A Prototype "Debugger" for Search Strategies

JINGFAN ZANG, Goldsmiths, University of London, UK
TONY RUSSELL-ROSE, Goldsmiths, University of London, UK

Knowledge workers such as healthcare information professionals, legal researchers, and librarians need to create and execute search strategies that are effective, efficient and error-free. The traditional solution is to use command-line query builders offered by proprietary database vendors. However, these are based on an archaic approach that offers limited support for the validation and optimisation of their output. Consequently, there are often errors in search strategies reported in the literature that prevent them from being effectively reused or extended. In this paper, we demonstrate a new approach that takes inspiration from software development practice and applies it to the challenge of search strategy formulation. We demonstrate a prototype 'debugger' which provides insight into the construction and semantics of search strategies, allowing users to inspect, understand and validate their behaviour and effects. This has the potential to eliminate many sources of error and offers new ways to validate, optimise and re-use search strategies and best practices.

CCS Concepts: • **Information systems** → **Search interfaces**; • **Computing methodologies** → *Representation of Boolean functions*; • **Software and its engineering** → Software testing and debugging.

Additional Key Words and Phrases: systematic searching, search strategies, query formulation, visualisation, debugging, optimisation

ACM Reference Format:
Jingfan Zang and Tony Russell-Rose. 2023. A Prototype "Debugger" for Search Strategies. In *ACM SIGIR Conference on Human Information Interaction and Retrieval (CHIIR '23), March 19–23, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3576840.3578321

1 INTRODUCTION

Progress in science is predicated on the recognition and exploitation of prior work. In life sciences in particular, it is vitally important that all relevant sources of evidence be considered when developing policy, guidance and interventions. Systematic literature reviews can play a key role in this by synthesising the complex, incomplete and at times conflicting findings of scientific research into a form that can readily inform decision making.

However, systematic reviews can take years to complete [4] and new research findings may be published in the interim, leading to a lack of currency and potential for inaccuracy [12]. Moreover, there are often mistakes in search strategies they rely on, and the key steps of validating, reviewing, and translating search strategies are often fragmented across a variety of unconnected, non-interoperable platforms [7]. Most researchers recognise the need for a robust approach to evidence synthesis. However, significant barriers remain. First, many researchers struggle with the conceptual process of analysing a research question, breaking it down into a structured set of information needs and then transforming that into a systematic search strategy [5]. Second, although the fundamental concepts of Boolean logic are relatively simple, the process by which they must be translated into the syntax required by traditional databases can be obfuscated, inefficient, and error-prone [9, 10]. Consequently, even with the support of trained information professionals, many

Manuscript submitted to ACM

```
1. randomized controlled trial.pt.
2. controlled clinical trial.pt.
3. randomized.ab.
4. placebo.ab.
5. clinical trials as topic.sh.
6. randomly.ab.
7. trial.ti.
8. 1 or 2 or 3 or 4 or 5 or 6 or 7
9.   (animals   not   (humans   and
animals)).sh.
10. 8 not 9
11. exp Child/
12. ADOLESCENT/
13. exp infant/
14. child hospitalized/
15. adolescent hospitalized/
16.  (child$ or infant$ or toddler$
or adolescen$ or teenage$).tw.
17. or/11-16
18. Child Nutrition Sciences/
19. exp Dietary Proteins/
20. Dietary Supplements/
21. Dietetics/
22. or/18-21
23. exp Infant, Newborn/
24. exp Overweight/
25. exp Eating Disorders/
26. Athletes/
27. exp Sports/
28. exp Pregnancy/
29. exp Viruses/
30.  (newborn$ or obes$ or "eating
disorder$" or pregnan$ or childbirth
or virus$ or influenza).tw.
31. or/23-30
32. 10 and 17 and 22
33. 32 not 31
```

Fig. 1. A multi-line search strategy [2]

researchers have difficulty in developing the efficient and effective search strategies that are essential to robust scientific practice.

To address this problem, we are developing a prototype 'debugger' which provides insight into the construction and semantics of search strategies, allowing users to run, debug and optimise them using visual tools and debugging aids. This has the potential to eliminate many sources of error, allowing users to pause and interrogate them at various stages, and offers new ways for them to be optimised, extended, and re-used.

1.1 Background and rationale

The process of evidence synthesis relies on painstaking and meticulous searching of multiple literature sources. These include published literature sources and other specialist databases and 'grey literature' (non-peer reviewed sources). The principal way in which such sources are interrogated is through the use of Boolean queries, which utilise a variety of keywords, operators and ontology terms. Systematic search strategies are typically built incrementally on a 'line by line' basis, resulting in an output such as that shown in Figure 1. Search strategies may also be expressed using a 'block based' approach which represents the combined semantics as a single composite query string [5].

The choice of search strategy is critical in ensuring that the process is not biased by easily accessible studies [9], and that it is transparent and repeatable. However, there are often mistakes in search strategies reported in the literature: in one sample of 63 published search strategies, at least one error was detected in 90% of these, including spelling errors, truncation errors, logical operator error, incorrect query line references, redundancy without rationale, and more [9, 10]. Evidently, despite the dedication and painstaking attention to detail of many individuals, the current process for creating effective search strategies is error-prone and inefficient.

1.2 Design challenge

The development of a 'debugger' for search strategies could offer clear benefit in addressing many of the sources of error and inefficiency outlined above. However, although such a concept sounds a relatively simple undertaking in principle, the practice presents a number of challenges:

- Systematic searching requires the interrogation of multiple literature databases, each with its own proprietary syntax. The absence of a 'universal syntax' for search means that any debugger may have to cope with an indefinite set of arbitrary syntaxes.
- Most proprietary DB syntaxes are formulated using a combination of declarative and imperative constructions, so 'running' a search strategy is a somewhat imprecise concept since they do not follow to the deterministic semantics expected of most programming languages
- Software development is typically predicated on meeting unambiguous objectives, such as passing unit or integration tests. By contrast, measuring the quality of a search strategy is usually a more subjective undertaking
- A key component of modern programming languages is abstraction, whereby lower-level details can be subsumed or hidden within higher-level structures. By contrast, literature search syntaxes offer no such facility so the process of debugging must invent and adopt new conventions for managing complexity
- Software debuggers are designed for professionals who are trained in the principles and practices of software development. By contrast, search strategies are created and maintained by information professionals who may understand the principles of optimisation and validation but have limited interest or experience in the practicalities of software development or debugging
- The process of debugging needs to adopt an appropriate conceptual model to present the task in such a way as the benefits of fine-grained interrogation and visualisation become apparent in a manner which is accessible to information professionals

In the following sections, we explore solutions to the challenges presented above. We do so in the form of a set of design concepts that has been implemented as an interactive prototype. We report on an initial evaluation of this prototype, discuss further ideas and suggestions for extending the work, and review the prospects for building a functional implementation of the prototype.

2 RELATED WORK

The idea of a debugger for search strategies may seem self-evident in principle, but in practice remains a relatively untried and untested concept and prior art is thus limited. Notable exceptions to this include searchrefiner [11] which is an open source tool to assist in formulating, visualising, and understanding Boolean queries, and the Systematic Review Accelerator [1]. The former allows researchers to visualise why specific queries retrieve particular citations, and helps users understand how to refine queries into more effective ones. The latter is a suite of tools designed to improve the efficiency and effectiveness of systematic reviews and includes searchrefiner.

More broadly, there is a substantial literature on the science and practice of debugging, but none to date has specifically applied this to the challenge of systematic searching. Likewise, there is a collection of prior art focused on the application of data visualisation to the broader task of search-query formulation (summarised in [8]) and a body of work focused on improving queries in systematic reviews (summarised in [11]), but this has yet to explicitly address the task of debugging as described herein.

3 DESIGN CONCEPT

3.1 Application architecture

One of the challenges of developing a debugger for search strategies is the multiplicity of formalisms - not just in terms of database syntaxes, but also in the ways they can be rendered, e.g. as multi-line or block-based variations. This suggests that one fruitful way of thinking about the design space is from a Model-View-Controller (MVC) perspective, to allow separation of the internal representations of information from the ways in which it is presented to the user [3].

The design challenge then becomes one of selecting the optimal combination of views to communicate the search logic so that it can be effectively understood, analysed and (if necessary) debugged. Since most information professionals have been trained using traditional 'string-based' representations, we assume that these will remain integral to the debugging experience (at least in the short-term). However, the scope for using visualisation and other interactive devices to enhance and communicate key aspects of search logic is considerable, and we aim to explore and exploit that in this work [6].

The overall application layout consists of a traditional (string-based) view on the right and a blank canvas for various visualisations on the left. The user has the option to interact with the search strategy using a variety of 'player' controls (at the bottom of the right hand panel). Selecting the 'play' button has the effect of running the search strategy, which executes line by line. This is illustrated simultaneously in both views, via a graphical representation on the left, and a moving highlight on the right as shown in Figure 2. The dynamic nature of this interaction (and those that follow) can be seen in the YouTube demo: https://youtu.be/itG3bYD790w.

3.2 Coding view

The right hand pane contains a string-based view known as the 'coding view'. String-based representations offer no mechanism for making explicit the logical structure that is implicit within them, so we extend this representation by adopting conventions from software development, using visual cues such as colour and indentation to allow the user to more readily see the structure and semantics of the query string [2]. Indentation is used to communicate the depth of nesting (within a Boolean expression) so that the mapping between logical structure and physical structure is made more apparent.

In search strategies encoded using the block-based approach, the concept of search lines and line numbering does not exist. We therefore adopt the convention that an opening parenthesis represents an increase in the depth of nesting and is thus used to increment the indentation on the following line, with a closing parenthesis having the opposite effect. We also separate clauses or sub-expressions with an additional line break.

Note that the use of an editable visualisation allows search blocks to be assigned human-readable labels that reflect their purpose and function, analogous to writing comments in code. This contrasts sharply with the current convention of grouping statements using arbitrary line numbers, a practice which has its roots in early programming languages and has long since been deprecated by the software engineering community [8]

3.3 Executing search strategies

There are two primary modes of operation for the prototype, the first of which is interactively running a search strategy to see its effect and understand its behaviour. For multi-line strategies, this is achieved via the use of controls that allow the user to select and inspect a particular range. Once a selection is made, the user can execute that section or step through it interactively. As it runs, the views simultaneously update: in the coding view, each step highlights a new line,

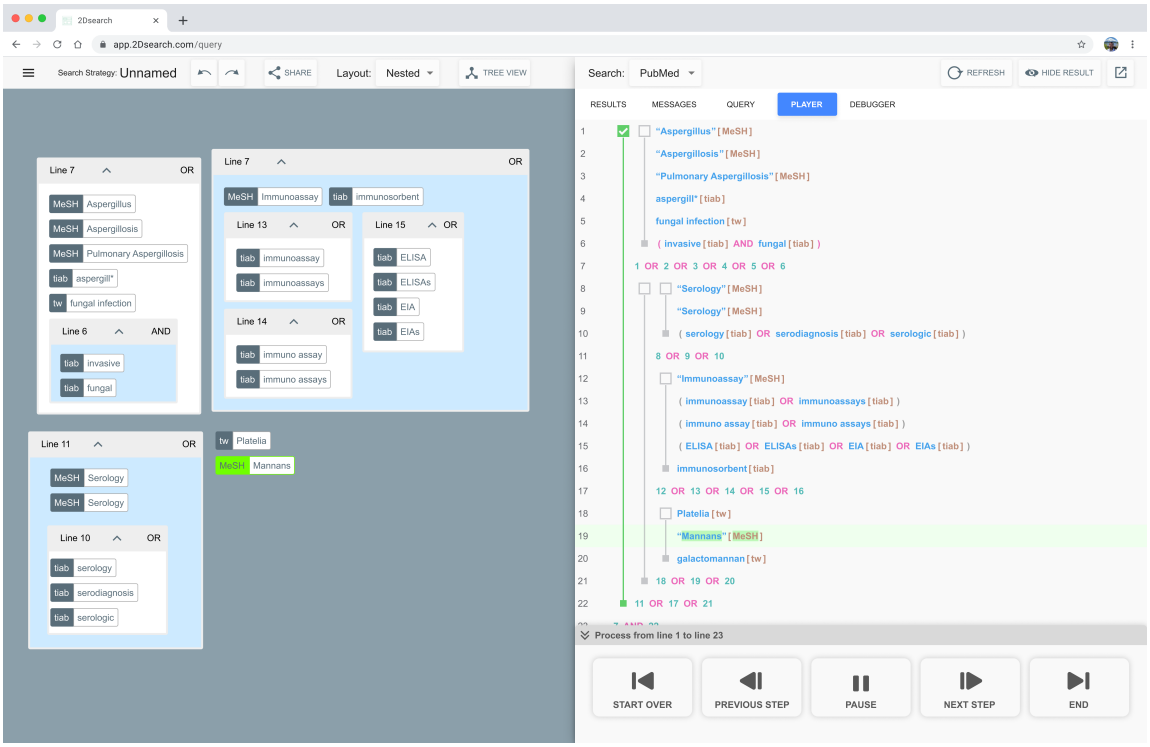

Fig. 2. The application architecture showing partial execution.

whilst on the canvas, the visualisation builds with the corresponding elements highlighted. Visualising the execution of block-based strategies is more challenging since there is no conceptual equivalent of ‘stepping through’ individual lines. Instead, execution proceeds from the outer levels of the strategy to the inner levels, following the layout conventions described above and using a highlighted area to show the current focus of execution (as shown in green in Figure 3).

3.4 Debugging search strategies

The second primary mode of operation is to interactively inspect, validate and debug a search strategy. This involves the detection and resolution of three types of error: syntactic, semantic and pragmatic. Syntactic errors (such as missing parentheses, unmatched quote symbols, etc.) are detected by the query parser and identified by means of a highlight and a notification. Once the user has resolved the issue, the strategy can be executed which retrieves the search results and progressively builds the visualisation with the corresponding elements highlighted. At this point, semantic errors (such as unmatched line references, orphaned lines etc.) are detected. These are also identified by means of a highlight and a notification. Finally, once any syntactic and semantic errors have been identified and resolved, pragmatic errors can be detected. Pragmatic errors (such as unsupported operators or field tags) vary according to the database, so by their nature can only be detected when a search strategy is executed. They do not necessarily prevent a search strategy from running, but can compromise the accuracy of the results. They are shown as notifications displaying the nature and location of the issue that the user can either resolve or dismiss (Figure 4).

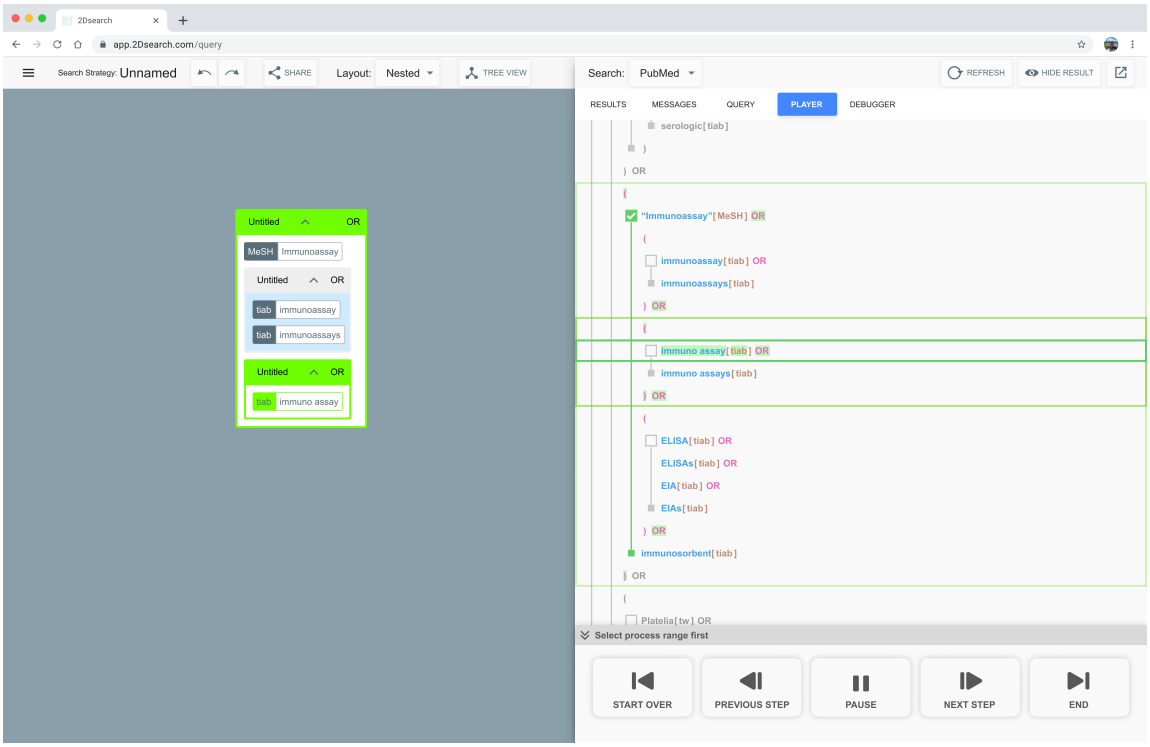

Fig. 3. Executing a block-based search strategy in the Coding view.

4 EVALUATION AND FURTHER WORK

Evaluation has thus far consisted of two rounds of interviews with a convenience sample of six information professionals with experience of structured searching. The objective was to determine how they would understand and react to the concept of executing and debugging search strategies. We summarise the findings as follows.

Most participants (4/6) felt that the use of animation could assist novices in understanding the logical structure of search strategies. P1 reflects on this: "*In terms of explaining how the search is built to students. This [showing the executing process on white board] is exactly what we do in the class. We build it [the search strategy] step by step until we get to the final result.*" This is supported by P2: "*In terms of pedagogy, that looks like a really useful tool to me, you can see the block is created, and you can see the relationship between the blocks. So you gradually build up a picture of how each element of the query [constructed together]. I like the way you could see that happening in real-time.*" Likewise, P5 reflects on being able to see search strategies being executed: "*It is also helpful for students to see how it works. What the system does, how this happens, so it's understanding what is going on when we are using this search strategy.*"

The concept of debugging search strategies was also positively received, with most (5/6) participants suggesting it would be suitable for their own working practices. P3 commented: "*If I wanted to start editing [the search strategy] in here [Debugging mode], it's more efficient to do it here.*" P4 supported this: "*I certainly think this one [Debugging a search strategy] is probably more useful in the use case of me doing my searches. So there's a tighter link between the source search code and the visual representation.*"

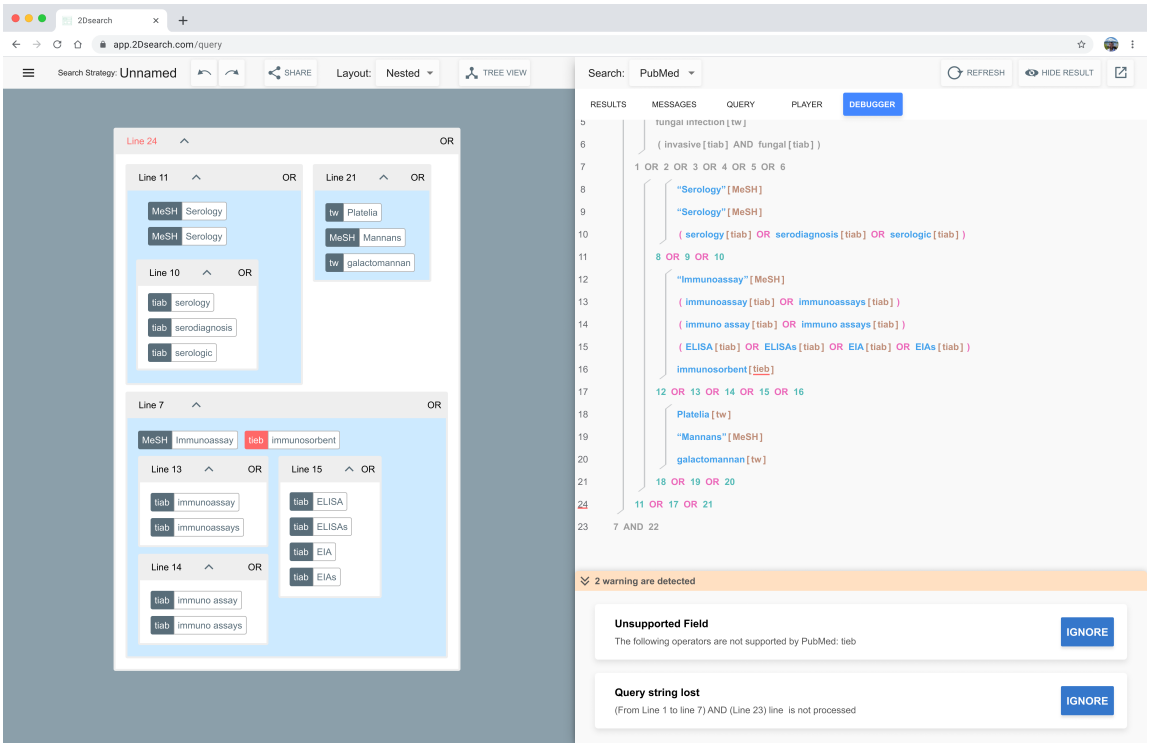

Fig. 4. Detecting and resolving pragmatic errors.

Participant feedback suggests that executing and debugging search strategies are complementary activities and providing support for these tasks could deliver benefits in efficiency and effectiveness. P2 reflects on this: “*They complement each other. The player (executing a search strategy) is building a connection from the command line to the 2D canvas and the debugger (debugging a search strategy) is doing the opposite.*”

However, this work remains a prototype and a number of open questions remain. First, visualising the execution of a search strategy remains a complex challenge, since precedents are few and there are many alternative design directions that could prove fruitful. In our own work, we have identified a promising direction for the application architecture and static components, but found that substantial movements on the canvas proved difficult for participants to follow. Further research into mental models and metaphors is required to determine how to make the animation sufficiently detailed to convey the key changes but not be unduly complex or distracting. Second, if the user edits a strategy using the visualisation view, the query string in the Coding view must also update to stay synchronised. However, the mapping between visualisations and query strings is not one-to-one, since the semantics of a given query string can be visualised in a number of ways, and a given visualisation can be represented by a number of equivalent query strings. This process by which the two views are updated and synchronised therefore needs further investigation. Finally, it would be useful to identify quantitative metrics by which the value of the approach (and the output that users generate) could be assessed, perhaps predicated on the reduction of errors in the various categories outlined above.

5 SUMMARY AND CONCLUSIONS

Systematic literature reviews can take years to complete and there are often mistakes in the search strategies they rely on. Taking inspiration from contemporary software practice, the development of a 'debugger' for search strategies could address many sources of error and inefficiency. However, developing such a tool is a deceptively complex challenge due to the different audience, content and context. In this paper, we have demonstrated a novel approach to executing and debugging search strategies. We have completed an initial evaluation with a number of expert searchers and found that executing and debugging search strategies were complementary activities, and that the concept offers the potential to increase efficiency and reduce errors. Significant design challenges remain, particularly around the use of animation to convey key insights and the need for an unambiguous mapping between the various views. In our next iteration we hope to resolve these issues and deploy the prototype as a fully functional web application.

REFERENCES

[1] Justin Clark, Paul Glasziou, Chris Del Mar, Alexandra Bannach-Brown, Paulina Stehlik, and Anna Mae Scott. 2020. A full systematic review was completed in 2 weeks using automation tools: a case study. *Journal of Clinical Epidemiology* 121 (may 2020), 81–90. https://doi.org/10.1016/j.jclinepi.2020.01.008

[2] Damian K Francis, Joanne Smith, Tawab Saljuqi, and Ruth M Watling. 2015. Oral protein calorie supplementation for children with chronic disease. *Cochrane Database of Systematic Reviews* 5 (27 may 2015), CD001914. https://doi.org/10.1002/14651858.{CD001914}.pub2

[3] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software* (1 ed.). Addison-Wesley Professional, Reading, Mass. 416 pages.

[4] Linh Hoang and Jodi Schneider. 2018. Opportunities for computer support for systematic reviewing - a gap analysis. *Transforming digital worlds : 13th International Conference, iConference 2018, Sheffield, UK, March 25-28, 2018, Proceedings. International Conference on Transforming Digital Worlds (13th : 2018 : Sheffield, England)* 10766 (15 mar 2018), 367–377. https://doi.org/10.1007/978-3-319-78105-1_40

[5] Andrew MacFarlane and Tony Russell-Rose. 2016. Search strategy formulation: a framework for learning. In *Proceedings of the 4th Spanish Conference on Information Retrieval*. 1–8.

[6] Tony Russell-Rose and Jon Chamberlain. 2019. An Open-Access Platform for Transparent and Reproducible Structured Searching. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR'19*. ACM Press, New York, New York, USA, 1293–1296. https://doi.org/10.1145/3331184.3331394

[7] Tony Russell-Rose, Jon Chamberlain, and Farhad Shokraneh. 2019. A visual approach to query formulation for systematic search. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*. 379–383.

[8] Tony Russell-Rose and Farhad Shokraneh. 2020. Designing the Structured Search Experience: Rethinking the Query-Builder Paradigm. *Weave: Journal of Library User Experience* 3, 1 (16 mar 2020). https://doi.org/10.3998/weave.12535642.0003.102

[9] José Antonio Salvador-Oliván, Gonzalo Marco-Cuenca, and Rosario Arquero-Avilés. 2019. Errors in search strategies used in systematic reviews and their effects on information retrieval. *Journal of the Medical Library Association: JMLA* 107, 2 (2019), 210. https://www.ncbi.nlm.nih.gov/pmc/articles/{PMC6466507}/

[10] Margaret Sampson and Jessie McGowan. 2006. Errors in search strategies were identified by type and frequency. *Journal of Clinical Epidemiology* 59, 10 (oct 2006), 1057–1063. https://doi.org/10.1016/j.jclinepi.2006.01.007

[11] Harrisen Scells and Guido Zuccon. 2018. searchrefiner: A Query Visualisation and Understanding Tool for Systematic Reviews. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management - CIKM '18*. ACM Press, New York, New York, USA, 1939–1942. https://doi.org/10.1145/3269206.3269215

[12] Kaveh G Shojania, Margaret Sampson, Mohammed T Ansari, Jun Ji, Steve Doucette, and David Moher. 2007. How quickly do systematic reviews go out of date? A survival analysis. *Annals of Internal Medicine* 147, 4 (21 aug 2007), 224–233. https://doi.org/10.7326/0003-4819-147-4-200708210-00179

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009