CNSVRE: A Query Reformulated Search System with Explainable Summarization for Virtual Research Environment

Na Li
Informatics Institute, University of Amsterdam
Amsterdam, Netherlands
n.li@uva.nl

Yangjun Zhang
Informatics Institute, University of Amsterdam
Amsterdam, Netherlands
y.zhang6@uva.nl

Zhiming Zhao
Informatics Institute, University of Amsterdam
Amsterdam, Netherlands
z.zhao@uva.nl

ABSTRACT

Computational notebook environments have drawn broad attention in data-centric research applications, e.g., virtual research environment, for exploratory data analysis and algorithm prototyping. Vanilla computational notebook search solutions have been proposed but they do not pay much attention to the information needs of scientific researchers. Previous studies either treat computational notebook search as a code search problem or focus on content-based computational notebook search. The queries being considered are neither research-concerning nor diversified whereas researchers' information needs are highly specialized and complex. Moreover, relevance evaluation for computational notebooks is tricky and unreliable since computational notebooks contain fragments of text and code and are usually poorly organized. To solve the above challenges, we propose a computational notebook search system for virtual research environment (VRE), i.e., CNSVRE, with scientific query reformulation and computational notebook summarization. We conduct a user study to demonstrate the effectiveness, efficiency, and satisfaction with the system.

CCS CONCEPTS

• **Information systems** → **Personalization**; **Query reformulation**; **Summarization**.

KEYWORDS

computational notebook search, virtual research environment, scientific query reformulation, code summarization

ACM Reference Format:
Na Li, Yangjun Zhang, and Zhiming Zhao. 2023. CNSVRE: A Query Reformulated Search System with Explainable Summarization for Virtual Research Environment. In *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3543873.3587360

1 INTRODUCTION

The virtual research environment, a collaborative platform designed to support research activities, has been widely used for supporting the research community with access to tools, resources, and services [Barker et al. 2019]. Computational notebook environments, e.g., Jupyter notebook, are promising solutions for hosting VRE services. Users can carry out frequent data manipulation, investigate various data analytic approaches, and demonstrate the access and usage of datasets, models, libraries, or APIs with computational notebooks [Perkel 2018; Rule et al. 2018]. Collecting computational notebooks and building a search system for VREs help scientists discover research resources published through computational notebooks and reduce repetitive work [Zhao et al. 2022].

There are several studies on computational notebook search [Horiuchi et al. 2022; Li et al. 2021]. One branch of work emphasizes code fragments of computational notebooks and returns code snippets given natural language queries [Li et al. 2021]. They essentially treat computational notebook search as a code search problem and aim to bridge the gap between programming language and natural language. Nevertheless, the queries being considered are rather programming-oriented than research-oriented, misaligned with researchers' sophisticated and specialized information needs. Another line of work investigates content-based computational notebook search (or notebook-to-notebook search), which uses computational notebooks as queries to retrieve relevant computational notebooks [Horiuchi et al. 2022]. These methods provide a representation for the entire computational notebook, i.e., texts and codes, but limit the application to scenarios where users already have a computational notebook in hand. Both studies do not touch on the explainability of search results. Given the complexity of computational notebook contents, the relevance judgment made by the system varies largely depending on the underlying models. A lack of explanations may dampen users' satisfaction and trust in the presented computational notebooks.

Despite the studies that have been carried out in computational notebook search, people often resort to general-purposed search engines or open-source code repositories for computational notebook search. The need for a system that facilitates collecting computational notebooks and facilitating computational notebook search that supports VRE is clear. Moreover, it is important to solve the challenges of computational notebook representation and query understanding. First, computational notebooks are long documents comprising free-form textual descriptions and executable codes. It is not straightforward to effectively model the entire computational notebook. Second, user queries are usually short and made up of several keywords that demand great efforts for query augmentation.

To solve the above challenges, we propose computational notebook search for VRE (CNSVRE) that supports scientific query understanding and computational notebook summarization. The user study suggests the efficiency and satisfaction of the system as well as the effectiveness of the summarization. The main contributions of our work are as follows. First, we collect the dataset and build

WWW '23 Companion, April 30-May 4, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9419-2/23/04.
https://doi.org/10.1145/3543873.3587360

a computational notebook search system for VRE. The system is implemented in a Jupyter environment that provides in-site computational notebook search for convenient code copy-pasting and for ease of code execution in the current working environment. Second, we reformulated the query of the users to improve query diversity and provide summarizations of the computational notebooks as the explanation of search results. Source codes can be found here[1].

2 RELATED WORK

2.1 Query modeling for computational notebook search

Query modeling is essential to improve query understanding and thus boost the performance of an information retrieval system. A VRE solution embedded in Jupyter environment has been introduced by Zhao et al. [2022] which includes a computational notebook search system as one essential component. A plain computational notebook search system has been proposed by Li et al. [2022]. However, the methods described above do not include query modeling which is important to satisfy researchers' information needs. Previous methods utilize the query history by users, user profiles, and user interaction for query modeling. Xu et al. [2008] proposes expanding a query with terms/phrases related to entities in the query. In contrast, Bhopale and Tiwari [2021] suggests using phrase embeddings to find semantically similar words as expansions to users' queries. Another exploratory study [Diaz 2016] presents query reformulation as a graph search problem. The graph comprises queries, retrieved results, and their relationships; the best query is obtained through graph navigation during user interactions.

Unlike the above methods, we explicitly extract scientific entities, categorized as *task*, *dataset*, *method* and augment the query using entities from an external knowledge graph (KG).

2.2 Explainability of code search

An important weakness of code search methods is the lack of explainability due to the absence of metrics that provides insights into models' decision-making process [Liu et al. 2021]. Cito et al. [2022] explores counterfactual explanations for models of source code, defined as minimal changes to the source code under which the model makes a different decision. However, counterfactual explanations are unnecessarily in-depth for a computational notebook seeker. Instead, we add a summary section on the search engine results page (SERP) along with relevance scores outputted by the summarization model to increase the explainability.

3 SYSTEM ARCHITECTURE

3.1 Overall system design

The overview of our framework is illustrated in Figure 1. The main advantages of our framework compared to a vanilla computational notebook search system [Li et al. 2022] are the improvements in user experiences through active query understanding and content summarization (marked as blue in Figure 1). The main building blocks of our frameworks are explained below:

[1]https://github.com/QCDIS/notebook-search-helm-charts

(1) The query reformulation module receives an initial query from a user, reformulates the query, and selects top ranked reformulated queries to users.
(2) A text summarization unit computes a dense description based on the texts inside the markdown cells and a code summarization unit produces a natural language summary for the code fractions. Both outputs will be concatenated into a single paragraph as the summary of the computational notebook.
(3) The computational notebook indexer and retriever processes raw computational notebooks to generate indexes and then uses the selected query to retrieve computational notebooks from the index database.
(4) The result interface displays two things to users: a short summarizing paragraph generated by the computational notebook summarizer for users to quickly examine the relevance and the original contents of computational notebooks for more detailed information.

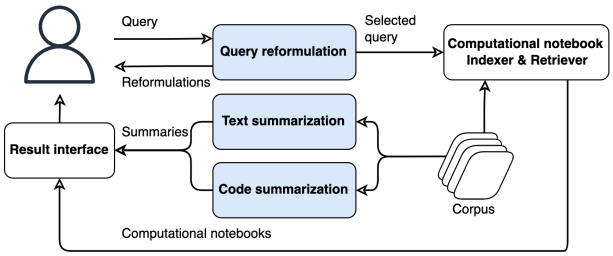

Figure 1: Proposed computational notebook search system with query reformulation, text summarization and code summarization.

Formally speaking, given a query q and a computational notebook corpus $C = \{x_1, x_2, \cdots, x_M\}$, the computational notebook search task is to return a ranked list of notebooks $[x_i], x_i \in C$ based on the relevance between x_i and q. A query reformulation unit $A : q \rightarrow \hat{q}$ outputs an enhanced query $\hat{q}$ for subsequent retrieval. Additionally, a summarizer $S : x_i \rightarrow z_i$ provides a short summary z_i for the computational notebook x_i to be displayed on the search engine results page.

3.1.1 Query reformulation. Query understanding is crucial for delivering useful information to users. Our framework performs active query understanding via query reformulation. The process can be formulated as follows:

(1) the user sends an initial query q_0 to the system;
(2) the system reformulates the query to generate N candidate queries $Q = \{q_1, q_2, \cdots, q_N\}$, and selects top M-ranked queries to be displayed to the user, $M \ll N$;
(3) the user clicks on one of the reformulated queries that mostly match their information needs.

We use an entity-aware approach to generate query expansions:

$$\hat{q} = A(q, G), \tag{1}$$

where G denotes an external KG containing scientific entities. We first extract entities from the query q using DyGIE++ [Wadden et al. 2019] and then search the KG for related entities as expansions.

3.1.2 Text and code summarization. The summarization of computational notebooks produces a compact yet informative natural language description for the contents. The difficulty of this task lies in the fact that both texts and codes should be summarized. Considering a computational notebook composed of multiple cells $x_i = [w_1^{(i)}, w_2^{(i)}, \cdots, w_m^{(i)}, c_1^{(i)}, c_2^{(i)}, \cdots, c_n^{(i)}]$, where $w_j^{(i)}$ denotes a markdown cell and $c_k^{(i)}$ stands for a code cell. The orders for code cells and markdowns are retained separately. The summarization z_i is computed as

$$z_i = S(x_i) = S_1([w_j^{(i)}]) \oplus S_2([c_k^{(i)}]) \quad (2)$$

where $\oplus$ is a concatenation operation, S_1 is a summarizer only for texts inside markdown cells $[w_j^{(i)}]$ and S_2 is dedicated to code summarization that takes codes inside code cells $[c_k^{(i)}]$ as input. The concatenated text and code summarizations becomes the summarization for the computational notebook. During implementation, we use T5 [Raffel et al. 2020] model for text summarization and CodeTrans [Elnaggar et al. 2021] for code summarization. We also provide a relevance score $rel = Sim(z_i, [w_j^{(i)}])$ as explanations for the summarization model. rel is computed as the cosine similarity between TF-IDF vectors of summaries and texts in markdown cells.

3.1.3 Computational notebook indexer and retriever. The computational notebook indexer and retriever are not the main focus of our framework. For the sake of resource efficiency, we use the basic BM25 algorithm and use Elasticsearch to handle the indexing and retrieval.

3.2 Dataset

In terms of the computational notebooks, we crawled 8,491 computational notebooks from Kaggle[2] and successfully processed and indexed 7,849 of them. We use data from PaperswithCode[3] as the external KG for query reformulation.

4 DEMONSTRATION

4.1 User interface

The result interface adopts a hierarchical information disclosure method for users to assess the relevance of returned results. First, it displays the titles of returned computational notebooks on the SERP. When users click on one item, the system reveals more details, including a short summary and metadata, e.g., data source, number of cells and programming language, as shown in Figure 2. It provides more reliable hints on the real contents, based on which users will decide whether to peruse the whole computational notebook. Worth to mention that, this is done under the users' own development environment; they can download desired computational notebooks directly into the working space.

Our framework is developed within a VRE [Zhao et al. 2022] as one of the main components to support collaborative research. VRE users are able to use our framework out of box. Apart from searching functionality, we provide a side channel for users to submit their usefulness judgments towards examined computational notebooks, which will be used for further research.

[2]https://www.kaggle.com
[3]https://paperswithcode.com/

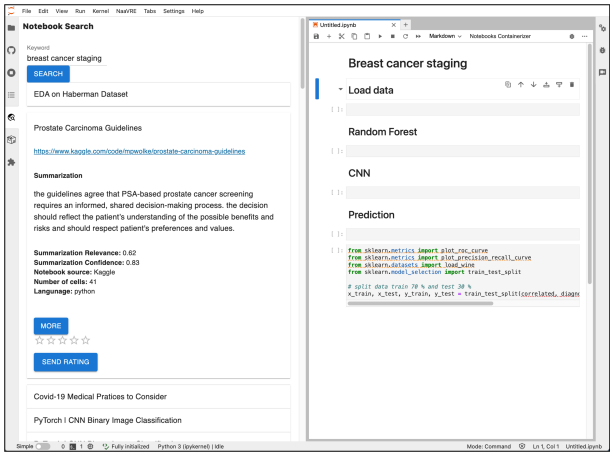

Figure 2: User interface (UI) for CNSVRE.

4.2 Query reformulation

Table 1 gives two examples for query reformulation. Given an query "graph models", the system first determines it as a *collection* entity and then retrieves related *methods* entities and appends them to the initial query.

Table 1: Query reformulation examples.

Initial query	Reformulated queries
video segmentation	video segmentation *bdd100k* video segmentation *segtrack-v2* video segmentation *conferencevideosegmentationdataset* video segmentation *tiktok dataset* video segmentation *pp-humanseg14k* video segmentation *petraw*
graph models	graph models *symbolic deep learning* graph models *pna* graph models *dualgcn* graph models *graph transformer* graph models *bigcn* graph models *diffpool* graph models *gin*

5 USER STUDY

5.1 User study design

We conducted a user study to evaluate the efficiency of the proposed system. We asked the participants to finish a computational notebook search task and answer a post-task questionnaire. Instructions were provided in written form, which contained guidance for accessing the system and performing the task. The task is described as below:

(1) Type your query in the search bar (something related to your research/project) and click "search".
(2) Go through the returned results and find the most desired notebooks for your needs.

(3) Change your query if necessary.

We only collected data for the purpose of this user study. All the data will be published after anonymization. And we explicitly asked for participants' consent to collect, store, analyze and publish their data. We recruited 6 participants (2 female and 4 male) who are researchers. The average age is 27.7, with the youngest being 23 and the oldest 34. Their fields and research interests are listed in Table 2.

Table 2: Research interests of participants.

Field	Research Interests
Computer vision	segmentation, diffusion models, medical imaging
Computer architecture	chip design and dissipation
Artificial intelligence	AI for WSI analysis
Medical education	equational technology
Computer science	big data engineering
Computer vision	computer vision

5.2 Results

5.2.1 Efficiency of the system. The consumed time and the number of queries being used are displayed in Table 3. On average, it took one participant 8.33 minutes to finish the task and on average 8.6 queries were formulated to search for computational notebooks. 50% of the participants confirmed that they successfully found desired computational notebooks, whereas the rest said the results were plentiful for some queries but exclusive for others. One participant reported that "I could find some nice/relevant notebooks if I search something general, i.e., segmentation. But if the search keywords become more specific, i.e., video segmentation, relevant results are scarce", which suggests a better coverage for more specified information needs.

Table 3: User study results.

	Mean	Std.
Consumed time (mins)	8.33	5.50
Number of queries	8.60	12.01

5.2.2 Satisfaction with the system. In terms of satisfaction, we asked participants to scale 1-5 if they agreed with the following statements:

- Statement 1: I found the system unnecessarily complex.
- Statement 2: I would imagine that most people would learn to use this system very quickly.

The mean scaling for statement 1 is 2.83, which means they did not find it overcomplex. 83% participants fully agree with Statement 2, indicating that our system is easy to use.

5.2.3 Effectiveness of summarization. We asked the participants to scale 1-5 for the degree they found "the summarization help you to judge the relevance of the computational notebooks? " The average value is 3.83, suggesting that participants found the summarization relatively useful for relevance assessment.

6 CONCLUSION AND FUTURE WORK

We propose CNSVRE, with a query reformulation module that solves the query ambiguity challenge needs and a computational notebook summarization module for the explainability of the search results. Based on the proposed system, we conduct a user study and verify the effectiveness, efficiency, and satisfaction of the system. We believe our system is applicable to the VRE for different scientific research domains, e.g., bioinformatics, and physiography studies.

ACKNOWLEDGMENTS

This work was partially funded by the European Union Horizon 2020 and Horizon Europe research and innovation programs through projects CLARIFY (860627), ENVRI-FAIR (824068), BlueCloud (862409), Blue-Cloud 2026 (101094227) and the LifeWatch ERIC.

REFERENCES

Michelle Barker, Silvia Delgado Olabarriaga, Nancy Wilkins-Diehr, Sandra Gesing, Daniel S Katz, Shayan Shahand, Scott Henwood, Tristan Glatard, Keith Jeffery, Brian Corrie, et al. 2019. The global impact of science gateways, virtual research environments and virtual laboratories. *Future Generation Computer Systems* 95 (2019), 240–248.

Amol P Bhopale and Ashish Tiwari. 2021. Leveraging Neural Network Phrase Embedding Model for Query Reformulation in Ad-hoc Biomedical Information Retrieval. *Malaysian Journal of Computer Science* 34, 2 (2021), 151–170.

Jürgen Cito, Isil Dillig, Vijayaraghavan Murali, and Satish Chandra. 2022. Counterfactual explanations for models of code. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*. 125–134.

Fernando Diaz. 2016. Pseudo-query reformulation. In *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*. Springer, 521–532.

Ahmed Elnaggar, Wei Ding, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Silvia Severini, Florian Matthes, and Burkhard Rost. 2021. CodeTrans: Towards Cracking the Language of Silicon's Code Through Self-Supervised Deep Learning and High Performance Computing. *arXiv preprint arXiv:2104.02443* (2021).

Misato Horiuchi, Yuya Sasaki, Chuan Xiao, and Makoto Onizuka. 2022. JupySim: Jupyter Notebook Similarity Search System. *Open Proceedings* (2022).

Na Li, Siamak Farshidi, Riccardo Bianchi, Spiros Koulouzis, and Zhiming Zhao. 2022. Context-Aware Notebook Search in a Jupyter-Based Virtual Research Environment. In *2022 IEEE 18th International Conference on e-Science (e-Science)*. IEEE, 393–394.

Xingjun Li, Yuanxin Wang, Hong Wang, Yang Wang, and Jian Zhao. 2021. NBSearch: Semantic Search and Visual Exploration of Computational Notebooks. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

Chao Liu, Xin Xia, David Lo, Cuiyun Gao, Xiaohu Yang, and John Grundy. 2021. Opportunities and challenges in code search tools. *ACM Computing Surveys (CSUR)* 54, 9 (2021), 1–40.

Jeffrey M Perkel. 2018. Why Jupyter is data scientists' computational notebook of choice. *Nature* 563, 7732 (2018), 145–147.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

Adam Rule, Aurélien Tabard, and James D Hollan. 2018. Exploration and explanation in computational notebooks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, Relation, and Event Extraction with Contextualized Span Representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 5784–5789.

Yang Xu, Fan Ding, and Bin Wang. 2008. Entity-based query reformulation using wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*. 1441–1442.

Zhiming Zhao, Spiros Koulouzis, Riccardo Bianchi, Siamak Farshidi, Zeshun Shi, Ruyue Xin, Yuandou Wang, Na Li, Yifang Shi, Joris Timmermans, et al. 2022. Notebook-as-a-VRE (NaaVRE): From private notebooks to a collaborative cloud virtual research environment. *Software: Practice and Experience* 52, 9 (2022), 1947–1966.