SALTBot: Linking Software and Articles in Wikidata

Jorge Bolinches[1], Daniel Garijo[1]

[1]Ontology Engineering Group, Universidad Politécnica de Madrid

Abstract

Research Software is becoming a recognized first class citizen to support and reproduce the results of scientific investigations. However, the link between software and their corresponding articles is often absent from Knowledge Graphs like Wikidata, thus making it challenging to retrieve implementations of existing papers. In this work we introduce the Software and Article Linker Toolbot (SALTBot), a bot for linking together GitHub code repositories with their corresponding scholarly articles in Wikidata based on their available citation information. In addition, SALTbot will automatically describe software entities with metadata. We have manually validated SALTbot in 500 code repositories with citation files, adding more than 30 new tools to the Wikidata Knowledge Graph.

Paper type: Resource
Code repository: https://github.com/SoftwareUnderstanding/SALTbot
Zenodo release: https://doi.org/10.5281/zenodo.8190001

1. Introduction

Research Software refers to the scripts, tools or computational pipelines developed throughout an investigation to support the main findings described in a scientific publication [1]. Research Software is becoming increasingly recognized as a research product,[1] and the scientific community has developed software citation principles [2] and citation formats [3] in order to recognize developers with the appropriate credit.

However, in most existing scholarly Knowledge Graphs to date (e.g., Open Alex [4], Wikidata [5], etc.) research software are not usually linked with their corresponding publications. This leads to three main problems: 1) lack of tool context, as articles usually complement research software with theoretical background, purpose and experimental results; 2) paper-implementation availability, as it becomes challenging to know which research papers include software for others to reuse; 3) author-developer credit, as some developers may have contributed to a software tool but not to its associated publication.

In this work, we address these issues by presenting SALTbot, a Software and Article Linker Toolbot, designed to find article and software entities in Wikibase instances in order to enrich and link them together. SALTbot takes as input one or multiple GitHub repositories and inspects them for references to existing articles in Wikidata. Then, if found, SALTbot will link software and their corresponding articles, creating a new software instance when necessary and enriching it with metadata. Our work includes two main contributions:

Wikidata'23: Wikidata workshop at ISWC 2023

j.bolinches@alumnos.upm.es (J. Bolinches); daniel.garijo@upm.es (D. Garijo)

[1] https://sfdora.org/read/

- A workflow designed to link software and articles with minimal user intervention, based on a manual analysis of dozens of software repositories with a link to a publication.
- SALTbot,[2] an end to end implementation of our workflow [6].

We have validated SALTbot manually by assessing its performance in over 500 GitHub repositories with citation files. As a result, we have added 33 new software instances, 104 metadata statements and over 40 new links between software tools and articles in the Wikidata Knowledge Graph.

The rest of the paper is structured as follows. We describe background knowledge in Section 2, introducing SALTbot in Section 3. Section 4 describes our efforts to validate SALTbot, Section 5 discusses the current limitations of our approach and Section 6 concludes the paper.

2. Background

In this section we briefly introduce the building blocks of SALTbot: 1) existing tools for automatically editing Wikibase [7][3] and Wikidata (Section 2.1) and 2) recent efforts towards standardizing software citation (Section 2.2).

2.1. Wikibase Bots

Bots in Wikibase are automated software applications capable of adding, modifying and removing statements from their corresponding Knowledge Graph. In Wikidata, these bots are developed by different communities to improve the completeness, accuracy and reliability of the information in the graph. Wikidata currently receives millions of monthly bot contributions, even surpassing author contributions during certain months.[4]

Bots are diverse, ranging from those which fetch data from external sources, adapt and integrate the data to the Wikidata model, those that add language tags, or those which improve qualifier descriptions of existing QNodes. There are more than 350 Wikidata officially approved bots,[5] and some of them enrich existing software tools in Wikidata. For example, Konstin's "Github to wikidata bot",[6] enriches entities with Github links with their software release metadata and project website. However, to the best of our knowledge there are no bots that analyze the actual contents of a code repository, such as the README and citation files, to link code repositories with bibliographical entities in Wikibase instances.

2.2. Software Citation Files

The scientific community has developed the Software Citation Principles [2], which led to the proposal of the Citation File Format [3] as a machine-readable metadata file for citing software

[2] https://github.com/SoftwareUnderstanding/SALTbot

[3] https://wikiba.se/

[4] https://stats.wikimedia.org/#/wikidata.org/content/edited-pages/normal|line|1-year|editor_type~group-bot*name-bot*user|monthly

[5] https://hgztools.toolforge.org/botstatistics/?lang=www&project=wikidata&dir=desc&sort=ec

[6] https://github.com/konstin/github-wikidata-bot

projects. Since GitHub implemented support for this representation,[7] an increasing number of developers have started to add these files in their repositories to obtain their corresponding credit (more than 10.000 to date). A CITATION.cff is a YAML file that usually contains the following information:

- Title: The title of the software project.
- Authors: The names of the software authors and contributors.
- Identifiers: A collection of identifiers (e.g., Digital Object Identifier) to uniquely identify the software project or its releases.
- License: The software's license information (e.g., MIT, GPL, Apache, etc.).
- Repository: The URL of the software's source code repository.
- Preferred citation: If the software project has already been described in a publication, this field describes the paper to be used to credit the software project's authors.

While the adoption for CFF files is growing, a wide number of researchers still credit articles describing their software contributions with plain BibTeX,[8] a common format used to reference articles in LaTeX publications (e.g., by adding their preferred citation in a README file).

3. Software and Article Linker Toolbot (SALTBot)

Figure 1 shows an overview of the architecture of SALTbot. Given one or multiple GitHub repository URLs as input, SALTbot finds software and scholarly articles related to each of the repositories in a Wikibase instance, analyzes the existing relationships between these entities, and introduce new links between them to complete a bidirectional relationship between articles and software, creating and characterizing new software instances when they do not exist in the graph. SALTbot is divided in the following modules:

- **Orchestrator**: The main module of SALTbot. It deals with the Wikibase configuration, proccesses the input, sends the parsed metadata to the handler module for each repository and calls the updater module to introduce data to the graph.
- **SOMEF**: We reuse the Software Metadata Extraction Framework [8, 9], a tool that produces a JSON with relevant metadata from both the README and CITATION.cff files contained in code repositories when provided with a repository URL.
- **Handler**: This module is in charge of sending and receiving data from all of SALTbot's modules in order to figure the necessary statements to add to the graph for one repository
- **Searcher**: Finds possible article and software entity QNodes from the graph based on the metadata extracted by SOMEF.
- **Analyzer**: Assesses the existing relationships between all the articles and software found and prints them to the user
- **Statement Definer**: Creates a list of statements and entities to create in order to link an article and software, asking for user validation if needed.
- **Updater**: Uploads statements to a target Wikibase Knowledge Graph in bulk.

[7] https://docs.github.com/en/repositories/managing-your-repositorys-settings-and-features/customizing-your-repository/about-citation-files

[8] https://www.bibtex.org/About/

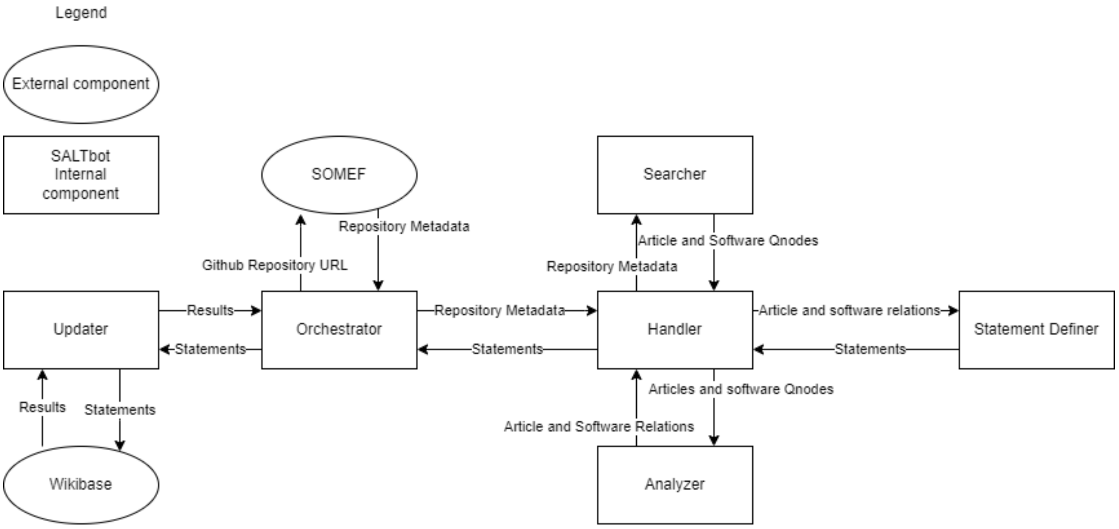


Figure 1: SALTbot Architecture. SALTbot reuses SOMEF [8] for software citation and metadata extraction, and the Wikibase API to retrieve potential existing paper candidates to link to software components.

3.1. SALTbot Assumptions

We designed SALTbot to be compatible with any Wikibase instance holding software tools and articles. However, since every Wikibase instance may have different node identifiers, SALTbot assumes that the Wikibase modeling is similar to the Wikidata modeling in terms of the existing entities, albeit their respective identifiers (QNodes) may be different. Therefore, the first step to configure the bot is to query the graph to find the necessary QNodes and PNodes needed to operate. The mandatory minimum items that SALTbot needs are the following:

- **“instance of” property PNode**: property used to check the existence of items of a specific type (both software and articles must be instances of something).
- **“main subject” property PNode**: property used to link an article with its specific software tool.
- **“described by source” property PNode**: property used to link a software with its specific article. This is the current practice by which existing articles and tools are currently linked in Wikidata, and hence we followed it.
- **“Scholarly article” entity QNode**: entity used to find scholarly articles in the graph. Every article must be an instance of this entity.
- **“Software category” entity QNode**: meta-class used to find software in the graph. Every software tool must be recursively an instance of a software category.
- **“Software” entity QNode**: entity used to add the mandatory ”instance of something” statement to the software created by SALTbot.

If one or more of these items are missing from the target KG, SALTbot will not run. Additionally, SALTbot queries the graph for some optional information to better characterize the software entities. These additional elements are:

- **“source code repository URL” PNode**: property used to link a software entity with its code repository URL.
- **“Free software” entity QNode**: entity used to add the mandatory “instance of something” statement to the software node created by SALTbot (if the software tool has a free license in the GitHub repository). If a software project does not have a free license, we categorize it as “Software”.
- **“programmed in” PNode**: property used to define the programming language in which a software entity is developed.
- **“download link” PNode**: property used to link a software entity with its specific article.
- **“copyright license” Pnode**: property used to specify the type of software license used by a software entity.
- **“version control system” and “web interface software” PNodes**: properties used as qualifiers when describing the source code repository of a software project.
- **“Git” and “GitHub” QNodes**: entities used with the two previous properties to add qualifiers to assign a source code repository URL to a software entity.

3.2. Workflow

Figure 2 shows an overview of the decision making workflow followed by SALTbot. We start from a code repository URL. The first step is to get all the relevant metadata using SOMEF, which generates a JSON file with all the metadata in the code repository. In particular, SOMEF detects citation information with one or more preferred citations from the authors in both BiBtex and CFF (in YAML) formats, which are the ones we focus on.

SALTbot then calls the Searcher module to parse all the BiBtex and YAML citations in order to find titles for scholarly articles, as well as other information such as the Digital Object Identifier (DOI) of an article, if present. Once the candidate titles are extracted, the bot will query the target KG for entities which are instances of scholarly article and whose label is the title from the citation, filtering by the corresponding DOI. If no articles are found using the parsed citation, the Searcher module will attempt to find scholarly article entities using the GitHub repository name. This strategy is less restrictive and consequently produces more vague results that need to be manually verified, but usually retrieves promising article candidates with a reference to the software project in their title.

The same process is also repeated to find software tools: we search for entities which inherit from the meta-class “software category” and whose label is similar to one of the parsed titles. These entities are then filtered out by comparing their source code URL repository with the URL provided to SALTbot.

We use DOIs to filter articles. If no DOIs are found in the parsed citation, or if these DOIs do not match those found in the article entities, SALTbot will require manual validation from users in order to select one of the articles found to proceed with the execution. Similarly, the repository URL allows identifying whether the software entities found correspond to the software component in the target repository. If no software candidates are found through their URL, SALTbot will ask to choose one of the found software components or to create a new one.

Next, the Analyzer module gathers all the previously existing relationships between the article and software in the graph. Using the Analyzer output, the Statement Definer will create

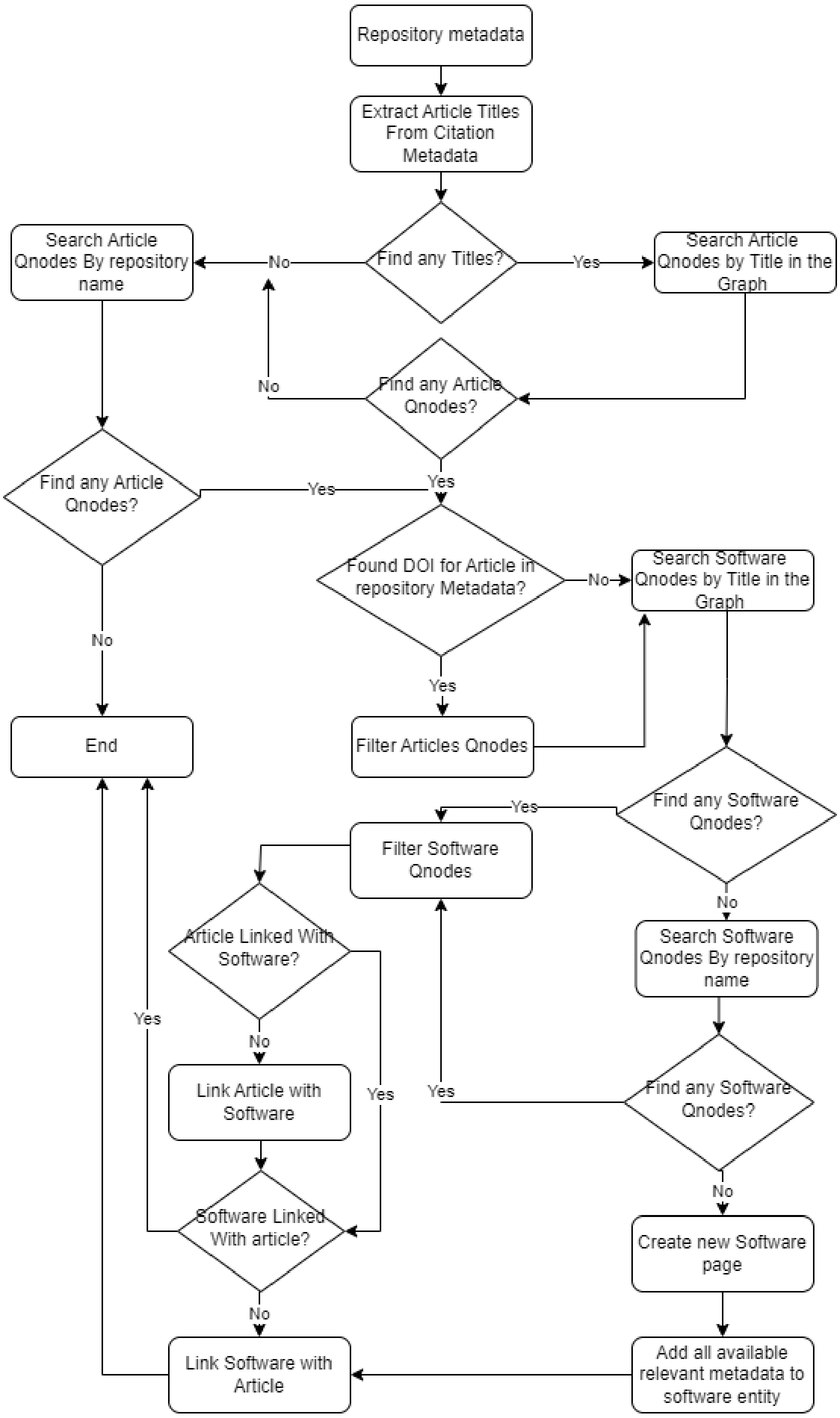

Repository metadata
Extract Article Titles From Citation Metadata
Find any Titles?
No
Yes
Search Article Qnodes By repository name
Search Article Qnodes by Title in the Graph
Find any Article Qnodes?
No
Yes
Find any Article Qnodes?
Yes
No
Found DOI for Article in repository Metadata?
No
Yes
Search Software Qnodes by Title in the Graph
End
Filter Articles Qnodes
Find any Software Qnodes?
Yes
No
Filter Software Qnodes
Article Linked With Software?
Yes
No
Search Software Qnodes By repository name
Link Article with Software
Yes
Yes
Find any Software Qnodes?
No
Software Linked With article?
No
Create new Software page
Link Software with Article
Add all available relevant metadata to software entity

Figure 2: SALTbot main decision making workflow. Starting from a code repository URL, SALTbot will extract the main citation metadata using SOMEF, search for the corresponding paper in Wikidata and then will attempt to identify whether the software already exists in the target KG. If the software tool exists, SALTbot will link it to the paper. If it does not, the bot will create a new page for the tool.

a list with the necessary statements to completely link the article and software entities. These statements are included in one of the following categories:

- If no software was found, SALTBot creates a new item which will be an instance of "software" and whose label will be the GitHub's repository name. These software pages are further enriched by using the repository's metadata such as the license, the source code repository URL, the programming languages in which the repository's code is written and the fact that it uses Git as a version control system. Additionally, if the license detected is a open license and the "Free software" QNode was found in the graph, the new software item will be characterized as free software (i.e., "software distributed under terms that allow users to freely run, study, change and distribute it and modified versions"[9]).
- If the article is not linked to its corresponding software project, SALTBot adds a new statement to the article using the "main subject software" PNode.
- If the software project is not linked to the article, SALTBot adds a new statement using the "described by source article" PNode.

Once the number of statements in the list is higher than a batch size defined by users, all statements are loaded to the target KG using the Updater module. This process is repeated by SALTbot any number of times for each of the code repository URLs provided as input.

3.3. Uploading statements to Wikidata/Wikibase

SALTbot can be used against a local Wikibase instance or to upload new contents in Wikidata. We build upon Wikibase Integrator,[10] a Python library designed to read and write data into Wikibase while solving compatibility and integration problems between different Wikibase instances.

In order to edit a specific Wikibase instance, SALTbot provides the necessary wrappers to automatically configure Wikibase Integrator. The following information is required for configuring SALTbot:

- A valid username and password in the desired Knowledge Graph
- The MediaWiki API URL of the target graph
- The Knowledge Graph SPARQL endpoint
- The Wikibase URL of the graph.

Any of the three last configuration items default to the corresponding Wikidata values if left unchanged. SALTbot will process each of the repositories in a semi-autonomous manner, asking for validation when necessary to decide which article or software to use if multiple candidates have been found.

[9] https://www.wikidata.org/wiki/Q341
[10] https://github.com/LeMyst/WikibaseIntegrator

4. SaltBot validation

In order to assess the correct behaviour of SALTbot, we tested the bot by gathering 500 repositories from GitHub with a “CITATION.cff” file using the GitHub API[11] and validating the results manually. The rationale behind our approach is to ensure the selection of code repositories with at least a suggested pointer to a publication.

Our selected 500 repositories[12] presented the following characteristics before our bot assessment was completed:

- 378 repositories had one or more mentions to scholarly articles (some refer to code deposits in archives like Zenodo).
- 46 repositories had their corresponding scholarly article page in Wikidata.
- 35 repositories had their corresponding software page in Wikidata.
- 12 scholarly article entities were previously linked through the property ”main subject” to their corresponding software entity.
- 5 software entities were previously linked through property ”described by source” to their corresponding article entity.

In order to perform the validation of SALTbot, we created a bot page[13] and a new username in Wikidata to keep a record of the contributions performed with the tool. These contributions can be seen in https://www.wikidata.org/wiki/Special:Contributions/SALTbotDev. Figure 3 shows an example with one of our contributions to Wikidata, by linking a newly added tool to an existing article.

After our manual validation, SALTbot enriched Wikidata with the following knowledge:

- 33 newly created software entities.
- 104 new software metadata statements.
- 34 scholarly articles linked with their corresponding software entity (this number includes articles whose software has been created in order to link them).
- 43 software entities linked with their corresponding scholarly articles (this number includes those software QNodes newly created by SALTbot in order to link them).

While validating SALTbot, we noticed how our approach blends in with the Wikidata ecosystem. Shortly after creating new software entities, other bots like Github-wiki-bot started improving existing page descriptions with their release contents (184 statements regarding software version identifiers and official pages were added to our newly created software entities).

5. Discussion

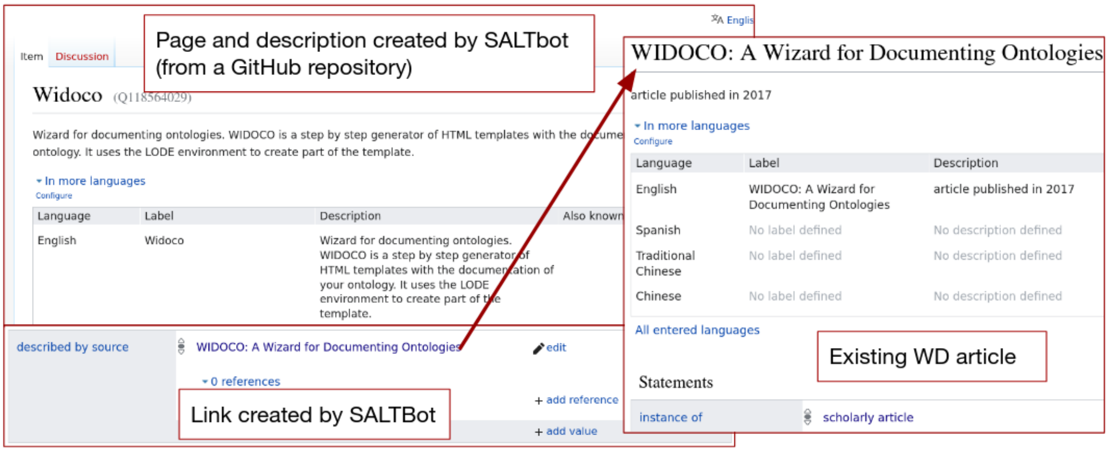
On 2018, GitHub reached the staggering milestone of holding more than a hundred million code repositories.[14] In comparison, ten thousand repositories with a CITATION.cff seems like a very

[11] https://api.github.com/

[12] Available at: https://github.com/SoftwareUnderstanding/SALTbot/blob/main/WikidataFindings.csv

[13] https://www.wikidata.org/wiki/User:SALTbot

[14] https://github.blog/2018-11-08-100M-repos/



Page and description created by SALTbot (from a GitHub repository)

Widoco (Q118564029)

Wizard for documenting ontologies. WIDOCO is a step by step generator of HTML templates with the docume ontology. It uses the LODE environment to create part of the template.

Language	Label	Description	Also known
English	Widoco	Wizard for documenting ontologies. WIDOCO is a step by step generator of HTML templates with the documentation of your ontology. It uses the LODE environment to create part of the template.	

described by source: WIDOCO: A Wizard for Documenting Ontologies

Link created by SALTBot

WIDOCO: A Wizard for Documenting Ontologies

article published in 2017

Language	Label	Description
English	WIDOCO: A Wizard for Documenting Ontologies	article published in 2017
Spanish	No label defined	No description defined
Traditional Chinese	No label defined	No description defined
Chinese	No label defined	No description defined

Existing WD article

Statements

instance of: scholarly article

Figure 3: An example result from SALTbot for Widoco, a tool for documenting ontologies. In this case, the bot creates both the page for the software tool, describes it with metadata (description, license, code repository, etc.) and links it to the existing article in Wikidata.

small percentage, but the number of repositories containing citation is slowly growing. This number also suggests that many research software projects in GitHub may be lacking a citation file to indicate the correct way of citing the software in a machine-readable way. We believe that continuously running SALTBot will increasingly enrich Wikidata with links between articles and software.

In addition, SALTBot enriches software entities with existing metadata by following current Wikidata practices for modeling software. Incorporating additional metadata elements (e.g., from Codemeta[15]) may help to further increase the usefulness of our contributions in the target Wikibase KG.

Our approach is orthogonal to the efforts of other platforms like Papers With Code[16] or Arxiv,[17] which scan data/software availability statements or whole publications (manually or automatically) to find the corresponding associated code repositories. Instead, we analyze code repositories assessing the direct citation preference declared by authors.

As for limitations, our main challenge is unambiguously identifying scholarly articles in Wikibase instances. Our approach attempts to use the article's DOI to identify it in the graph, however, this presents the following issues:

- Not all repositories have an explicit reference to the article's DOI.
- Not all scholarly articles are currently linked to their corresponding DOI in Knowledge Graphs.
- Scholarly articles may have other identifiers, such as an arXiv ID or a Zenodo ID, which may also be missing in the citation or README files.

Currently we address these issues by asking for user input, which hinders full process

[15] https://github.com/codemeta/codemeta/blob/master/crosswalks/Wikidata.csv
[16] https://paperswithcode.com/
[17] https://arxiv.org/

automation for some repositories. Relying on external sources like OpenAlex[18] and Crossref[19] may help address this problem.

Finally, SALTbot relies on linking software to publications that already exist in Wikibase/Wikidata. Papers that are not part of the KG are currently out of the scope of the application. However, as shown in our manual validation, a significant number of tools belong to articles that are not currently part of the KG, so creating new article pages may be beneficial to include more tool implementations.

6. Conclusions and Future Work

In this paper we introduced SALTbot, our effort towards enriching Wikibase/Wikidata with the software implementations of existing research articles. We have manually validated our approach with 500 code repositories, resulting in 33 new software entities and over 40 new software-paper links. SALTbot contributions are integrated within the Wikidata ecosystem, with other bots building and expanding on our work. We believe that, as developers continue adopting best software citation practices, SALTbot will become increasingly useful to the Wikidata and scientific communities.

Our future work includes three main improvements. First, we are currently running SALTbot on nearly ten thousand additional repositories with CFF files, manually validating the results when needed. Second, we are exploring running the bot on repositories with other types of citation files (e.g., through BiBtex), which are also detected by SOMEF. Finally, we will explore automatically creating scholarly article entities in the same way we do with software entities. However, this feature requires further research, especially when determining how to correctly characterize scholarly articles in Knowledge Graphs (avoiding possible duplicates), how much article metadata can be obtained from the citation found in a code repository, and how to assess the validity of the final results.

Acknowledgments

This work was supported by the Comunidad de Madrid under the Multiannual Agreement with Universidad Politécnica de Madrid (UPM) in the line Support for R&D projects for Beatriz Galindo researchers, in the context of the V PRICIT (Regional Programme of Research and Technological Innovation) and through the UPM call Research Grants for Young Investigators.

References

[1] N. P. Chue Hong, D. S. Katz, M. Barker, A.-L. Lamprecht, C. Martinez, F. E. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, et al., FAIR Principles for Research Software (FAIR4RS Principles), 2022. doi:10.15497/RDA00068.

[18] https://openalex.org/

[19] https://www.crossref.org/

[2] A. M. Smith, D. S. Katz, K. E. Niemeyer, Software citation principles, PeerJ Computer Science 2 (2016) e86.

[3] S. Druskat, J. H. Spaaks, N. Chue Hong, R. Haines, J. Baker, S. Bliven, E. Willighagen, D. Pérez-Suárez, O. Konovalov, Citation File Format, 2021. doi:10.5281/zenodo.5171937.

[4] J. Priem, H. Piwowar, R. Orr, Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts, 2022. arXiv:2205.01833.

[5] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.

[6] J. Bolinches, D. Garijo, SoftwareUnderstanding/SALTbot: SALTbot 0.0.1: First stable release, 2023. URL: https://doi.org/10.5281/zenodo.8190001. doi:10.5281/zenodo.8190001.

[7] D. Diefenbach, M. D. Wilde, S. Alipio, Wikibase as an infrastructure for knowledge graphs: The eu knowledge graph, in: A. Hotho, E. Blomqvist, S. Dietze, A. Fokoue, Y. Ding, P. Barnaghi, A. Haller, M. Dragoni, H. Alani (Eds.), The Semantic Web – ISWC 2021, Springer International Publishing, Cham, 2021, pp. 631–647.

[8] A. Kelley, D. Garijo, A Framework for Creating Knowledge Graphs of Scientific Software Metadata, Quantitative Science Studies (2021). doi:10.1162/qss_a_00167.

[9] A. Mao, D. Garijo, S. Fakhraei, Somef: A framework for capturing scientific software metadata from its documentation, in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 3032–3037. doi:10.1109/BigData47090.2019.9006447.