DeepGit: Promoting Exploration and Discovery of Research Software with Human-Curated Graphs

Yilin Xia
University of Illinois
Urbana-Champaign, IL, USA
yilinx2@illinois.edu

Shin-Rong Tsai
University of Illinois
Urbana-Champaign, IL, USA
srtsai@illinois.edu

Matthew Turk
University of Illinois
Urbana-Champaign, IL, USA
mjturk@illinois.edu

VLDB Workshop Reference Format:
Yilin Xia, Shin-Rong Tsai, and Matthew Turk. DeepGit: Promoting Exploration and Discovery of Research Software with Human-Curated Graphs. VLDB 2025 Workshop: DaSH.

1 INTRODUCTION

Computational methods are central to modern scientific research [9], and thus research software plays a vital role. Consequently, familiarizing oneself with a new field involves not only reading academic publications but also reviewing domain-specific software tools. However, exploring and discovering such tools remains challenging, as they are often scattered across various platforms [23] and many lack formal citations for reference [12, 22]. This challenge is exacerbated by the fact that the current research software ecosystem lacks centralized discovery platforms comparable to Google Scholar. Although GitHub (arguably the most popular modern software development and distribution platform) offers some support for locating research software, it is typically biased toward popularity metrics [14] and focuses on individual repositories, lacking insight into **inter-repository relationships**. A more structured approach – considering both repositories and their relationships – is essential for improving research software discovery, supporting informed research decisions, and increasing software visibility.

Graphs are commonly used to support relationship-aware approaches, as they provide intuitive representations of entities and their connections [6]. In practice, such graphs are often constructed using automated methods [27]. However, the construction of domain knowledge graphs requires explicit conceptualization [1], and automated techniques often fail to capture domain-specific nuances that are essential for accurate representations [8]. Furthermore, graph schema is typically defined by the creators, restricting domain experts and other end-users to predefined queries, thus reducing their ability to shape the graph's structure according to their needs. Therefore, human involvement is crucial in the graph construction process to capture domain-specific knowledge and ensure the graph's relevance.

We introduce DeepGit, an open source, domain-aware engine that utilizes a human-curated graph for exploring and discovering research software on GitHub [1]. DeepGit allows users to narrow down potential GitHub topics [4], define semantic relationships among repositories, interactively extract subgraphs by applying metadata filters, and explore underlying patterns through Graph Retrieval-Augmented Generation (GraphRAG) [18]. By incorporating human efforts, DeepGit provides researchers the ability to customize and construct domain-specific subgraphs to explore, discover, and review research software. These capabilities also directly advance *findability*, one of the fundamental FAIR principles that promotes the reuse and sustainability of research software [15].

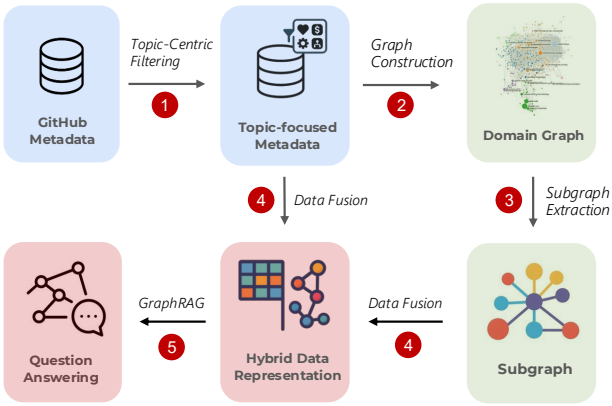

Figure 1: DeepGit follows a five-step process: (1) building a topic-focused knowledge base from GitHub metadata, (2) constructing a domain graph, (3) extracting subgraphs, (4) fusing data into a hybrid representation, and (5) enabling question answering via GraphRAG.

2 DEEPGIT

To actively engage humans in the exploration and discovery process, DeepGit is composed of three major components: Topic-Centric Filtering, Graph Construction, and Subgraph Extraction & GraphRAG.

2.1 Topic-Centric Filtering

Topic-Centric Filtering is a critical step for subsequent exploration, as it narrows down the vast GitHub metadata[2] into a domain-focused dataset. We adopt the topic summarization and modeling method [13, 24] to guide the following procedure:

Given a topic X, DeepGit retrieves a set of repositories $\mathcal{R} = \{R_1, R_2, \ldots\}$ from GitHub metadata tagged with X, and aggregates

Proceedings of the VLDB Endowment. ISSN 2150-8097.

[1] GitHub is used in the initial phase. We will later switch to a more comprehensive research software database, such as MOSS [21]

[2] Data acquisition involves using GitHub API or DuckDB [20] to query existing metadata—provided in JSON format—sourced from Kaggle. The dataset comprises 3,985,968 repositories, totaling 3.07 GB in size [5].

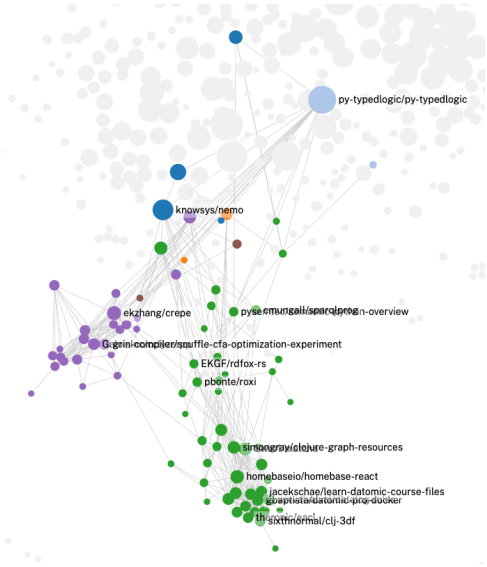

Figure 2: The colored nodes represent 98 repositories tagged with the "Datalog" topic, connected by 388 edges. Gray nodes (not all are shown) represent the remaining repositories in the corpus $\mathcal{R}'$ for logic programming. An edge is drawn between two repositories if they share at least one logic programming topic (e.g., Prolog). Node size reflects an importance score computed using PageRank. Of particular interest is the repository "py-typedlogic", which, despite representing a relatively small community, shares several topics with other repositories .

all topics contained in $\mathcal{R}$ into a set $\mathcal{T} = \{T_1, T_2, \ldots\}$. The set $\mathcal{T}$ is iteratively refined through a user-centered, three-stage process:

(1) **Frequency-Based Filtering**: Topics are counted and presented as a histogram. Users actively select a frequency range to filter and obtain a subset $\mathcal{T}_{\text{freq}} \subseteq \mathcal{T}$.
(2) **LLM-Powered Refinement**: As $\mathcal{T}_{\text{freq}}$ can remain large, Large Language Models are utilized for continued refinement, with users customizing prompts to guide the refinement process and get a refined subset $\mathcal{T}_{\text{llm}} \subseteq \mathcal{T}_{\text{freq}}$.
(3) **Manual Auditing**: Through manual auditing, users review and add missing topics to enhance $\mathcal{T}_{\text{llm}}$, yielding the curated topic set $\mathcal{T}_{\text{final}}$.

To further refine the knowledge base, *users may optionally reapply the above process*, treating $\mathcal{T}_{\text{final}}$ as the new input. This iterative procedure continues until no new related topics are identified, following the spirit of the alternate fix-point method [3]. Finally, repositories $\mathcal{R}'$ associated with topics in $\mathcal{T}'_{\text{final}}$ are extracted for graph construction.

2.2 Graph Construction

With the repository set $\mathcal{R}'$ and its associated metadata, users can construct a graph $G = (V, E)$, where V represents repositories and $E \subseteq V \times V$ denotes edges that are created based on user-specified criteria. DeepGit provides multiple edge definition strategies, allowing users to enable one or more criteria simultaneously:

- **Topic-Based Linking**: Repositories sharing a user-defined number of common topics from $\mathcal{T}_{\text{final}}$ are linked.
- **Contributor Overlap**: Repositories with sufficient contributor overlap, based on a user-defined threshold, are linked to reveal collaboration patterns.
- **Shared Organization**: Repositories maintained within the same GitHub organization are linked, highlighting internal structures within institutions or foundations.
- **Common Stargazers**: Repositories are linked if they share a sufficient number of stargazers [11].
- **Software Dependency**: Using Software Bill of Materials (SBOM) data [10], an edge is created when a declared dependency relationship exists between repositories.

Using the dimensions described above (which can also be easily extended with additional ones), users can construct graph G that reflect the specific relationships they are interested in. Once graph G is created, users can apply graph clustering algorithms (e.g., Louvain [19]) to identify communities. They can iteratively refine the edge definition until the resulting clusters align with their needs.

2.3 Subgraph Extraction & GraphRAG

However, navigating a large domain graph can still be overwhelming [26]. DeepGit addresses this challenge by enabling users to extract subgraphs based on various criteria. Users can filter the graph using repository metadata—such as commit activity, star count, programming language, or subtopic (e.g., Datalog within Logic Programming). Additionally, they can leverage graph properties, using clustering algorithms to select specific community clusters (Figure 2) or ranking methods such as PageRank [2] to extract high-importance nodes.

Additionally, DeepGit incorporates SubgraphRAG approach [17] by leveraging graph embeddings that integrate both textual content — including README files, source code, and research papers referenced by the repositories — with the graph's relational structure, i.e., user-defined edges between repositories. This hybrid representation allows DeepGit to support more effective question answering, not only through explicit graph relationships but also uncovering insights that may not be captured in the subgraph alone.

3 DEMONSTRATION & FUTURE WORK

Core Demonstration: A preliminary version of DeepGit is already available [25], enabling exploration of prebuilt graphs in selected domains (e.g., logic programming). We are currently in the process of implementing the proposed features. Our objective is to present a comprehensive demonstration at the workshop.

Future Work. As future work, we plan to incorporate metadata from platforms [21] beyond GitHub and integrate graph query languages (e.g., Cypher [7]) into the interface. This will offer users an alternative method for subgraph extraction. We also intend to conduct Human-Computer Interaction evaluations (e.g., System Usability Scale [16]) through interviews with domain researchers and research software engineers (RSEs) from open-source communities to assess and refine DeepGit.

REFERENCES

[1] Bilal Abu-Salih. 2021. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications* 185 (2021), 103076.

[2] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30, 1-7 (1998), 107–117.

[3] Ashok Chandra and David Harel. 1982. Structure and complexity of relational queries. *Journal of Computer and system Sciences* 25, 1 (1982), 99–128.

[4] Juri Di Rocco, Davide Di Ruscio, Claudio Di Sipio, Phuong Nguyen, and Riccardo Rubei. 2020. Topfilter: an approach to recommend relevant github topics. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–11.

[5] Peter Elmers. 2025. GitHub Public Repository Metadata. https://www.kaggle.com/datasets/pelmers/github-repository-metadata-with-5-stars

[6] George Fletcher, Jan Hidders, Josep Lluís Larriba-Pey, et al. 2018. *Graph Data Management*. Springer.

[7] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 international conference on management of data*. 1433–1445.

[8] Marvin Hofer, Daniel Obraczka, Alieh Saeedi, Hanna Köpcke, and Erhard Rahm. 2024. Construction of knowledge graphs: Current state and challenges. *Information* 15, 8 (2024), 509.

[9] Caroline Jay, Robert Haines, and Daniel S Katz. 2020. Software must be recognised as an important output of scholarly research. *arXiv:2011.07571* (2020).

[10] Maya Kaczorowski. 2024. Secure at every step: How GitHub's dependency graph is generated. GitHub Blog. https://github.blog/enterprise-software/secure-software-development/secure-at-every-step-how-githubs-dependency-graph-is-generated/

[11] Andrei Kashcha, Erik Bjäreholt, and Zachary Blackwood. 2024. *anvaka/map-of-github*. https://github.com/anvaka/map-of-github

[12] Daniel S Katz and Neil P Chue Hong. 2024. Special issue on software citation, indexing, and discoverability. *PeerJ Computer Science* 10 (2024).

[13] Hannah Kim, Dongjin Choi, Barry Drake, Alex Endert, and Haesun Park. 2019. TopicSifter: Interactive search space reduction through targeted topic modeling. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*.

[14] Simon Koch, David Klein, and Martin Johns. 2024. The Fault in Our Stars: An Analysis of GitHub Stars as an Importance Metric for Web Source Code. In *Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2024*.

[15] Anna-Lena Lamprecht, Leyla Garcia, Mateusz Kuzak, Carlos Martinez, Ricardo Arcila, Eva Martin Del Pico, Victoria Dominguez Del Angel, Stephanie Van De Sandt, Jon Ison, Paula Andrea Martinez, et al. 2020. Towards FAIR principles for research software. *Data Science* 3, 1 (2020), 37–59.

[16] James R Lewis. 2018. The system usability scale: past, present, and future. *International Journal of Human–Computer Interaction* 34, 7 (2018), 577–590.

[17] Mufei Li, Siqi Miao, and Pan Li. 2024. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *arXiv preprint arXiv:2410.20724* (2024).

[18] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph Retrieval-Augmented Generation: A Survey. arXiv:2403.08921 [cs.IR] arXiv preprint arXiv:2403.08921.

[19] Xinyu Que, Fabio Checconi, Fabrizio Petrini, and John A Gunnels. 2015. Scalable community detection with the louvain algorithm. In *2015 IEEE international parallel and distributed processing symposium*. IEEE, 28–37.

[20] Mark Raasveldt and Hannes Mühleisen. 2019. Duckdb: an embeddable analytical database. In *Proceedings of the 2019 international conference on management of data*. 1981–1984.

[21] MOSS repository contributors. 2024. *Map of Open Source Science (MOSS)*. https://github.com/numfocus/MOSS

[22] David Schindler, Erjia Yan, Sascha Spors, and Frank Krüger. 2023. Retracted articles use less free and open-source software and cite it worse. *Quantitative Science Studies* 4, 4 (2023), 820–838.

[23] Alexander Struck. 2018. Research software discovery: An overview. In *2018 IEEE 14th International Conference on e-Science (e-Science)*. IEEE, 33–37. https://doi.org/10.1109/eScience.2018.00016

[24] Shuai Wang, Zhiyuan Chen, Geli Fei, Bing Liu, and Sherry Emery. 2016. Targeted topic modeling for focused analysis. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1235–1244.

[25] Y. Xia, S.-R. Tsai, and M. Turk. 2025. DeepGit: Promoting Exploration and Discovery of Research Software. https://go.illinois.edu/deepgit.

[26] Vahan Yoghourdjian, Yalong Yang, Tim Dwyer, Lee Lawrence, Michael Wybrow, and Kim Marriott. 2020. Scalability of network visualisation from a cognitive load perspective. *IEEE transactions on visualization and computer graphics* 27, 2 (2020), 1677–1687.

[27] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. *Comput. Surveys* 56, 4 (2023), 1–62.