pybool_ir: A Toolkit for Domain-Specific Search Experiments

Harrisen Scells
Leipzig University

Martin Potthast
Leipzig University and ScaDS.AI

ABSTRACT

Undertaking research in domain-specific scenarios such as systematic review literature search, legal search, and patent search can often have a high barrier of entry due to complicated indexing procedures and complex Boolean query syntax. Indexing and searching document collections like PubMed in off-the-shelf tools such as Elasticsearch and Lucene often yields less accurate (and less effective) results than the PubMed search engine, i.e., retrieval results do not match what would be retrieved if one issued the same query to PubMed. Furthermore, off-the-shelf tools have their own nuanced query languages and do not allow directly using the often large and complicated Boolean queries seen in domain-specific search scenarios. The `pybool_ir` toolkit aims to address these problems and to lower the barrier to entry for developing new methods for domain-specific search. The toolkit is an open source package available at https://github.com/hscells/pybool_ir.

CCS CONCEPTS

• **Information systems** → **Specialized information retrieval**; **Information retrieval query processing**; *Search engine indexing*.

KEYWORDS

Python, Domain Specific Search, Lucene

ACM Reference Format:
Harrisen Scells and Martin Potthast. 2023. `pybool_ir`: A Toolkit for Domain-Specific Search Experiments. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3591819

1 INTRODUCTION

Indexing and searching domain-specific document collections is often challenging for various reasons. From the indexing side, depending on the quality of the raw documents to be indexed, much pre-processing is required, especially for documents with many fields. From the querying side, implementing domain-specific query languages is tedious and error-prone, especially if the goal is to reproduce a domain-specific retrieval system. One example of a domain-specific document collection frequently used by information retrieval (IR) researchers is PubMed, which contains over 34 million documents at the time of writing. PubMed uses a custom Boolean query language. A Boolean query that one formulates for PubMed will not work on another search engine like Google Scholar. Translating a PubMed query into a query for another search engine is also usually not reliable: PubMed documents have specific fields such as MeSH, a hierarchy of medical terminology used to tag and classify documents.

It is difficult to fully exploit the PubMed document collection given that the implementation of the PubMed indexing and searching pipelines are proprietary and not publicly available. It is a rare instance of a domain-specific search engine with an API that one could use to issue queries in the PubMed query language syntax, but this can be slow, rate limited, and limited in functionality. The alternative for IR researchers is to download the document collection, index, and search in it themselves. However, once the document collection has been indexed, including correct indexing of the many fields a PubMed document contains, the main problem is searching the collection. One must correctly implement the Boolean query syntax used by PubMed: this involves not just fielded search for terms and phrases (which can have wildcards) but date range filters, MeSH explosion (i.e., subsumption of child MeSH terms in the hierarchy), and the Boolean operator semantics. These challenges are present across domain-specific search research.

`pybool_ir` was developed to facilitate clean-room re-implementations of domain-specific search engines. Not only does `pybool_ir` re-implement the indexing process, ensuring that documents are accurately parsed and indexed, but it also provides an interface for closely replicating domain-specific query languages, which are often variations of the Boolean query syntax. The `pybool_ir` toolkit uses the `pylucene` library, which provides a low-level, direct interface with Lucene in Python.[1] The use of `pylucene` enables `pybool_ir` to more closely re-implement features of existing search engines such as PubMed directly in Python. No separate search engine written in another programming language needs to be maintained, like in the case of `pyserini` [7] and `pyterrier` [8], both of which require programming in Java to implement additional functionality. In short, what differentiates `pybool_ir` from other toolkits is the tight integration with domain-specific collections and the ability to closely replicate query languages. In more detail, `pybool_ir` has the following design goals: (1) Indexing pipelines should be concise and straightforward, with support for documents with multiple fields and indexing arbitrary document corpora. (2) Test collections should be easy to load and perform experiments with, including support for existing and arbitrary collections. (3) Implementation of and experimentation with domain-specific query languages should be made as easy as possible. The rest of this paper demonstrates the philosophy of these goals with two concrete examples: a fully operational reproduction of PubMed search,[2] and the first Boolean search engine for the IR Anthology.[3]

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591819

[1] https://lucene.apache.org/pylucene/index.html
[2] https://pubmed.chatnoir.eu
[3] https://IR.chatnoir.eu

```
("Acne Vulgaris"[Mesh] OR Acne[tiab] OR Blackheads[tiab] OR
 Whiteheads[tiab] OR Pimples[tiab]) AND ("Phototherapy"[Mesh] OR
 "Blue light"[tiab] OR Phototherapy[tiab] OR Phototherapies[tiab]
 OR "Photoradiation therapy"[tiab] OR "Photoradiation
 Therapies"[tiab] OR "Light Therapy"[tiab] OR "Light
 Therapies"[tiab]) AND (Randomized controlled trial[pt] OR
 controlled clinical trial[pt] OR randomized[tiab] OR
 randomised[tiab] OR placebo[tiab] OR "drug therapy"[sh] OR
 randomly[tiab] OR trial[tiab] OR groups[tiab]) NOT
 (Animals[Mesh] not (Animals[Mesh] and Humans[Mesh]))
```

Listing 1: Example of a relatively short Boolean query used to search with PubMed from the Wang et al. [21] collection. Longer queries in this collection are up to ten times as large. This query still shows the nuances of domain-specific search: fielded search (i.e., square brackets), implicit query expansion (i.e., use of MeSH terms, which by default include all children within the hierarchy), and the semantics of Boolean operators (e.g., how the NOT operator is implicitly an AND+NOT combination).

```python
from pybool_ir.experiments.collections import load_collection
from pybool_ir.experiments.retrieval import RetrievalExperiment
from ir_measures import *
import ir_measures

# Automatically downloads, then loads this collection.
col = load_collection("ielab/sysrev-seed-collection")

# Point the experiment to your index, your collection.
with RetrievalExperiment(indexer=PubmedIndexer("./pubmed"),
                         collection=col) as experiment:
        # Get the run of the experiment.
        # This automatically executes the queries.
        run = experiment.run

# Evaluate the run using ir_measures.
ir_measures.calc_aggregate([SetP, SetR, SetF], col.qrels, run)
```

Listing 2: Executing PubMed queries from a published test collection and evaluating them. `pybool_ir` will automatically download the test collection and the output of an experiment can be used directly in `ir-measures`.[4]

2 FEATURE SHOWCASE

We first demonstrate the ability of `pybool_ir` to accurately replicate the indexing and searching pipelines of PubMed. Next, we showcase a research use-case for `pybool_ir` by detailing how it can be used to modify documents at indexing-time to support custom query syntax. We then show how to index arbitrary document collections, and finally the integrations `pybool_ir` has with libraries like `ir_datasets` and `pyserini`.

2.1 Replicating PubMed Search

This section explains (1) basic usage guidelines for how to employ `pybool_ir` for retrieval experiments, and (2) the results of these experiments comparing `pybool_ir` to the PubMed search engine (i.e., via the Entrez API [15]). The experiments are conducted using the 2022 baseline document collection of PubMed and the test collection from Wang et al. [21]. This collection is used because it only contains queries issued to the PubMed search engine, unlike the CLEF-TAR collections [4–6], which contain queries issued to other

[4]https://github.com/terrierteam/ir_measures

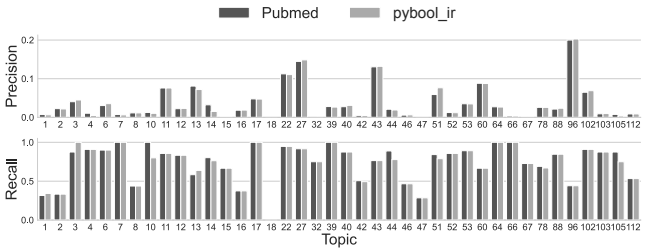

Figure 1: Precision and recall for each query in the Wang et al. [21] collection, comparing PubMed to `pybool_ir`. Note that the query issued to PubMed is identical to the query issued in `pybool_ir`.

```python
from pybool_ir.pubmed.index import PubmedIndexer

with PubmedIndexer("./pubmed-pico",
                   store_fields=True,
                   fields=["P", "I", "O"]) as idx:
        idx.bulk_index("path/to/baseline", fields={
                # The pico function returns the annotation
                # given a document ID.
                "P": lambda doc: pico(doc.id, "population"),
                "I": lambda doc: pico(doc.id, "intervention"),
                "O": lambda doc: pico(doc.id, "outcome")
        })
```

Listing 3: Indexing PubMed with additional PICO fields. This listing demonstrates a feature of `pybool_ir` that allows documents to be modified at index time, but these fields could also be computed offline.

search engines (e.g., Embase). Support for these query languages in `pybool_ir` is planned in future releases. Listing 1 contains a query from the Wang et al. collection to give an understanding of the PubMed query syntax being used.

Listing 2 shows how to use `pybool_ir` for running the experiment. This snippet assumes that the PubMed document collection has already been indexed, which can also be accomplished with `pybool_ir` as it includes a command line tool for downloading and indexing document collections. Special care has been taken for collections like PubMed within `pybool_ir` to ensure that fields like the publication date are indexed correctly. The queries issued to PubMed and `pybool_ir` are identical and no manual processing was required to execute the queries in `pybool_ir`.

Figure 1 reports precision and recall for all topics in the Wang et al. collection. Most results are identical, with no statistical differences (two-tailed paired t-test). Further, the average document overlap across topics (i.e., set overlap between the documents retrieved by PubMed and by `pybool_ir`) is approximately 92.5% (σ=8%). On average, PubMed retrieved 1325.95 (σ=1722.86) documents, while `pybool_ir` retrieved 1386.08 (σ=1644.93). These results show that it is possible to accurately replicate domain-specific search engines using `pybool_ir`. This opens the door to new research possibilities that would otherwise be challenging, if not impossible to study using off-the-shelf tools or APIs, such as extending the syntax of PubMed queries. The following section demonstrates this.

```python
from pybool_ir.pubmed.index import PubmedIndexer
from pybool_ir.experiments.collections import Collection
from pybool_ir.experiments.retrieval import RetrievalExperiment

from ir_measures import *
import ir_measures

# Tell the PubMed query parser about the new fields.
parser = PubmedQueryParser(fields=["P", "I", "O"])

# Load our new custom collection that
# has queries that search on the PICO fields.
col = Collection.from_dir("./sysrev-seed-pico/")

# Experiment loads annotated collection.
with RetrievalExperiment(PubmedIndexer(Path("./pubmed-pico")),
                         collection=col,
                         query_parser=parser)) as exp:
        pico_run = exp.run

# Evaluate the run using ir_measures.
ir_measures.calc_aggregate([SetP, SetR, SetF], col.qrels, pico_run)
```

Listing 4: Running the retrieval experiment on the PubMed PICO index. Note the loading of a collection from a directory.

Search Engine	Recall	Precision	F1
PubMed	0.7362	0.0367	0.0651
pybool_ir	**0.7273**	0.0366	0.0649
+PICO	0.5911	**0.0400**	**0.0675**

Table 1: Results using PICO-annotated documents and queries. Results are similar to those from the original paper: PICO fields on queries increase precision but trade-off this gain in effectiveness for lower recall.

2.2 Reproducing PICO Search

In addition to supporting the indexing of existing collections such as PubMed, it is possible to modify documents at indexing time, or to add additional pre-computed fields to the index. This section shows how custom indexing and query processing pipelines can be implemented. We do this by reproducing the experiments from Scells et al. [17]. In short, these experiments involve indexing additional fields that correspond to different types of clinical information: **P**opulation, **I**ntervention, **C**ontrol, and **O**utcome. The procedure for extracting these annotations for PubMed documents is identical to the original work. However, rather than using Elasticsearch for indexing and searching, as in the original work, `pybool_ir` is used instead. The main benefit here being that queries do not need to be translated into Elasticsearch queries. Listing 3 shows the code required to index PubMed alongside the PICO annotations. Listing 4 shows the code required to run the experiment. Queries were hand-annotated by the authors to search using the additional fields.

Table 1 contains the main results for the PICO experiments. Consistent with previous results, the addition of PICO annotations reduces the recall while increasing precision. Figure 2 provides a topic-oriented view of the results, showing the difference in F1 between the `pybool_ir` PubMed index and the index with PICO annotations. Together, these experiments demonstrate how easy it is to use `pybool_ir` for researching new domain-specific search methods, namely experimenting with modifying and extending existing query languages.

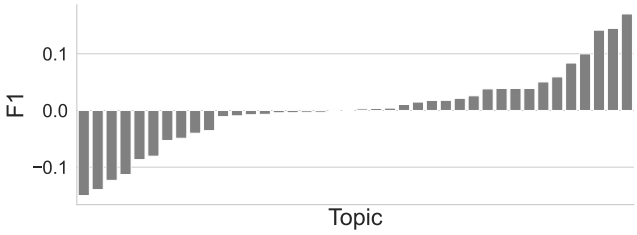

Figure 2: Difference in F1 when using PICO annotations (`pybool_ir` versus `pybool_ir` + PICO).

```
{ "id": "2013.sigirconf_conference-2013.2",
"date": 1541498845.0,
"authors": ["Ryen White"],
"title": "Beliefs and biases in web search",
"abstract": "People's beliefs, and unconscious biases that arise ...
"booktitle": "The 36th International ACM SIGIR conference on ...
"pages": "3-12",
"publisher": "ACM",
"year": "2013",
"doi": "10.1145/2484028.2484053",
"venue": "SIGIR" }
```

Listing 5: Example document from the IR Anothology to be indexed. Only a selection of document fields are shown.

```python
from pybool_ir.index.generic import JsonlIndexer

with JsonlIndexer("ir-anthology", store_fields=True,
                  fields=["#authors", "title", "abstract",
                          "booktitle", "pages", "publisher",
                          "year", "doi", "venue"]) as idx:
        idx.bulk_index("ir-anthology-raw.jsonl")
```

Listing 6: Indexing data from the IR Anthology. The `pybool_ir` library has support for indexing arbitrary collections. Note the usage of the special '#' symbol for the author field, which ensures that each author is indexed separately.

2.3 Indexing the IR Anthology

We next demonstrate indexing and searching on a dataset that is not natively supported by `pybool_ir`. The IR Anthology [11] is a corpus of information retrieval publications. Listing 5 shows what the raw data of the IR Anthology looks like for a single document. Each document in the raw corpus is a JSON object, and the corpus is represented as a JSONL file, where each document is stored on a new line. This is also similar to the JSONLD format which Elasticsearch uses; meaning that corpora in these formats can be indexed in `pybool_ir` with little effort. Listing 6 shows the complete code required to index the data (three lines in total). Once indexed, searches can be performed on the index in several ways. As there are no Cranfield-style test collections for the IR Anthology yet, we instead demonstrate how one can perform ad-hoc searches using `pybool_ir`. Listing 7 shows the first method, which can be achieved with only a handful of lines of Python code. Listing 8 shows the second method, which uses the command-line tool bundled with `pybool_ir` to perform an interactive search. `pybool_ir` simplifies analysing collections of documents using complex queries, e.g., estimating the popularity of different keywords in a document while filtering different attributes such as author, venue, or year.

```python
from pybool_ir.query import GenericQueryParser
from pybool_ir.index.generic import GenericSearcher
from pybool_ir.experiments.retrieval import AdHocExperiment

with AdHocExperiment(GenericSearcher("ir-anthology"),
                     'beliefs:title AND "Ryen White":authors',
                     query_parser=GenericQueryParser()) as exp:
        docs = exp.run
[d.doc_id for d in docs] #-> ['2013.sigirconf_conference-2013.2']
```

Listing 7: Performing a search on the indexed IR Anthology. The `pybool_ir` library has several different query parsers to experiment with query languages, a generic query language similar to the default Lucene syntax is shown here.

```
$ pybool_ir generic search -i ir-anthology
pybool_ir 0.0.4
loaded: ir-anthology
?>beliefs:title AND "Ryen White":authors
hits: 1
2013.sigirconf_conference-2013.2 Beliefs and biases in web ...
```

Listing 8: Performing a search on the indexed IR Anthology, but using the interactive search on the command line.

```
$ pybool_ir ir-datasets index \
        --collection-name cord19/trec-covid \
        --index ./indexes/trec-covid
[ ... ]
$ pybool_ir experiment retrieval \
        --collection-name ird:cord19/trec-covid \
        --index ./indexes/trec-covid \
        --run-path trec-covid.run \
        -e SetR -e SetP -e MAP -e nDCG
{SetR: 0.8405, SetP: 0.01282, AP: 0.0107, nDCG: 0.4048}
```

Listing 9: Indexing the TREC-COVID collection directly from `ir-datasets` and then immediately performing a retrieval experiment.

2.4 Experiments with `ir_datasets`

The `ir_datasets` catalogue contains information retrieval test collections (document corpora, topics, and relevance assessments). `pybool_ir` is tightly integrated with `ir_datasets` to the extent that indexing a collection and obtaining baseline retrieval metrics is possible with two commands on the command line. Listing 9 shows the two commands for indexing and performing a baseline retrieval experiment using the TREC-COVID collection [19]. Given that one of the design goals of `pybool_ir` is to experiment with query parsing and query languages in general, the integration with `ir_datasets` provides a platform for conducting information retrieval experiments with complex queries such as domain-specific Boolean query languages. The experiments in Listing 9 support the same arguments as the Python code, allowing one to study the effect of different query parsers. This integration allows others to use `ir_datasets` for domain-specific collections.

Existing instances of collections in the `ir_datasets` catalogue that may benefit from the investigation into domain-specific queries include the args.me collection [1, 20], which contains complex documents containing different aspects of argumentation and the TREC tracks that use medical corpora, e.g., the genomics track [3], the clinical decisions support track [18], and the precision medicine track [13] which all also have documents containing multiple fields.

```
$ python -m pyserini.search.lucene \
        --topics /path/to/topics \
        --index ./indexes/trec-covid
```

Listing 10: Searching the TREC-COVID collection that was indexed using `pybool_ir` with `pyserini`.

2.5 Compatibility with `pyserini`

Finally, since `pybool_ir` uses Lucene, indexes created with it are compatible with `pyserini`. This means that one can index documents using either toolkit and then perform retrieval experiments using the other. Listing 10 shows the `pyserini` command one would use to search the index created with `pybool_ir`. This compatibility makes `pybool_ir` a viable tool to implement more advanced indexing techniques for use in `pyserini` and `pybool_ir` also inherits many useful features from `pyserini` for search.

3 DISCUSSION

Boolean retrieval is an essential tool for many scientific and specialised research fields. Medical information retrieval, particularly systematic review literature search, is one of the primary examples where Boolean retrieval is the norm rather than the exception. For researchers in these domains, there are tools that exist to assist with developing queries [14, 16, 22], as well as a variety of tools to assist with ranking or filtering retrieved documents [2, 9, 10, 12], to name some prominent tools in the medical domain. However, few tools exist for the information retrieval practitioner (i.e., the target audience of this demo) to study the retrieval or ranking mechanisms in these domain-specific systems; existing tools often do not support the particular indexing and search requirements of domain-specific search. `pybool_ir` provides the tools for information retrieval practitioners to study and improve the underlying mechanisms that domain-specific search systems depend on.

4 CONCLUSIONS

Future plans for `pybool_ir` include replication of more domain-specific search engines from the biomedical, legal, and patent search domains. We envision that the indexing of documents as well as domain-specific query languages, e.g., the Boolean query syntax in the case of PubMed, are to be replicated for these search engines. Due to how `pybool_ir` parses and represents queries, it is also possible to use `pybool_ir` for query analysis by navigating the parse tree, and programmatically modify the syntax of queries using code, such as extracting clauses from a query or performing query expansion on Boolean queries. These research avenues were previously challenging to study, since the barrier to entry was high compared to ad-hoc search. `pybool_ir` lowers this barrier for any kind of domain-specific IR. `pybool_ir` is open source and freely available from https://github.com/hscells/pybool_ir.

ACKNOWLEDGMENTS

Dr Harrisen Scells is the recipient of an Alexander von Humboldt Stiftung Research Fellowship. This work was partially funded by the European Commission under GA 101070014 (OpenWebSearch.EU).

REFERENCES

[1] Yamen Ajjour, Henning Wachsmuth, Johannes Kiesel, Martin Potthast, Matthias Hagen, and Benno Stein. 2019. Data Acquisition for Argument Search: The Args.Me Corpus. In *42nd German Conference on Artificial Intelligence (KI 2019)*, Christoph Benzmüller and Heiner Stuckenschmidt (Eds.). Springer, Berlin Heidelberg New York, 48–59. https://doi.org/10.1007/978-3-030-30179-8_4

[2] 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 (2020), 81–90.

[3] William R. Hersh, Ravi Teja Bhuptiraju, Laura Ross, Phoebe Johnson, Aaron M. Cohen, and Dale F. Kraemer. 2004. TREC 2004 Genomics Track Overview. In *TREC*.

[4] Evangelos Kanoulas, Dan Li, Leif Azzopardi, and Rene Spijker. 2017. CLEF 2017 Technologically Assisted Reviews in Empirical Medicine Overview. In *CEUR Workshop Proceedings: Working Notes of CLEF 2017: Conference and Labs of the Evaluation Forum*.

[5] Evangelos Kanoulas, Dan Li, Leif Azzopardi, and Rene Spijker. 2019. CLEF 2019 Technology Assisted Reviews in Empirical Medicine Overview. In *CEUR Workshop Proceedings: Working Notes of CLEF 2018: Conference and Labs of the Evaluation Forum*, Vol. 2380.

[6] Evangelos Kanoulas, Rene Spijker, Dan Li, and Leif Azzopardi. 2018. CLEF 2018 Technology Assisted Reviews in Empirical Medicine Overview. In *CEUR Workshop Proceedings: Working Notes of CLEF 2018: Conference and Labs of the Evaluation Forum*.

[7] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, Virtual Event Canada, 2356–2362.

[8] Craig Macdonald, Nicola Tonellotto, Sean MacAvaney, and Iadh Ounis. 2021. PyTerrier: Declarative Experimentation in Python from BM25 to Dense Retrieval. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, Virtual Event Queensland Australia, 4526–4533.

[9] Iain J Marshall, Joël Kuiper, and Byron C Wallace. 2016. RobotReviewer: Evaluation of a System for Automatically Assessing Bias in Clinical Trials. *Journal of the American Medical Informatics Association* 23, 1 (2016), 193–201.

[10] Mourad Ouzzani, Hossam Hammady, Zbys Fedorowicz, and Ahmed Elmagarmid. 2016. Rayyan—a Web and Mobile App for Systematic Reviews. *Systematic Reviews* 5, 1 (2016), 210. https://doi.org/10.1186/s13643-016-0384-4

[11] Martin Potthast, Sebastian Günther, Janek Bevendorff, Jan Philipp Bittner, Alexander Bondarenko, Maik Fröbe, Christian Kahmann, Andreas Niekler, Michael Völske, Benno Stein, and Matthias Hagen. 2021. The Information Retrieval Anthology. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, Virtual Event Canada, 2550–2555. https://doi.org/10.1145/3404835.3462798

[12] Piotr Przybyła, Austin J. Brockmeier, Georgios Kontonatsios, Marie-Annick Le Pogam, John McNaught, Erik von Elm, Kay Nolan, and Sophia Ananiadou. 2018. Prioritising References for Systematic Reviews with RobotAnalyst: A User Study. *Research Synthesis Methods* 9, 3 (2018), 470–488. https://doi.org/10.1002/jrsm.1311

[13] Kirk Roberts, Dina Demner-Fushman, Ellen M. Voorhees, William R. Hersh, Steven Bedrick, and Alexander J. Lazar. 2018. Overview of the TREC 2018 Precision Medicine Track. In *TREC*.

[14] Tony Russell-Rose and Philip Gooch. 2018. 2dSearch: A Visual Approach to Search Strategy Formulation. In *Proceedings of the 1st Biennial Conference on Design of Experimental Search and Information Retrieval Systems*.

[15] Eric Sayers. 2010. A General Introduction to the E-Utilities. *Entrez Programming Utilities Help [Internet]. Bethesda: National Center for Biotechnology Information* (2010).

[16] Harrisen Scells and Guido Zuccon. 2018. Searchrefiner: A Query Visualisation and Understanding Tool for Systematic Reviews. In *Proceedings of the 27th International Conference on Information and Knowledge Management*. 1939–1942.

[17] Harrisen Scells, Guido Zuccon, Bevan Koopman, Anthony Deacon, Leif Azzopardi, and Shlomo Geva. 2017. Integrating the Framing of Clinical Questions via PICO into the Retrieval of Medical Literature for Systematic Reviews. In *Proceedings of the 26th International Conference on Information and Knowledge Management*. 2291–2294.

[18] Matthew S. Simpson, Ellen M. Voorhees, and William Hersh. 2014. Overview of the TREC 2014 Clinical Decision Support Track. In *TREC*.

[19] Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. TREC-COVID: Constructing a Pandemic Information Retrieval Test Collection. *ACM SIGIR Forum* 54, 1 (Feb. 2021), 1:1–1:12.

[20] Henning Wachsmuth, Martin Potthast, Khalid Al-Khatib, Yamen Ajjour, Jana Puschmann, Jiani Qu, Jonas Dorsch, Viorel Morari, Janek Bevendorff, and Benno Stein. 2017. Building an Argument Search Engine for the Web. In *4th Workshop on Argument Mining (ArgMining 2017) at EMNLP*, Kevin Ashley, Claire Cardie, Nancy Green, Iryna Gurevych, Ivan Habernal, Diane Litman, Georgios Petasis, Chris Reed, Noam Slonim, and Vern Walker (Eds.). Association for Computational Linguistics, 49–59.

[21] Shuai Wang, Harrisen Scells, Justin Clark, Bevan Koopman, and Guido Zuccon. 2022. From Little Things Big Things Grow: A Collection with Seed Studies for Medical Systematic Review Literature Search. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[22] Jingfan Zang and Tony Russell-Rose. 2023. A Prototype "Debugger" for Search Strategies. In *Proceedings of the 2023 Conference on Human Information Interaction and Retrieval*. ACM, Austin TX USA, 417–421. https://doi.org/10.1145/3576840.3578321