool is an application written in Python and was tested on Linux, Mac, and Windows devices, but it can run in any system that can fulfil the requirements described in the documentation [18]. Current version of application supports papers from the following digital libraries: IEEE, ACM, Science Direct, Springer, Wiley and Scopus. More digital libraries will be supported in the future. The tool can obtain full text of the article directly from the publisher website or, using optical character recognition, parse the full text from PDF of the paper. Detailed description of ASH architecture and usage manual can be found in the ASH Documentation [18].

ASH fulfils many of the SLR tool requirements described in [1], but because the solution already integrates the search from across the different libraries, its possible to add more features in order to satisfy other requirements and overcome the barriers Al-Zubidy et al. mentioned. The tool proposed here covers only the search phase of SLR, it is not intended to support the later stages of SLR, for which there are already useful tools available, e.g., Buhos [15]. This means that our tool allows a user to easily generate the input required by others tools that can be used to support the rest of the SLR process.

5 Evaluation

The tool was evaluated on the problem described in motivation, see Sect. 3. The tool was tasked to collect all the papers related to "Mutation testing" and in the obtained set to find all papers that mentioned "Mutation testing in C++". ASH managed to successfully download full text of more then 95% of articles from the search results of "Mutation testing" across all supported digital libraries (for all other references only the abstract with bibliographic information was downloaded). It managed to remove the duplicates among the papers (about 20% of all results from all digital libraries). This means that, even though some articles full text could not be downloaded, the set of papers which was obtained by the tool as a base for further search was not worse than the one provided by digital libraries.

The tool managed to find all the relevant articles, even these which did not mention "C++" in title, abstract or keywords. For most of the digital libraries, the search results of "Mutation testing in C++" were close to the results obtained by digital libraries search engine. However, for a few of the digital libraries, ASH obtained much better results, either by providing more relevant results due to searching inside full text, or by filtering the false positive search results (e.g., when digital library search engine treated "C++" as "C").

The detailed results and description of the evaluation can be found in Evaluation section (https://github.com/LechMadeyski/AutomatedSearchHelper/wiki/Evaluation) of the ASH documentation [18] due to the imposed paper length limit. The results are very promising, although ASH does not support all digital libraries yet, it can already be used to reduce the amount of manual work during the SLR process.

6 Conclusions

In this article, we explained how important it can be to analyze the full text of articles during the initial search in the SLR process. Our contribution is a tool (ASH) that automatically downloads and searches the full text of articles, allowing users to obtain higher levels of completeness for their SLR searches. The tool does not yet support all digital libraries, but we demonstrated, that even without this, ASH is already able to obtain the full text of about 90% articles, for searches related to Computer Science topics. Moreover, the results obtained by the tool can be much more accurate for specific research questions than standard approaches.

The tool is available for use, and further features of tool are currently under development. We aim to increase the number of supported Digital Libraries, including also ones not related to Computer Science. Additionally, the user interface of the tool can be improved depending on the future user needs and the feedback received in further testing stages.

Acknowledgements. The authors thank Prof. Barbara Kitchenham for reviewing this paper before its submission.

References

1. Al-Zubidy, A., Carver, J.C.: Identification and prioritization of SLR search tool requirements: an SLR and a survey. Empir. Softw. Eng. **24**(1), 139–169 (2019)
2. BA, K., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical report (2007), version 2.3
3. Bowes, D., Hall, T., Beecham, S.: SLuRp: a tool to help large complex systematic literature reviews deliver valid and rigorous results. In: Proceedings of the 2nd International Workshop on Evidential Assessment of Software Technologies, pp. 33–36. EAST 2012. ACM, New York, NY, USA (2012)
4. Dieste, O., Grimán, A., Juristo, N.: Developing search strategies for detecting relevant experiments. Empir. Softw. Eng. **14**(5), 513–539 (2008)
5. Fernández-Sáez, A.M., Genero Bocco, M., Romero, F.P.: SLR-TOOL - a tool for performing systematic literature reviews. In: Proceedings of the 5th International Conference on Software and Data Technologies, vol. 2: ICSOFT, pp. 157–166. INSTICC, SciTePress (2010). https://doi.org/10.5220/0003003601570166
6. Kitchenham, B.: Procedures for performing systematic reviews. Technical report (08 2004)
7. Kitchenham, B., Brereton, P., Budgen, D.: The educational value of mapping studies of software engineering literature. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, vol. 1, pp. 589–598. ACM, New York, NY, USA (2010)
8. Kitchenham, B., Budgen, D., Brereton, P.: Evidence-Based Software Engineering and Systematic Reviews. CRC Press, Boca Raton (2016)
9. Kitchenham, B.A., et al.: Refining the systematic literature review process-two participant-observer case studies. Empir. Softw. Eng. **15**(6), 618–653 (2010)
10. Kitchenham, B.A., Dybå, T., Jørgensen, M.: Evidence-based software engineering. In: ICSE 2004: International Conference on Software Engineering, pp. 273–281 (2004)
11. Marchezan, L., Bolfe, G., Rodrigues, E., Bernardino, M., Basso, F.P.: Thoth: a web-based tool to support systematic reviews. In: ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). pp. 1–6 (2019)
12. Marshall, C., Brereton, P.: Tools to support systematic literature reviews in software engineering: a mapping study. In: ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 296–299 (2013)
13. Marshall, C., Kitchenham, B., Brereton, P.: Tool features to support systematic reviews in software engineering - a cross domain study. e-Informatica Softw. Eng. J. **12**(1), 79–115 (2018). https://doi.org/10.5277/e-Inf180104
14. Molléri, J.S., Benitti, F.B.V.: SESRA: a web-based automated tool to support the systematic literature review process. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering. EASE 2015, ACM, New York, NY, USA (2015)
15. Navarrete, C.B., Malverde, M.G.M., Lagos, P.S., Mujica, A.D.: Buhos: a web-based systematic literature review management software. SoftwareX **7**, 360–372 (2018)

16. Petrović, G., Ivanković, M.: State of mutation testing at Google. In: Proceedings of the 40th International Conference on Software Engineering Software Engineering in Practice - ICSE-SEIP 2018. ACM Press (2018)
17. Ramampiaro, H., Cruzes, D., Conradi, R., Mendona, M.: Supporting evidence-based software engineering with collaborative information retrieval. In: 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2010), pp. 1–5 (2010). https://doi.org/10.4108/icst.collaboratecom.2010.9
18. Sośnicki, M.: (2020). https://github.com/LechMadeyski/AutomatedSearchHelper/wiki