Journal of Informetrics 15 (2021) 101139

Contents lists available at ScienceDirect

Journal of Informetrics

journal homepage: www.elsevier.com/locate/joi

In-code citation practices in open research software libraries

Abdulkareem Alsudais

College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj, 11942, Saudi Arabia

ARTICLE INFO

Article history:
Received 10 August 2020
Received in revised form 3 January 2021
Accepted 26 January 2021

Keywords:
Research software
In-code citation
Software citation
Open software
Citation networks

ABSTRACT

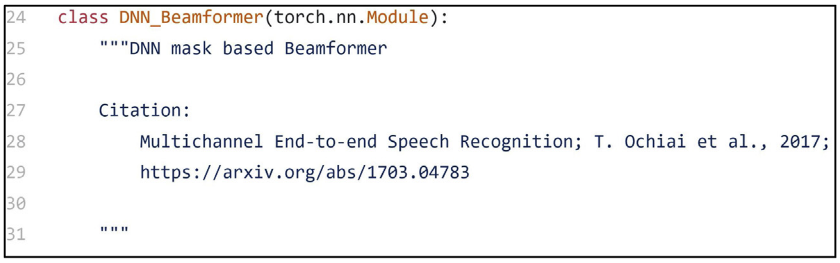
While several aspects of open research software libraries have been studied, their in-code citation practices remain an unexplored area. In-code citations are citations of published scientific contributions added in the programming source code of research software libraries. In this paper, 73 such libraries are analyzed to determine the availability and consistency of in-code citations and reference sections. Findings indicate that 54 (73.9 %) of these libraries have at least one in-code citation. However, 47 had at least one incomplete citation and 89.3 % of libraries with several citations used multiple formats for citations. For reference sections, 36 of the libraries investigated in this study had at least one section. Still, inconsistencies in formats were also present as 83.3 % of the libraries with two or more sections used multiple formats for the sections, which may prevent automated programmers from indexing and collecting the list of references. Most importantly, this study investigates the availability of a systematic method that allows for the linking of references and in-code citations. Findings indicate that only six of the libraries had such a method, although many did not fully implement the method.

1. Introduction

In recent years, scientists from several academic disciplines have contributed to the scientific community through the release of open research software libraries. These libraries are open source packages available freely online and developed by researchers to automate, organize, and implement solutions related to research problems important to one or more scientific community. Researchers' sharing of their research software is motivated by a desire to improve the software's "design, development, use, testing, and patching" (Park & Wolfram, 2019). The software is used for tasks such as data collection, cleaning, analysis, and automation (AlNoamany & Borghi, 2018). Moreover, research software can be "employed in the scientific discovery process or a research object itself" (Hasselbring, Carr, Hettrick, Packer, & Tiropanis, 2020). To "seek greater academic recognition of their work" (Li, Chen, & Yan, 2019), authors of research libraries often publish academic papers that describe their software papers (Pianosi, Sarrazin, & Wagener, 2020). In many cases, the primary scientific contribution of these papers is actually the research software. GitHub, a software development tool and version control system with an online platform commonly used in open source projects, is often used by researchers to host and provide a reliable access to their libraries (Cosentino, Cánovas Izquierdo, & Cabot, 2017). GitHub and its hosted repositories have been used in research papers to study several topics such as open collaboration (Cheng, Zhang, Yang, & Yan, 2020), software engineering issues (Catolino, Palomba, Zaidman, & Ferrucci, 2019; Chen et al., 2020; Zou, Xuan, Xie, Chen, & Xu, 2019), and recommendation systems that link papers to repositories (Shao et al., 2020). While the methods for citing research software libraries in scientific papers has gained interest recently (Chassanoff et al., 2018; Li et al., 2019; Park & Wolfram, 2019; Smith, Katz,

E-mail address: A.Alsudais@psau.edu.sa

https://doi.org/10.1016/j.joi.2021.101139
1751-1577/© 2021 Elsevier Ltd. All rights reserved.


```
class DNN_Beamformer(torch.nn.Module):
    """DNN mask based Beamformer

    Citation:
        Multichannel End-to-end Speech Recognition; T. Ochiai et al., 2017;
        https://arxiv.org/abs/1703.04783

    """
```

Fig. 1. An example of an in-code citation from the library ESPnet-ST (Inaguma et al., 2020). The citation is placed inside a multiline comment section.

Niemeyer, & FORCE11 Software Citation Working Group, 2016), the issue of how these libraries cite other scholarly works in their programming code remains unexplored.

In-code citations are citations of published scientific contributions added in the programming source code of software libraries. Fig. 1 shows one example of an in-code citation. While there are many kinds of software libraries, the in-code citations studied in this work are limited to those used in open research software libraries. The primary motivation for studying such citations is to discover how they are currently being used and how to make them visible for automated retrieval. While many libraries currently include citations, it is possible that they use a variety of methods for adding in-code citations because of a lack of uniform standards. Thus, these in-code citations remain hidden, invisible, and undiscoverable. Therefore, studying how they are being used can eventually lead to an understanding of how to make them visible to web crawlers and then available for macro indexing. Moreover, studying them may help researchers who want to find out whether certain methods have been implemented in the programming codes of open research software libraries. It can also help connect researchers to others who are working on similar projects. Additionally, studying in-code citations may help to reveal certain best practices. Thus, a second motivation is to identify best practices for in-code citations in open research software libraries and then recommend them if successfully found. Finally, it is important to note that finding these in-code citations does not immediately translate to a suggestion to include them in databases that currently track and link citations such as Google Scholar or Web of Science. It is possible that once discovered, dedicated services that only track in-code citations can be developed.

The objective of this paper is to investigate in-code citation practices in open research software libraries. More specifically, I will focus on how these libraries cite other papers or research software in their lines of source code. This study quantifies the number of packages that include citations in their code, the number that have reference sections, and the number that include a bibliography. For the purposes of my inquiry, it is salient that the bibliographies included in these packages the references used in the library in a systematic way that allows for automated linking of references and in-code citations. Finally, this study also examines inconsistency in the in-code citations and reference sections used by each library.

In this paper, I aim to answer the following research questions:

- RQ1: Do open research software libraries cite published papers or other research software in their lines of programming code?
- RQ2: When libraries cite scholarly work, do they include the full reference, comprised of the names of all the authors, the title, the publication name, and the year?
- RQ3: Do libraries use several formats for citations when citing scholarly work?
- RQ4: Do libraries list references used in a code block by including designated reference sections added to the top of files or programming objects like classes and functions?
- RQ5: When libraries have multiple designated reference sections, do they all appear in a uniform format?
- RQ6: Do libraries include a document listing all the referenced scientific work cited in the library along with a mechanism to easily connect the references to the exact location where they were used?

The rest of the paper is organized as follows: Section 2 includes a review of relevant scholarship related to open research software libraries. Section 3 describes the dataset and research methodology of the current study on in-code citation practices. Section 4 contains the results of the study. Sections 5 and 6 present a discussion of my findings and a conclusion.

2. Literature review

2.1. Citations and research software

While academic citation issues in regard to open research software libraries have been studied, to the best of my knowledge, no previous work has addressed the problem of how these libraries cite other scholarly work in their source code. One similar area of research that sheds light on the current study is the investigation into the ways in which open research software libraries are cited in scholarly published papers (Katz et al., 2019; Li, Lin, & Greenberg, 2016; Li, Yan, & Feng, 2017; Pan, Yan, Cui, & Hua, 2018; Pan, Yan, Cui, & Hua, 2019). For example, in one paper, the authors analyzed 13 top journals in the information science domain and discovered that papers tend not to cite software using the citation recommended by the developers of the libraries, or that they fail to provide any formal citations for the software used in their studies at all (Pan et al., 2019). Similar findings were reported by Li et al. (2017), who examined the programming language R and many of its libraries and discovered inconsistencies in how they had been cited in papers. In another study, the authors provided an in-depth analysis of several ways in which the popular R package “lme4” had been cited in papers (Li et al., 2019). In a large-scale study, Park and Wolfram (2019) discovered that most research software rarely gets cited. In yet another study, Li and Yan (2018) created a new method for discovering the R libraries used in published research papers, and employed it to rank the packages most commonly used. In a large-scale study on how research software is cited, Sandt et al. (2019) found that “current citation practices and recommendations do not match proposed citation standards.” In another paper, Druskat (2020) discussed several challenges to do with the current model of research software citations. These papers call attention to some of the issues related to citations in open research software libraries, including the limited number of citations most research software libraries receive, as well as the presence of incorrect software library citations. This paper studies the similar yet currently unexplored area of in-code citation practices employed by these libraries. A widespread implementation of proper standards for in-code citations may help remedy some of these issues explored by other scholars regarding open research software libraries.

2.2. *Research software issues*

Several authors have also studied problems related to research software in general. The sustainability of research software has recently been an area of concern (Anzt et al., 2020; Chassanoff & Altman, 2020; Chassanoff et al., 2018; Druskat & Katz, 2018). Similarly, multiple attempts have been made to set minimum requirements and best practices for research software (Alliez et al., 2020; Hasselbring et al., 2020; Jiménez et al., 2017; Lamprecht et al., 2020; Smith et al., 2016). Wilkinson et al. (2016) proposed four principles to guide appropriate research data practices: The data need to be “findable, accessible, interoperable, and reusable.” They indicated that “the principles apply not only to ‘data’ in the conventional sense, but also to the algorithms, tools, and workflows that led to that data.” Several authors have investigated the importance of sharing research software. One paper focusing on research software developed for computational materials science discussed several challenges relating to the verification and validation of such software (Vogel, Druskat, Scheidgen, Draxl, & Grunske, 2019). Heumüller, Nielebock, Krüger, and Ortmeier (2020)) studied all the papers published at the International Conference on Software Engineering (ICSE) and discovered that only 58.5 % of the papers with a software made it publicly and freely available. The importance of authors making available online the software code used in their academic papers has recently been emphasized in several academic disciplines (Celi, Citi, Ghassemi, & Pollard, 2019; Ince, Hatton, & Graham-cumming, 2012). These efforts are motivated by the need to validate and reproduce results found in published papers. Scholars from several research domains have studied issues that are specific to their area of research. For example, in machine learning, scientists created paperswithcode.com, a website that automatically links papers to their GitHub repositories, aggregates results reported in the papers, provides leaderboard tables classified by the tasks they target, and ranks state-of-the-art methods for each task (Kardas et al., 2020). Similarity, Shao et al. (2020) created paper2repos, a system that attempts to link scientific papers to projects available on GitHub using machine learning techniques.

3. Methodology

3.1. *Dataset*

3.1.1. *Web of science*

The dataset consists of open research software libraries. To control for differences in code-commenting practices, I only considered software packages with most of the source code written in the programming language Python. I used three sources for the dataset with the intent of providing broader context for in-code citation practices. The three sources and the process used to find the libraries are in Fig. 2. The first source is Web of Science (WoS), which indexes published academic papers and conference proceedings from all academic disciplines. Initially, I considered using Google Scholar. However, I selected WoS instead because Google Scholar includes many citations from sources that are not academic journals (Martín-martín, Orduna-malea, Thelwall, & López-cózar, 2018). The objective of using WoS was to identify libraries that have been used extensively by researchers regardless of their research domains. To find open research software libraries, I searched for the word “python” under the “topic” label. The following lists of words were also added to a second “topic” label: "software," "tool," "library," "package," "system," or "framework." Therefore, the full search phrase was:

TOPIC: (python) AND TOPIC: ("software" OR "tool" OR "library" OR "package" OR "system" OR "framework").

Searching for the word “python” alone resulted in several false positive cases, returning papers irrelevant to the research topic. These included papers on the snake species of the same name, as well as papers on general topics related to the programming language Python, such as how to optimize it or how to best teach it. After running the search phrase, I organized the results by the number of citations each paper received in order to identify the most-used research software packages. I am aware that this sorting procedure may tend to favor larger research disciplines, which is a potential constraint

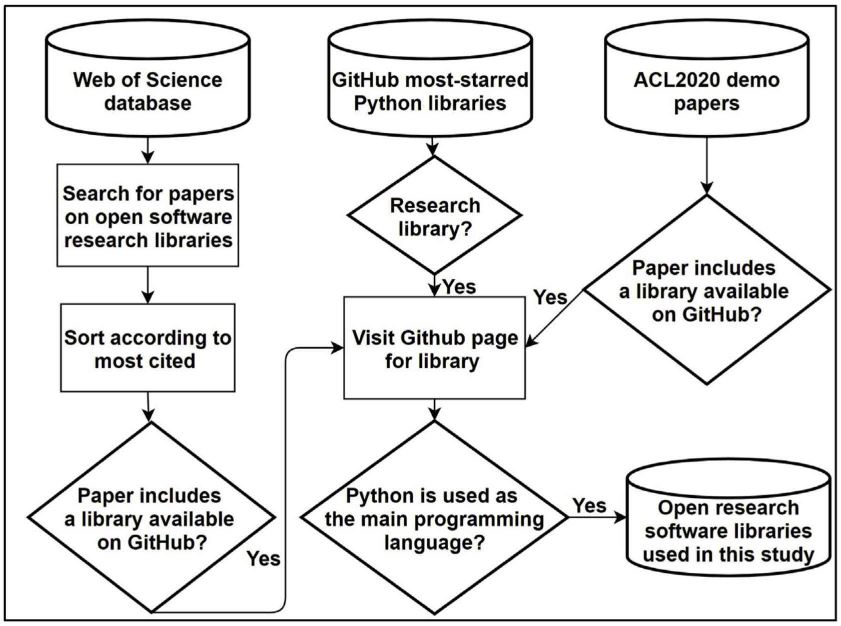
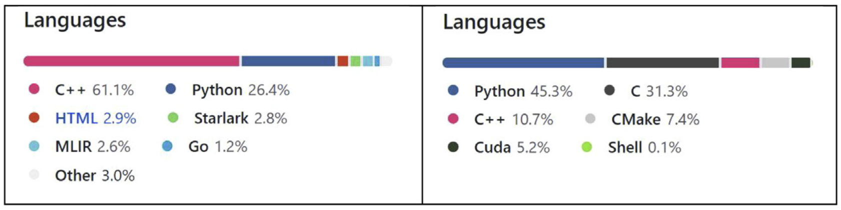

Web of Science database
Search for papers on open software research libraries
Sort according to most cited
Paper includes a library available on GitHub?
Yes
GitHub most-starred Python libraries
Research library?
Yes
Visit Github page for library
Python is used as the main programming language?
Yes
Open research software libraries used in this study
ACL2020 demo papers
Paper includes a library available on GitHub?
Yes

Fig. 2. A flow chart illustrating the process used to find the open software research libraries used in this study. Libraries with any ñoänswers were not included in the study.


Languages

● C++ 61.1% ● Python 26.4%
● HTML 2.9% ● Starlark 2.8%
● MLIR 2.6% ● Go 1.2%
● Other 3.0%

Languages

● Python 45.3% ● C 31.3%
● C++ 10.7% ● CMake 7.4%
● Cuda 5.2% ● Shell 0.1%

Fig. 3. Examples of the number of lines of code as indicated by GitHub for two libraries. The library on the left, TensorFlow, was removed because C++ was the most used language while the one on the right, tomopy, was kept since Python was the most used language.

of this first set. However, my objective was nevertheless to identify the most-used libraries regardless of any other factor that could have impacted the ranking.

For each of the papers processed, I examined the availability of an open software library in GitHub. Accordingly, papers without an open software library were removed. When a repository in GitHub was found, I also examined the programming language most used in the source code of the library. GitHub ranks the programming languages used in a library according to the number of lines used per language. Papers where the programming language with the highest number of lines was not Python were removed. Fig. 3 shows examples from two libraries and their programming languages as indicated by GitHub. The objective was to find 30 open software libraries currently available on GitHub and written in Python. This number of packages was found after reading the first 87 papers.

To identify if a paper had an open library available on GitHub, I searched for references to an open library, as well as a URL for the library, in the text of the paper. If one was not found, I input the paper name as a query on GitHub. If that query did not produce results, I made a final attempt to find an open library by searching Google with the title of the paper along with the word "GitHub." When I found a GitHub project for the library, I viewed the content of the "read.me" file, which typically includes textual descriptions of the project. The purpose of this was to confirm that the package was written by the authors of the paper and not by unaffiliated developers who had written and uploaded their own implementation for a method, algorithm, or idea introduced in the paper. I observed during this step that many research software libraries use names that are not unique, and thus these same names are used by other packages and in some cases by other research libraries. For some of the packages, an updated version of the software was available, and it was used in this study instead of the original version introduced in the referenced papers. The research domains provided by WoS suggest that these 30 packages were written for a variety of disciplines such as astronomy & astrophysics (Collaboration et al., 2013), biochemistry & molecular biology (Sukumaran & Holder, 2010), chemistry (O'Boyle, Tenderholt, & Langner, 2007), and computer science (Pedregosa et al., 2011).

3.1.2. GitHub Most-starred Libraries

The second source of open research software libraries analyzed in this study was a list of the most-starred Python libraries hosted by GitHub (Github, n.d.-a). GitHub allows registered users to "star" libraries, presumably so that they can easily access later or show their appreciation for the libraries. The most-starred list is generated by GitHub when one views the Python "topic" in the website. Since GitHub is popular among software developers and individuals in the computer science field, the most-starred libraries are likely to be related to this research domain. Thus, compared to libraries found using WoS, this list provides different types of packages, related to machine learning, natural language processing, computer vision, and other areas of study within the computer science discipline. Moreover, since the selection is based on the number of stars, this list presumably generates research software libraries that are commonly used by industry professionals. It was observed that some of the packages from the WoS list that are for research domains other than computer science had very few stars (i.e., a small number of GitHub users had liked/favorited the library) compared to those found using this second source.

For each package listed, I examined the content to determine if the library was for a research software. Many of the packages listed were related to web development or learning materials and thus were not included in this study. Several packages that are popular and ranked high on the list such as TensorFlow and PyTorch were also not included because GitHub indicated that most of their source code was not written in Python ("PyTorch, " n.d., "TensorFlow, " n.d.). Finally, only packages with published research papers were included. There were many packages that were not related to research and thus they were not introduced in published papers as scientific contributions. These include many web development libraries, for example. The result of this process was a list of 30 libraries covering several areas of research within the computer science domain, such as machine learning (Lemaitre, Nogueira, & Aridas, 2017; Pedregosa et al., 2011), natural language processing (Bird, Klein, & Loper, 2009; Mckinney, 2010; Qi, Zhang, Zhang, Bolton, & Manning, 2020), and computer vision (Bulat & Tzimiropoulos, 2017; Zhu, Liu, Lei, & Li, 2017). Two of the libraries found, scikit-learn (Pedregosa et al., 2011) and scikit-image (Van Der Walt et al., 2014), also appear in the first list. Thus, the number of new libraries after excluding those two is 28 and the total number of libraries from the first two sources is 58.

3.1.3. ACL2020

While the first two lists focused on packages from the computer science and other research domains, this third and final list targets packages recently published in the proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL2020). ACL2020 is the premier academic conference for researchers working on natural language processing and computer linguistics. The papers in this list were all accepted to the system demonstrations track, which included 43 papers in total. The call for papers stated that "submissions may range from early research prototypes to mature production-ready systems. Of particular interest are publicly available open-source or open-access systems" (ACL, 2020). Many of the accepted papers did not include open research software libraries and were not considered in this study. Several described systems were available as online demos and did not include an accessible open source code. Others were not written in Python. One library had removed the URL link for the GitHub repository added to the paper. Accordingly, I found that 16 of the 43 papers included open research packages that met the selection criteria. The popularity of the libraries as measured by the number of citations or stars was not a selection factor for this third list. Moreover, all the papers were published in 2020, whereas the first two lists include much less recent scholarship. One library, Stanza (Qi et al., 2020), is also present in the second list. In summary, when the three lists are combined, the total number of libraries when counting repeated libraries only once is 73 libraries. Table 1 lists all the libraries that were found for the three sources.

3.2. Analytic approach

I classified the steps followed to answer the research questions into two groups. The first group of research questions was concerned with the availability of in-code citations, designated reference sections, and complete bibliographies. The second group was concerned with the contents of the references used and included an assessment of the completeness of the citations, the consistency of the formats of in-code citations, and the consistency of designated reference sections. A summary of the process used to answer the research questions is in Fig. 4. All the libraries were tested for the items in the first group, while only those with in-code citations and designated reference sections were tested for the items in the second group. Table 2 summarizes the steps that correspond to each research question. Items in Group 1 are related to the availability of certain citation attributes, while items in Group 2 concern the quality and content of citations and reference sections if available. This means that items in Group 1 indicate positive practices employed by libraries, while those in Group 2 suggest negative patterns that have been adopted.

To respond to the research questions, two individuals analyzed the source code available in the GitHub pages for all the libraries. The coders worked independently, and the results included in this study are based on a final examination of the libraries completed in December of 2020. Because many of the libraries are still going through developments, it is possible that different results will be found when future analysis of the source code is conducted. The intercoder reliability following the work of the two coders was 89.9 %. This value was calculated based on comparing the six answers found for each of the 73 libraries (total number of answers is 438). In instances of disagreement between the two coders, the libraries were revisited by both coders in order to determine the correct answers for the disputed values. In this study, disagreements were resolved by confirming the availability of certain elements that one of the coders might have missed. For example, if one coder indicated that there were no in-code citations for one library, and the other mentioned that one was found,

Table 1
Summary of the dataset collected from the three sources.

Subset	Number of Libraries	Research Domain	List of Libraries
Web of Science	30	Several research domains such as astronomy & astrophysics, biochemistry & molecular biology, chemistry, and computer science.	scikit-learn (Pedregosa et al., 2011), matplotlib (Hunter, 2007), htseq (Anders, Pyl, & Huber, 2015), cclib (O'Boyle et al., 2007), astropy (Robitaille et al., 2013), pynast (Caporaso et al., 2010), psychopy (Peirce, 2007), eman2 (Tang et al., 2007), ipython (Pérez & Granger, 2007), DendroPy (Sukumaran & Holder, 2010), Easyfig (Sullivan, Petty, & Beatson, 2011), scikit-image (Van Der Walt et al., 2014), pymatgen (Ong et al., 2013), mdanalysis (Michaud-Agrawal, Denning, Woolf, & Beckstein, 2011), OpenSesame (Mathôt, Schreij, & Theeuwes, 2012), ProDy (Bakan, Meireles, & Bahar, 2011), Dioptas (Prescher & Prakapenka, 2015), BerkeleyGW (Deslippe et al., 2012), pyrad (Eaton, 2014), qutip (Johansson, Nation, & Nori, 2012), pymc (Patil, Huard, & Fonnesbeck, 2010), galpy (Bovy, 2015), pyradiomics (Van Griethuysen et al., 2017), mayavi (Ramachandran & Varoquaux, 2011), tomopy (Gürsoy et al., 2014), MNE-python (Gramfort et al., 2013), ETE (Huerta-Cepas et al., 2010), MMTK (Hinsen, 2000), cobrapy (Ebrahim, Lerman, Palsson, & Hyduke, 2013), PyMVPA (Hanke et al., 2009).
GitHub Most-starred Libraries	30	Several areas within the computer science domain such as computer vision and natural language processing	scikit-learn (Pedregosa et al., 2011), pandas (Mckinney, 2010), SPLEETER (Hennequin, Khlif, Voituret, & Moussallam, 2020), gensim (Rehurek & Sojka, 2010), prothet (Taylor & Letham, 2018), nltk (Bird et al., 2009), AllenNLP (Gardner et al., 2018), FAIRSEQ (Ott et al., 2019), Pattern (De Smedt & Daelemans, 2012), TPOT (Olson, Bartley, Urbanowicz, & Moore, 2016), Auto-Keras (Jin, Song, & Hu, 2019), sympy (Meurer et al., 2017), Dask (Rocklin, 2015), Pyro (Bingham et al., 2019), TensorLayer (Dong et al., 2017), Chainer (Tokui et al., 2019), Imbalanced-learn (Lemaitre et al., 2017), CuPy (Okuta, Unno, Nishino, Hido, & Loomis, 2017), Stanza (Qi et al., 2020), face-alignment (Bulat & Tzimiropoulos, 2017), scikit-image (Van Der Walt et al., 2014), librosa (McFee et al., 2015), Snips NLU (Coucke et al., 2018), PyOD (Zhao et al., 2019), Adanet (Weill et al., 2019), LightFM (Kula, 2015), Sacred (Greff, Klein, Chovanec, Hutter, & Schmidhuber, 2017), Tensorwatch (Shah, Fernandez, & Drucker, 2019), yellowbrick (Bengfort & Bilbro, 2019), 3DDFA (Zhu et al., 2017).
ACL2020	16	Natural language processing, an area within the computer science domain	ADVISER (Li, Ortega et al., 2020), CLIReval (Sun, Sia, & Duh, 2020), ConvLab-2 (Zhu et al., 2020), DIALOGPT (Zhang et al., 2020), ESPnet-ST (Inaguma et al., 2020), exBERT (Hoover, Strobelt, & Gehrmann, 2020), GAIA (Li, Zareian et al., 2020), jiant (Pruksachatkun et al., 2020), MT-DNN (Liu et al., 2020), MixingBoard (Gao, Galley, & Dolan, 2020), Penman(Goodman, 2020), pyBART (Tiktinsky, Goldberg, & Tsarfaty, 2020), OpusFilter (Aulamo, Virpioja, & Tiedemann, 2020), Stanza (Qi et al., 2020), TextBrewer (Yang et al., 2020), Torch-Struct (Rush, 2020)

the disagreement was resolved by finding the actual instance where the in-code citation took place. For reproducibility purposes, the source code for all the libraries used in this study were also downloaded in case there is a need to revisit the code in the future.

3.2.1. RQs related to availability

For the first group of steps, the coders analyzed the source code for all the packages. For each library, they first determined whether at least one citation was found in the code. This answers the first research question, which attempts to quantify the number of libraries with at least one citation used in the source code. To find a citation, the coders visited the GitHub pages for each project that included the source code and searched the source code using the following search terms: arxiv, cite, citation, citations, refer, reference, references, paper

Since these projects include several file types, only Python files were considered. In Python, there are several methods for adding natural language comments to the source code. These comments are often added in order to describe the code, document issues that need to be addressed, or provide instructions for how to properly use the code. Comments are initiated

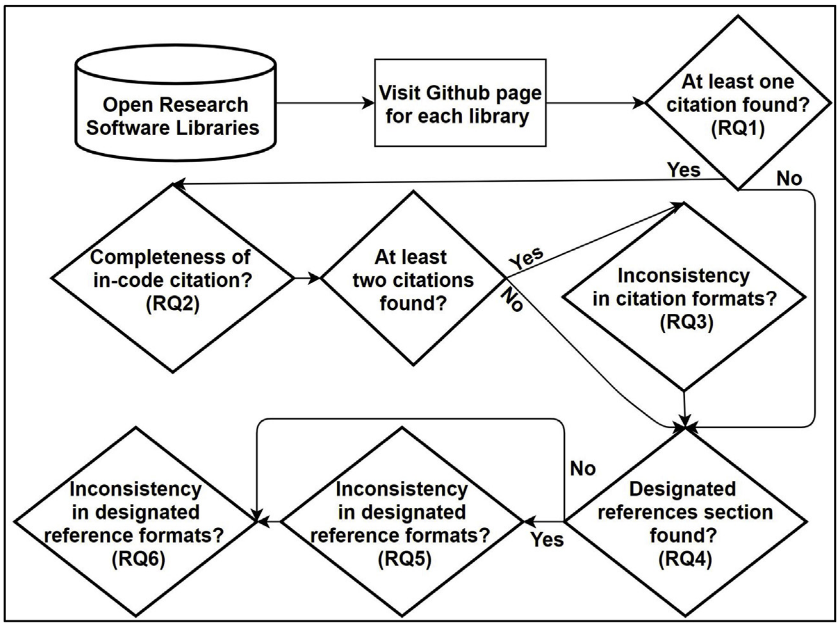

Open Research Software Libraries
Visit Github page for each library
At least one citation found? (RQ1)
Yes
No
Completeness of in-code citation? (RQ2)
At least two citations found?
Yes
No
Inconsistency in citation formats? (RQ3)
Designated references section found? (RQ4)
No
Yes
Inconsistency in designated reference formats? (RQ5)
Inconsistency in designated reference formats? (RQ6)

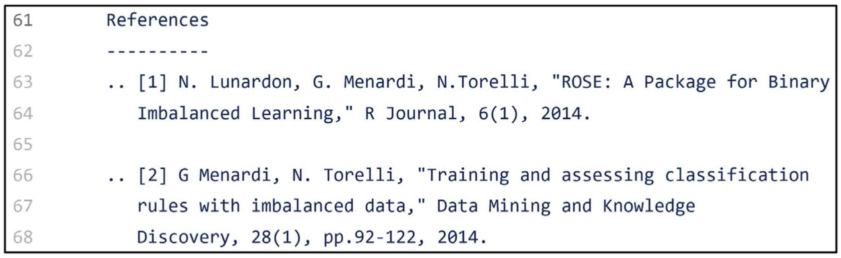
Fig. 4. Summary of the process used to answer the research questions. The questions were answered with "no" when they were skipped. Arrows with no "yes" or "no" in the figure were followed regardless of the answer.

Table 2
Summary of the steps followed to answer the research questions.

Group	Step	RQ	Name	Description
1	1.1	1	In-code citation	Denotes the **availability** of at least one citation in the source code that cites other scientific work
	1.2	4	Designated reference section	Indicates the **availability** of at least one designated reference section present either in the beginning of a file or above a programming object such as a class or a function.
	1.3	6	Bibliography/automatic way to link all citations to full references	Denotes the **availability** of a bibliography listing all the references used in the source code and a mechanism to retrieve their locations, which enable an automatic retrieval of references
2	2.1	2	Completeness of in-code citation	In the cases where a citation is used but no section is available to read the entire reference, indicates whether at least one reference used **lacks** one of the following attributes: complete names of authors, title of the paper, year, or name of publication.
	2.2	3	Citation formats	Denotes the **consistency** of citations based on a determination of whether at least two citations used in the library are presented in different formats.
	2.3	5	Designated reference section formats	Denotes the **consistency** of reference sections based on a determination of whether two sections are presented in different formats and thus cannot be automatically retrieved.

in Python using several methods. One way is to use the character "#" to indicate the start of a single line of comments. For multiline comments, also known as comment blocks, three quotation marks are used to signal the beginning and end of the block. Compilers and interpreters ignore the lines devoted to natural language comments. In this study, these comments are important because in-code citations are likely to be found within them.

In the initial research stage, libraries with at least one citation were grouped together. The goal of this first step was to find all the libraries with at least one citation in the source code that cited other scientific work. Libraries with only self-citations were not included. The next step was to identify which libraries in this group had a designated reference section. In academic papers, the in-text citations tend to be brief and the full details of the references can be viewed in the reference section. However, in research software libraries, cited scholarly works were observed to be mentioned only in the code, with no additional sections that contained the full details of the reference. Therefore, the objective for this second step was to find all the reference sections added to each library. These sections can be available at the top of Python files or at the beginning of programming objects such as functions or classes. Fig. 5 shows an example of these sections. The same search terms used for the first step were also employed here. The objective of studying the availability of these sections was to detect whether there is a standardized method in which they have been used. If that is the case, then in-code citations can be automatically retrieved by writing computer programs that are to search for these sections according to a programmed text matching


```
References
----------
.. [1] N. Lunardon, G. Menardi, N.Torelli, "ROSE: A Package for Binary
   Imbalanced Learning," R Journal, 6(1), 2014.

.. [2] G Menardi, N. Torelli, "Training and assessing classification
   rules with imbalanced data," Data Mining and Knowledge
   Discovery, 28(1), pp.92-122, 2014.
```

Fig. 5. Example of a designated reference section from the library Imbalanced-learn (Lemaitre et al., 2017).

Table 3
Descriptive statistics for the three sets based on the number of Python files in each package.

Subset	Mean	Median	Min	Max
Web of Science	349.6	206	3	1395
GitHub Most-starred	315.8	143.5	16	1374
ACL2020	160	93.5	12	609
All	275.1	147.6	3	1395

function. The final step in this sequence was to find a full bibliography or a machine-readable method of connecting in-code citations to references. The objective here was to find bibliographies that included all the references used in the library and a method to connect citations in the code to entries in the bibliography, similar to those used in scientific papers. Though the bibliography file is not usually embedded in the code, it should be available in a format such as ".bib," and list all references used in the library.

3.2.2. RQs related to quality and contents

In order to come up with the items in Group 2, the coders followed a sequence of actions that began with the examination of the contents of in-code citations and reference sections. For libraries with in-code citations, a search was conducted to determine whether a full reference of the citation was available anywhere else in the library. If no reference was found, the coders assessed the completeness of the citations in order to determine if any of the information typically required in references was missing. Information typically required in references includes: Full author names, the paper title, the name of the publication, and the publication year. For libraries with at least two in-code citations, the citations were evaluated to determine whether two different formats were used in the same library. For example, one might list the full names of the authors while another only used their first initials. Finally, I investigated the consistency of designated reference sections when multiple sections were available in a library. In other words, I wanted to identify whether the library used multiple formats for reference sections, since it is possible that two different signatures (e.g., "References:" and "###References###") could be used to signal reference sections in the same library. The use of multiple formats is problematic because it prevents automated programs from searching the source code to locate reference sections that match a specific format and thus makes it harder for both researchers and programs to find citations used in the library.

4. Results

4.1. Descriptive statistics

This section describes the results of this investigation of citation practices in open research software libraries. First, I will describe statistics for the dataset and its three subsets using the number of files and the publication dates for each paper. Table 3 includes statistics that describe the number of files available for the libraries in each subset. The "All" row shows the summary for the entire dataset. The average number of files for libraries in each subset suggests that the libraries vary in size. However, it is important to note that the number of files should not be used as the sole indicator of library size since the number of programming codes in the files varies. Put differently, a library with fewer Python files may actually be larger than others when total lines of code are calculated and used as indicators for size. Nevertheless, the average number of files in a library does give an idea of the variation in size between the libraries studied. Unsurprisingly, the numbers were larger for packages in the WoS and GitHub Most-starred sets than for packages in the ACL2020 set. This is perhaps because libraries in the ACL2020 set were recently developed and published. Moreover, the WoS and GitHub Most-starred sets have several extensive open packages with a large number of developers whose code has been written over a long period of time. The results in the table were initially collected in 2020. To ensure that they are as up to date as possible, the data were recollected in January of 2021. It is possible that many of these libraries have been updated since then and thus the numbers are different. Moreover, the numbers are based on using the search function provided by GitHub, which has limitations (Github, n.d.-b)

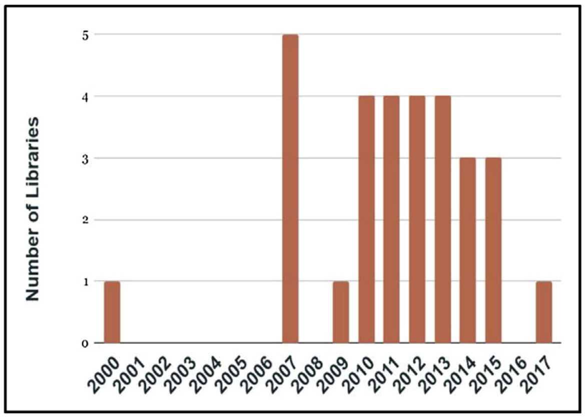


Fig. 6. Publication dates for WoS libraries.

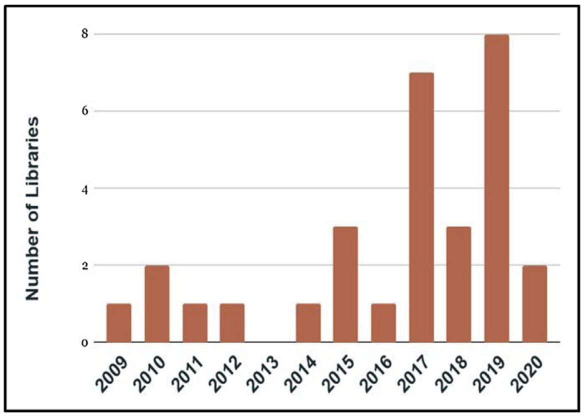


Fig. 7. Publication dates for GitHub Most-starred libraries.

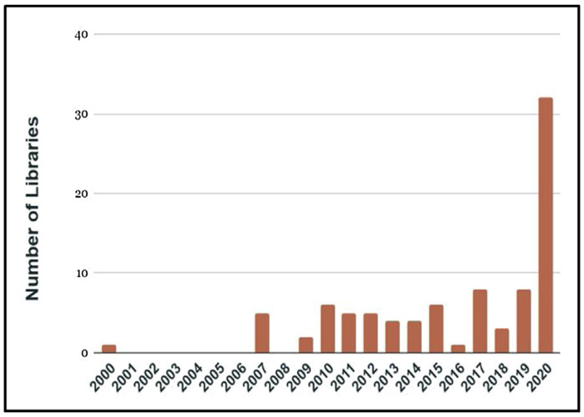


Fig. 8. Publication dates for all libraries.

In order to determine publication dates, I examined the dates on the papers for when the software was introduced in publications. Figs. 6, 7 and 8 show a summary of the publication dates for the libraries in the WoS subset, GitHub Most-starred subset, and for all the libraries from the three sources, respectively. As displayed in Fig. 6, 32 of the libraries in the dataset were published in 2020. This is because all the libraries in the ACL2020 subset were published in 2020. While most of the packages in the GitHub Most-starred subset were published after 2016, none in the WoS subset were published after 2017. Moreover, one in the WoS subset was published in 2000 and the GitHub page for the paper suggests that it is still maintained and continuously updated by the author. The variety in the publication dates for libraries in the three sets provides a broader context for citation practices.

Table 4
Summary of results related to RQ1, RQ2, and RQ3.

Subset	Count	In-code Citations (RQ1)	Incomplete in-code citations (RQ2)	Two or more in-code citations	Inconsistent formats for in-code citations (RQ3)
Web of Science	30	22 (73.3 %)	17 (%77.2)	19 (63.3 %)	17 (89.4 %)
GitHub Most-starred	30	23 (76.6 %)	22 (95.6 %)	19 (63.3 %)	18 (94.7 %)
ACL2020	16	12 (75 %)	11 (91.6 %)	12 (75 %)	10 (83.3 %)
All	73	54 (73.9 %)	47 (87 %)	47 (64.3 %)	42 (89.3 %)

4.2. RQ1 (In-code Citation), RQ2 (Completeness of in-code Citation), and RQ3 (Citation Formats)

For the 73 libraries, 54 had at least one in-code citation, while only 19 did not have a single citation to other scholarly work. This disparity suggests that in-code citations are common in open research software libraries. However, 7 of the 54 libraries included only one citation and thus were not considered in order to respond to RQ3. This means that 64.3 % of libraries contained two or more citations, though the actual number of citations varied fairly widely across libraries. Additionally, some seemed to have many citations across several files while others had only a few that were all listed in the same Python file. There was also a fair amount of variation across citations in terms of the details that they provided. For example, I observed that it was common to add links to papers hosted by the preprint hosting service “arxiv.org.” In addition, some citations included links to other websites where the papers were made accessible. Table 4 lists the numbers found for the three source sets and for the entire dataset. The “Count” column refers to the total number of libraries in the sets. As stated, the total number of packages is 73 and not 76 because three packages were repeated in two sets. The “In-code Citations” column refers to the number of packages with at least one citation. This addresses the first research question, which focuses on quantifying the number of open research software libraries with at least one in-code citation.

The “Incomplete In-code Citations” column refers to the number of libraries that had at least one citation that did not include the names of the authors, title, data, or name of the publication. The results indicated that 47 of the 54 libraries with at least one in-code citation had missing information. Libraries with complete bibliographies available in an external file that allowed for the retrieval of the full details of a citation were not counted. This was the case for six of the libraries, which used aliases for citations and then included a full reference in a separate file that titled each reference with a unique alias. I defined an in-code citation as missing information when it did not list the name of the paper or publication, only provided a link to the paper, only included the name of the first author, or did not add the year of publication. The percentages in the column were calculated by dividing the number of libraries with incomplete citations by the number of libraries with at least one citation. For example, the percentage for “All” (Table 4, row 4, column 2) was 87 % after dividing 47, the number of libraries with incomplete citations, by 54, the total number of libraries with at least one citation. This high number suggests that rigorous citation practices are not implemented by open research software libraries. One limitation here, though, is that libraries that use more citations are more likely to include citations with missing information.

Finally, RQ3 led me to consider the lack of uniformity in the textual structures of in-code citations used by the same library. To measure this, a library has to have at least two in-code citations in order to compare how they are used and determine whether their structures are different. Table 4 includes a column that shows the number of packages with at least two citations. Results showed that all but five libraries used multiple formats for citations. This suggests that within the same library, variations of in-code citation practices commonly exist. Furthermore, findings suggest that 89 % of the projects with at least two citations used several formats for citations. One possible explanation for this is the collaborative nature of many of these projects. Many individuals write the code, and each uses their own understanding of acceptable in-code citation practices. The percentages in the last column were calculated based on using the total number of packages in each set with at least two citations. To summarize, the results presented in this section suggest that while in-code citations are widely used in libraries, they are often implemented using inconsistent methods and frequently lack important information that helps connect the citations to published papers.

4.3. RQ4 (Designated Reference Sections) and RQ5 (Designated Reference Section Formats)

For 36 of the 73 libraries, at least one designated reference section was included, which listed every reference used in the programming block. In some cases, the sections appeared in the beginning of the Python files with a listing of all the previous work on which the code in the file relied. In others, the sections appeared in the beginning of a programming object such as a class or a function. I observed that libraries used several ways to signal the start of a reference section. Many began with “References:” or “**References**” often followed by a line with the following characters: “############” or “----------.” In some cases, an empty line or a direct listing followed the “references” declaration. To determine if there were inconsistencies in the structures of designated reference sections, all the libraries with at least two such sections were identified. The results, which are listed in Table 5, show that just six of the libraries containing *at least* one reference section contained *only* one reference section. This indicates that when libraries use reference sections, they are likely to employ them more than once. The percentages for the column “Two or more designated reference sections” are based on the number of libraries while the percentages in the last column are based on the number of libraries with two or more sections.

Table 5
Summary for results related to RQ4 and RQ5.

Subset	Number of Libraries	Designated reference section (RQ4)	Two or more designated reference sections	Inconsistent formats for designated reference section (RQ5)
Web of Science	30	17 (56.6 %)	15 (50 %)	13 (86.6 %)
GitHub Most-starred	30	16 (53.3 %)	12 (40 %)	12 (100 %)
ACL2020	16	5 (31.2 %)	5 (31.2 %)	2 (40 %)
All	73	36 (49.3 %)	30 (41 %)	25 (83.3 %)

While it is promising that many libraries were using designated sections for references, results indicated that libraries used several differing structures for these sections. For 25 of the 30 with multiple sections, at least two sections were in an inconsistent format. On one hand, using designated reference sections suggests protocols that could be utilized to develop a system that tracks and indexes references used in libraries. On the other hand, the use of several differing structures for reference sections means that researchers are more likely to overlook references when conducting projects. Thus, the benefits of these sections are limited when considered for automated applications or web crawlers that scrape references, for example. Moreover, over the course of this study, I observed that the word “references” was often used in different contexts within the same library. This makes it difficult to assume that an instance of the word “references” within a library would indicate the beginning of a specific reference section.

4.4. RQ6: availability of a bibliography/systematic method

The final research question concerns the existence of a systematic method implemented by libraries to 1) add in-code citations using a standardized format, and 2) list all the references used in the library with a programmable way to retrieve their exact locations in the package. For the 73 libraries, only six implemented such a method. Four of these libraries were in the WoS set, one was in the GitHub set, and one was in the ACL2020 set. This low number suggests that academic citations added to libraries are not easily retrievable.

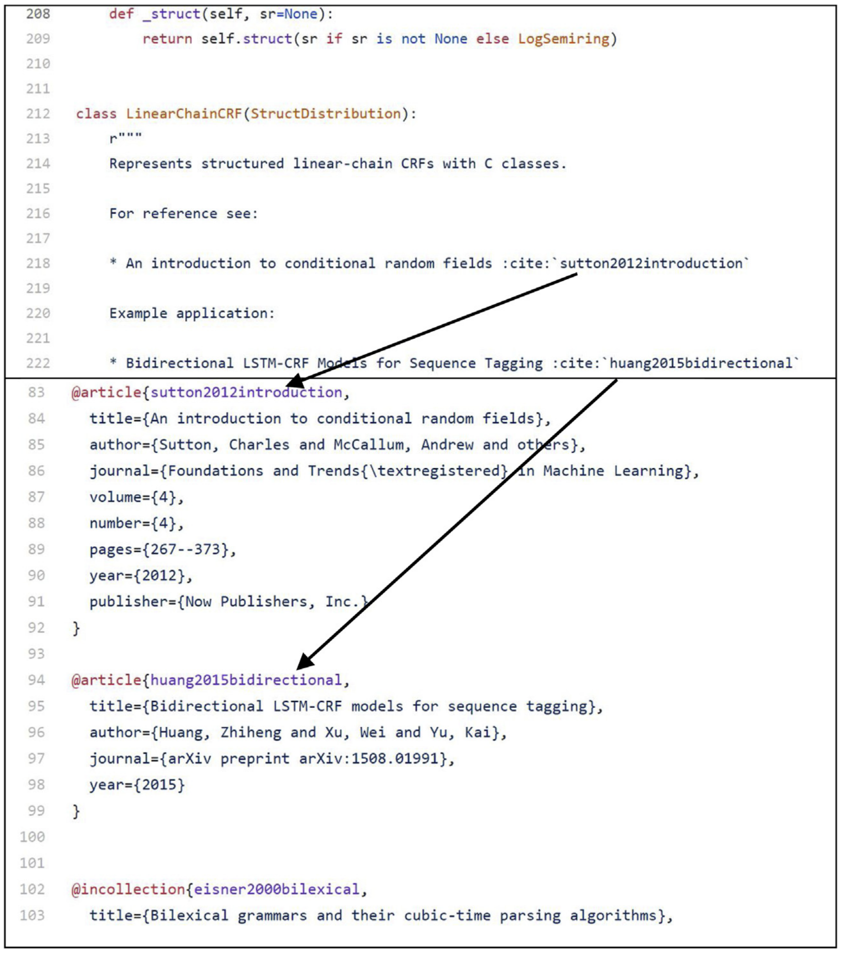
Five of these libraries implemented an identical method. The method relies on using a file with the “.bib” file extension, which is a format commonly used to organize, maintain, and share a group of references. The “.bib” files in the libraries that were studied contained a list of references and each had an alias or an identifying name. When a citation was added to the programming code, a tag such as “:cite:” or “:ref:” was used to indicate the start of a citation. Fig. 9 demonstrates how the three libraries use “.bib” files to automatically link citations to references in a searchable way. The screenshots are from the library PyTorch-Struct (Rush, 2020). On top, a snippet from the code with two internal citations is displayed. In lines 218 and 222, two citations are shown. At the bottom of the figure, a screenshot from the “.bib” file in the same library is displayed. Lines 83 and 94 contain the references for the citations used in the Python file. This allows automated programs interested in discovering all references used in a library to read the “.bib” file and then search the Python code looking for any instance of the tag “:cite:” or the exact name of the references.

5. Discussion

The findings of this study document the lack of a widespread use of uniform in-code citation standards that allow for automatic retrieval and indexing of references used in the programming code of open research software libraries. This may affect the performance of tools that attempt to link papers to code snippets where they are employed. For the three research questions concerned with the availability of in-code citations, reference sections, and a bibliography, only four libraries had all three. Additionally, 19 libraries did not have any of these three availability attributes. The summary of these results is in Fig. 10. The chart shows that 34 libraries had two of the three attributes present. Fig. 11 shows the overlap between the attributes. For example, there were 32 libraries with an in-code citation and a designated reference section but no bibliography.

Despite these constraints, this work identifies several ideal practices that can be beneficial for the scientific community if they are adopted and commonly used in open research software libraries. One library, ETE (Huerta-Cepas, Dopazo, & Gabaldón, 2010), came up with a creative solution for in-code citations. The solution is based on the creation of a Python dictionary object[1] that they used to add the list of references. Each item in the dictionary has a unique key and a text entry that contains the full reference. Then, when an in-code citation is added, a programmable object is used to link the citation using the predefined keys. The ideal and best practices were found in the following libraries: MNE-Python (Gramfort et al., 2013), PyOD (Zhao, Nasrullah, & Li, 2019), PyMVPA (Hanke et al., 2009), PyTorch-Struct (Rush, 2020), and tomopy (Gürsoy, De Carlo, Xiao, & Jacobsen, 2014). As described in Section 4.4, these libraries include a “.bib” file that has all the references used

[1] Python dictionary objects can store several objects, each with its own identifying key. For a full explanation, please see https://docs.python.org/3/tutorial/datastructures.html#dictionaries.



```
    def _struct(self, sr=None):
        return self.struct(sr if sr is not None else LogSemiring)


class LinearChainCRF(StructDistribution):
    r"""
    Represents structured linear-chain CRFs with C classes.

    For reference see:

    * An introduction to conditional random fields :cite:`sutton2012introduction`

    Example application:

    * Bidirectional LSTM-CRF Models for Sequence Tagging :cite:`huang2015bidirectional`
```

```
@article{sutton2012introduction,
  title={An introduction to conditional random fields},
  author={Sutton, Charles and McCallum, Andrew and others},
  journal={Foundations and Trends{\textregistered} in Machine Learning},
  volume={4},
  number={4},
  pages={267--373},
  year={2012},
  publisher={Now Publishers, Inc.}
}

@article{huang2015bidirectional,
  title={Bidirectional LSTM-CRF models for sequence tagging},
  author={Huang, Zhiheng and Xu, Wei and Yu, Kai},
  journal={arXiv preprint arXiv:1508.01991},
  year={2015}
}


@incollection{eisner2000bilexical,
  title={Bilexical grammars and their cubic-time parsing algorithms},
```

Fig. 9. An example of a retrievable way to add in-code citations as implemented by PyTorch-Struct (Rush, 2020). On top, the Python code where the citations were added; on bottom, part of the contents of a ".bib" file with the references.

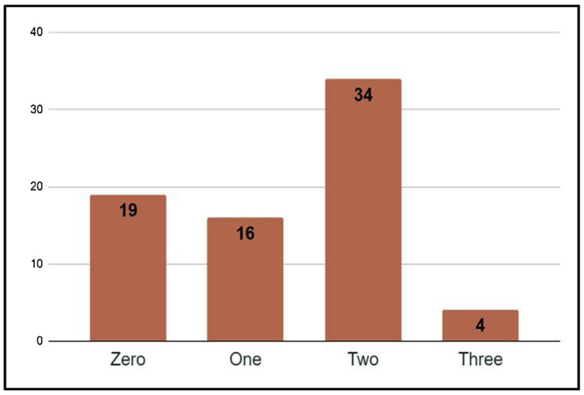


Fig. 10. Summary of the number of libraries with and without availability attributes.

in the source code. Each entry in the file represents a reference and includes all the details commonly added to references. However, it was observed that some of the references in the file did not seem to be used as citations in the source code. Additionally, many other libraries included a ".bib" file but used it only to list the references cited in the paper that was published about the library. Finally, one library used a ".bib" file and also added citations explicitly without including them in the file. As these examples demonstrate, simply retrieving any ".bib" file in a library and then using it to extract all the

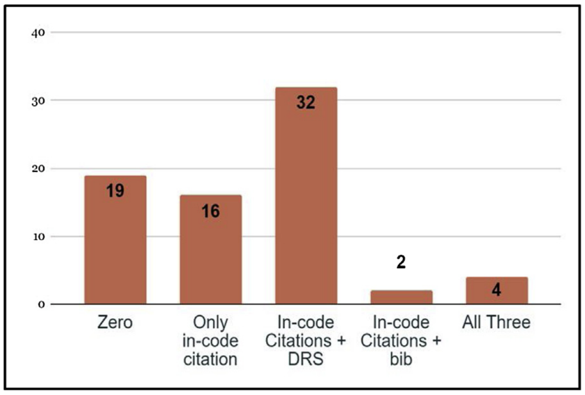


Fig. 11. Summary of the number of libraries and attributes, illustrating the overlap between attributes.

citations in the source code may not be a method that is guaranteed to work in all cases. Nevertheless, based on the findings exhibited here, it is evident that immediate action could be taken by open research software libraries to include a ".bib" file in their source code. Doing so would make their references retrievable by scrapers developed to collect and then index references.

One imperative question to address is: If these are the ideal best practices for in-code citations, how can other libraries be influenced to implement the same techniques in their projects? I posit that it would be beneficial for outlets that commonly publish papers that explain new open research software libraries to encourage this technique. For example, the Journal of Machine Learning Research often publishes papers that describe new libraries. Similarly, the ACL conference annually accepts "demo" papers that include many open research software libraries. If such outlets describe the protocols of acceptable in-code citation practices, more libraries may be influenced to use them. Furthermore, it is possible that other scientists simply do not know or think of these methods, which rely on including a ".bib" file and using it to link in-code citations to references. Thus, it follows that published explanations of these best practices, along with increased implementation, would equip more scientists and developers with the awareness necessary to implement best practices themselves.

It is important to note that the numbers presented in this paper may be affected by the potential limitations of this study. First, large open source projects may include Python files that were developed by others. Thus, the variations in citation structure in one library could be attributed to the addition of files incorporated from other libraries. A second potentially limiting factor is that the libraries selected for two of the sets were chosen based on their popularity. This means that one threat to the validity of my findings is the selection process. I did not, for instance, create a large collection of libraries and randomly draw a sample to use in the study. Third, the research questions were answered by manually examining each library. This process was completed by only two individuals. Thus, even though the process was repeated to ensure the accuracy of the information, it is possible that citations or references were missed. Fourth, this study relies on using the search feature in GitHub. This feature may contain limitations that prevented some citations from being discovered. For example, GitHub indicates that only files with sizes that are less than 384 KB can be searched (Github, n.d.-b). Finally, this study does not examine the quantity of citations used in each library. As discovered in this paper, most libraries do not currently provide a mechanism for the automatic retrieval of references. Therefore, finding all the references in a project may not be possible due to the large sizes of these projects.

6. Conclusion

In this paper, an exploratory study on in-code citations in open research software libraries was conducted. Findings suggest that while most libraries in the study use at least one in-code citation, most use multiple formats for citations or use citations that are missing important information. The primary contribution of this work is to document the lack of a systematic method for in-code citations that allows for automatic retrieval and linking of references used in open research software libraries. This paper also provides information about best practices currently followed by several libraries. These practices could be the solution for many of the current problems concerning in-code citations in research libraries. It is important to note the vital role these libraries have played in the advancements of several research domains. Research in the natural language processing field, for example, relies on the utilization of these open libraries. Ultimately, good citation practices are essential in all academic fields because they allow for the continuation of many different types of inquiry. Clearing up the confusion and eliminating the accidental erasure of references that are common within these libraries would go a long way towards improving the types of research that are available to and possible for scholars in multiple fields.

Author contributions

Abdulkareem Alsudais: Conceived and designed the analysis, Collected the data, Contributed data or analysis tools, Performed the analysis, Wrote the paper.

Acknowledgement

This project was supported by the Deanship of Scientific Research at Prince Sattam bin Abdulaziz University, Al-Kharj, Saudi Arabia.

References

ACL. (2020). *Call for system demonstrations* Retrieved April 20, 2020, from. https://acl2020.org/calls/demos/

Alliez, P., Cosmo, R. D. i, Guedj, B., Girault, A., Hacid, M.-S., Legrand, A., & Rougier, N. (2020). Attributing and referencing (Research) software: Best practices and outlook from Inria. *Computing in Science & Engineering*, *22*, 39–52.

AlNoamany, Y., & Borghi, J. A. (2018). Towards computational reproducibility: Researcher perspectives on the use and sharing of software. *PeerJ Computer Science*, *4*, e163. https://doi.org/10.7717/peerj-cs.163

Anders, S., Pyl, P. T., & Huber, W. (2015). HTSeq-A Python framework to work with high-throughput sequencing data. *Bioinformatics*, *31*(2), 166–169. https://doi.org/10.1093/bioinformatics/btu638

Anzt, H., Bach, F., Druskat, S., Loewe, A., Renard, B. Y., Struck, A., . . . & Weeber, R. (2020). *An environment for sustainable research software in Germany and beyond: Current state, open challenges, and call for action* 001 (May). pp. 1–17.

Aulamo, M., Virpioja, S., & Tiedemann, J. (2020). OpusFilter: A configurable parallel corpus filtering toolbox. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 150–156). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.20

Bakan, A., Meireles, L. M., & Bahar, I. (2011). ProDy: Protein dynamics inferred from theory and experiments. *Bioinformatics*, *27*(11), 1575–1577. https://doi.org/10.1093/bioinformatics/btr168

Bengfort, B., & Bilbro, R. (2019). Yellowbrick: Visualizing the scikit-learn model selection process. *The Journal of Open Source Software*, *4*(35) https://doi.org/10.21105/joss.01075

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., . . . & Goodman, N. D. (2019). Pyro: Deep universal probabilistic programming. *Journal of Machine Learning Research*, *20*(28), 1–28, 6. Retrieved from. http://jmlr.org/papers/v20/18-403.html

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python* (1st ed.). O'Reilly Media, Inc.

Bovy, J. (2015). Galpy: A python library for galactic dynamics. *The Astrophysical Journal Supplement Series*, *216*(2) https://doi.org/10.1088/0067-0049/216/2/29

Bulat, A., & Tzimiropoulos, G. (2017). How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks). *International conference on computer vision*.

Caporaso, J. G., Bittinger, K., Bushman, F. D., Desantis, T. Z., Andersen, G. L., & Knight, R. (2010). PyNAST: A flexible tool for aligning sequences to a template alignment. *Bioinformatics*, *26*(2), 266–267. https://doi.org/10.1093/bioinformatics/btp636

Catolino, G., Palomba, F., Zaidman, A., & Ferrucci, F. (2019). Not all bugs are the same: Understanding, characterizing, and classifying bug types. *The Journal of Systems and Software*, *152*, 165–181. https://doi.org/10.1016/j.jss.2019.03.002

Celi, L. A., Citi, L., Ghassemi, M., & Pollard, T. J. (2019). The PLOS ONE collection on machine learning in health and biomedicine: Towards open code and open data. *PloS One*, *14*(1), 1–7. https://doi.org/10.1371/journal.pone.0210232

Chassanoff, A., & Altman, M. (2020). Curation as "Interoperability With the Future": Preserving Scholarly Research Software in Academic Libraries. *Journal of the Association for Information Science and Technology*, *71*(3), 325–337. https://doi.org/10.1002/asi.24244

Chassanoff, A., Alnoamany, Y., Thornton, K., Borghi, J., Chassanoff, A., & Borghi, J. (2018). Software curation in research libraries: Practice and promise. *Journal of Librarianship and Scholarly Communication*, *6*.

Chen, Z., Jiang, R., Zhang, Z., Pei, Y., Pan, M., & Zhang, T. (2020). The Journal of Systems and Software Enhancing example-based code search with functional semantics R. *The Journal of Systems and Software*, *165*, Article 110568 https://doi.org/10.1016/j.jss.2020.110568

Cheng, X., Zhang, Z., Yang, Y., & Yan, Z. (2020). Open collaboration between universities and enterprises : A case study on GitHub. *Internet Research*, *30*(4), 1251–1279. https://doi.org/10.1108/INTR-01-2019-0013

Collaboration, T. A., Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., . . . & Streicher, O. (2013). . pp. 1–9. *Astrophysics astropy: A community python package for astronomy* (33).

Cosentino, V., Cánovas Izquierdo, J. L., & Cabot, J. (2017). A systematic mapping study of software development with GitHub. *IEEE Access*, *5*, 7173–7192. https://doi.org/10.1109/ACCESS.2017.2682323

Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., . . . & others. (2018). *Snips Voice Platform: An embedded Spoken Language understanding system for private-by-design voice interfaces* arXiv Preprint arXiv:1805.10190. pp. 12–16.

De Smedt, T., & Daelemans, W. (2012). Pattern for Python. *Journal of Machine Learning Research*, *13*(66), 2063–2067. Retrieved from. http://jmlr.org/papers/v13/desmedt12a.html

Deslippe, J., Samsonidze, G., Strubbe, D. A., Jain, M., Cohen, M. L., & Louie, S. G. (2012). BerkeleyGW: A massively parallel computer package for the calculation of the quasiparticle and optical properties of materials and nanostructures. *Computer Physics Communications*, *183*(6), 1269–1289. https://doi.org/10.1016/j.cpc.2011.12.006

Dong, H., Supratak, A., Mai, L., Liu, F., Oehmichen, A., Yu, S., & Guo, Y. (2017). TensorLayer: A versatile library for efficient deep learning development. *ACM Multimedia*,. Retrieved from. http://tensorlayer.org

Druskat, S. (2020). Software and dependencies in research citation graphs. *Computing in Science & Engineering*, *22*(2), 8–21. https://doi.org/10.1109/MCSE.2019.2952840

Druskat, S., & Katz, D. S. (2018). Mapping the research software sustainability space. In *2018 IEEE 14th international conference on e-Science (e-Science)* (pp. 25–30). https://doi.org/10.1109/eScience.2018.00014

Eaton, D. A. R. (2014). PyRAD: Assembly of de novo RADseq loci for phylogenetic analyses. *Bioinformatics*, *30*(13), 1844–1849. https://doi.org/10.1093/bioinformatics/btu121

Ebrahim, A., Lerman, J. A., Palsson, B. O., & Hyduke, D. R. (2013). COBRApy: COnstraints-based reconstruction and analysis for Python. *BMC Systems Biology*, *7* https://doi.org/10.1186/1752-0509-7-74

Gao, X., Galley, M., & Dolan, B. (2020). MixingBoard: A knowledgeable stylized integrated text generation platform. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 224–231). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.26

Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N. F., . . . & Zettlemoyer, L. (2018). AllenNLP: A deep semantic natural language processing platform. In *Proceedings of workshop for NLP open source software NLP-OSS* (pp. 1–6). https://doi.org/10.18653/v1/W18-2501

Github. (n.d.-a). Python. Retrieved April 15, 2020, from https://github.com/topics/python?o=desc&s=stars.

Github. (n.d.-b). Searching code. Retrieved December 15, 2020, from: https://docs.github.com/en/free-pro-team@latest/github/searching-for-information-on-github/searching-code.
Goodman, M. W. (2020). Penman: An open-source library and tool for AMR graphs. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 312–319). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.35
Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., . . . & Hämäläinen, M. (2013). MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, (December (7)) https://doi.org/10.3389/fnins.2013.00267
Greff, K., Klein, A., Chovanec, M., Hutter, F., & Schmidhuber, J. (2017). The sacred infrastructure for computational research. In K. Huff, D. Lippa, D. Niederhut, & M. Pacer (Eds.), *Proceedings of the 16th python in science conference* (pp. 49–56). https://doi.org/10.25080/shinma-7f4c6e7-008
Gürsoy, D., De Carlo, F., Xiao, X., & Jacobsen, C. (2014). TomoPy: A framework for the analysis of synchrotron tomographic data. *Journal of Synchrotron Radiation*, *21*(5), 1188–1193. https://doi.org/10.1107/S1600577514013939
Hanke, M., Halchenko, Y. O., Sederberg, P. B., Hanson, S. J., Haxby, J. V., & Pollmann, S. (2009). PyMVPA: A python toolbox for multivariate pattern analysis of fMRI data. *Neuroinformatics*, *7*(1), 37–53. https://doi.org/10.1007/s12021-008-9041-y
Hasselbring, W., Carr, L., Hettrick, S., Packer, H., & Tiropanis, T. (2020). From FAIR research data toward FAIR and open research software. *Information Technology*, *62* https://doi.org/10.1515/itit-2019-0040
Hennequin, R., Khlif, A., Voituret, F., & Moussallam, M. (2020). Spleeter: A fast and efficient music source separation tool with pre-trained models. *The Journal of Open Source Software*, *5*(50), 2154.
Heumüller, R., Nielebock, S., Krüger, J., & Ortmeier, F. (2020). Publish or perish, but do not forget your software artifacts. In *Emprical software engineering*.
Hinsen, K. (2000). The molecular modeling toolkit: A new approach to molecular simulations. *Journal of Computational Chemistry*, *21*(2), 79–85. https://doi.org/10.1002/(SICI)1096-987X(20000130)21:2<79::AID-JCC1>3.0.CO;2-B
Hoover, B., Strobelt, H., & Gehrmann, S. (2020). exBERT: A visual analysis tool to explore learned representations in transformer models. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 187–196). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.22
Huerta-Cepas, J., Dopazo, J., & Gabaldón, T. (2010). ETE: A python environment for tree exploration. *BMC Bioinformatics*, *11* https://doi.org/10.1186/1471-2105-11-24
Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 99–104. https://doi.org/10.1109/MCSE.2007.55
Inaguma, H., Kiyono, S., Duh, K., Karita, S., Yalta, N., Hayashi, T., & Watanabe, S. (2020). ESPnet-ST: All-in-one speech translation toolkit. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 302–311). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.34
Ince, D. C., Hatton, L., & Graham-cumming, J. (2012). The case for open computer programs. *Nature*, *482*, 485–488. https://doi.org/10.1038/nature10836
Jiménez, R. C., Kuzak, M., Alhamdoosh, M., Barker, M., Batut, B., Borg, M., . . . Cosmo, . . . & Di, R. (2017). Four simple recommendations to encourage best practices in research software. *F1000Research*, 1–12. https://doi.org/10.12688/f1000research.11407.1
Jin, H., Song, Q., & Hu, X. (2019). Auto-keras: An efficient neural architecture search system. *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 1946–1956.
Johansson, J. R., Nation, P. D., & Nori, F. (2012). QuTiP: An open-source Python framework for the dynamics of open quantum systems. *Computer Physics Communications*, *183*(8), 1760–1772. https://doi.org/10.1016/j.cpc.2012.02.021
Kardas, M., Czapla, P., Stenetorp, P., Ruder, S., Riedel, S., Taylor, R., & Stojnic, R. (2020). *AxCell: Automatic extraction of results from machine learning papers* arXiv Preprint. Retrieved from. https://arxiv.org/abs/2004.14356
Katz, A. D. S., Bouquin, D., Hong, N. P. C., Hausman, J., Chivvis, D., Clark, T., . . . & Wu, M. (2019). *Software citation implementation challenges*. pp. 1–26.
Kula, M. (2015). Metadata embeddings for user and item cold-start recommendations. In T. Bogers, & M. Koolen (Eds.), *Proceedings of the 2nd workshop on new trends on Content-based recommender systems Co-Located With 9th {ACM} conference on recommender systems (RecSys 2015)* (Vol. 1448) (pp. 14–21). CEUR-WS.org. Retrieved from: http://ceur-ws.org/Vol-1448/paper4.pdf
Lamprecht, A., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin, E., . . . & Angel, D. (2020). pp. 37–59. *Towards FAIR principles for research software* (3) https://doi.org/10.3233/DS-190026
Lemaitre, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, *18*(17), 1–5. Retrieved from. http://jmlr.org/papers/v18/16-365
Li, K., & Yan, E. (2018). Co-mention network of R packages: Scientific impact and clustering structure. *Journal of Informetrics*, *12*(1), 87–100. https://doi.org/10.1016/j.joi.2017.12.001
Li, K., Chen, P., & Yan, E. (2019). Challenges of measuring software impact through citations: An examination of the lme4 R package. *Journal of Informetrics*, *13*(1), 449–461. https://doi.org/10.1016/j.joi.2019.02.007
Li, K., Lin, X., & Greenberg, J. (2016). Software citation, reuse and metadata considerations: An exploratory study examining LAMMPS. *Proceedings of the association for information science and technology*, 1–10.
Li, K., Yan, E., & Feng, Y. (2017). How is R cited in research outputs? Structure, impacts, and citation standard. *Journal of Informetrics*, *11*(4), 989–1002. https://doi.org/10.1016/j.joi.2017.08.003
Li, C.-Y., Ortega, D., Väth, D., Lux, F., Vanderlyn, L., Schmidt, M., . . . & Vu, N. T. (2020). ADVISER: A toolkit for developing multi-modal, multi-domain and socially-engaged conversational agents. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 279–286). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.31
Li, M., Zareian, A., Lin, Y., Pan, X., Whitehead, S., Chen, B., . . . & Freedman, M. (2020). GAIA: A fine-grained multimedia knowledge extraction system. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 77–86). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.11
Liu, X., Wang, Y., Ji, J., Cheng, H., Zhu, X., Awa, E., . . . & Gao, J. (2020). The microsoft toolkit of multi-task deep neural networks for natural language understanding. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 118–126). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.16
Martín-martín, A., Orduna-malea, E., Thelwall, M., & López-cózar, E. D. (2018). Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories. *Journal of Informetrics*, *12*(4), 1160–1177. https://doi.org/10.1016/j.joi.2018.09.002
Mathôt, S., Schreij, D., & Theeuwes, J. (2012). OpenSesame: An open-source, graphical experiment builder for the social sciences. *Behavior Research Methods*, *44*(2), 314–324. https://doi.org/10.3758/s13428-011-0168-7
McFee, B., Raffel, C., Liang, D., Ellis, D. P. W., McVicar, M., Battenberg, E., & Nieto, O. (2015). Librosa: Audio and music signal analysis in python. *Proceedings of the 14th python in Science Conference*, *Vol. 8*, 18–25.
Mckinney, W. (2010). Data structures for statistical computing in python. *Proceedings of the 9th python in science conference*, 51–56.
Meurer, A., Smith, C. P., Paprocki, M., Čert'ik, O., Kirpichev, S. B., Rocklin, M., . . . & Scopatz, A. (2017). SymPy: Symbolic computing in Python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103
Michaud-Agrawal, N., Denning, E. J., Woolf, T. B., & Beckstein, O. (2011). MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. *Journal of Computational Chemistry*, *32*(10), 2319–2327. https://doi.org/10.1002/jcc.21787
O'Boyle, N. M., Tenderholt, A. L., & Langner, K. M. (2007). Cclib: A library for package-independent computational chemistry algorithms. *Journal of Computational Chemistry*, *29*(5), 839–845. https://doi.org/10.1002/jcc
Okuta, R., Unno, Y., Nishino, D., Hido, S., & Loomis, C. (2017). CuPy: A NumPy-compatible library for NVIDIA GPU calculations. *Proceedings of workshop on machine learning systems (LearningSys) in the thirty-first annual conference on neural information processing systems (NIPS)*,. Retrieved from. http://learningsys.org/nips17/assets/papers/paper_16.pdf

Olson, R. S., Bartley, N., Urbanowicz, R. J., & Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the genetic and evolutionary computation conference 2016* (pp. 485–492). https://doi.org/10.1145/2908812.2908918

Ong, S. P., Richards, W. D., Jain, A., Hautier, G., Kocher, M., Cholia, S., . . . & Ceder, G. (2013). Python Materials Genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, *68*, 314–319. https://doi.org/10.1016/j.commatsci.2012.10.028

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., . . . & Auli, M. (2019). Fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 conference of the North {A}merican chapter of the association for computational linguistics (Demonstrations)*. pp. 48–53. Minneapolis, Minnesota: Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-4009

Pan, X., Yan, E., Cui, M., & Hua, W. (2018). Examining the usage, citation, and diffusion patterns of bibliometric mapping software: A comparative study of three tools. *Journal of Informetrics*, *12*(2), 481–493. https://doi.org/10.1016/j.joi.2018.03.005

Pan, X., Yan, E., Cui, M., & Hua, W. (2019). How important is software to library and information science research? A content analysis of full-text publications. *Journal of Informetrics*, *13*(1), 397–406. https://doi.org/10.1016/j.joi.2019.02.002

Park, H., & Wolfram, D. (2019). Research software citation in the Data Citation Index: Current practices and implications for research software sharing and reuse. *Journal of Informetrics*, *13*(2), 574–582. https://doi.org/10.1016/j.joi.2019.03.005

Patil, A., Huard, D., & Fonnesbeck, C. J. (2010). PyMC: Bayesian stochastic modelling in Python. *Journal of Statistical Software*, *35*(4), 1–81. https://doi.org/10.18637/jss.v035.i04

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peirce, J. W. (2007). PsychoPy—Psychophysics software in Python. *Journal of Neuroscience Methods*, *162*(1), 8–13. https://doi.org/10.1016/j.jneumeth.2006.11.017

Pérez, F., & Granger, B. E. (2007). IPython: A system for interactive scientific computing. *Computing in Science & Engineering*, *9*(3), 21–29. https://doi.org/10.1109/MCSE.2007.53

Pianosi, F., Sarrazin, F., & Wagener, T. (2020). How successfully is open-source research software adopted? Results and implications of surveying the users of a sensitivity analysis toolbox. *Environmental Modelling & Software*, *124*(November 2019), 104579. https://doi.org/10.1016/j.envsoft.2019.104579

Prescher, C., & Prakapenka, V. B. (2015). DIOPTAS: A program for reduction of two-dimensional X-ray diffraction data and data exploration. *High Pressure Research*, *35*(3), 223–230. https://doi.org/10.1080/08957959.2015.1059835

Pruksachatkun, Y., Yeres, P., Liu, H., Phang, J., Htut, P. M., Wang, A., . . . & Bowman, S. R. (2020). Jiant: A software toolkit for research on general-purpose text understanding models. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 109–117). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.15

PyTorch. (n.d.). Retrieved November 7, 2020, from https://github.com/pytorch/pytorch.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020). Stanza: A {Python} natural language processing toolkit for many human languages. *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations*,

Ramachandran, P., & Varoquaux, G. (2011). Mayavi: 3D visualization of scientific data. *Computing in Science & Engineering*, *13*(2), 40–51. https://doi.org/10.1109/MCSE.2011.35

Rehurek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks*. pp. 45–50.

Robitaille, T. P., Tollerud, E. J., Greenfield, P., Droettboom, M., Bray, E., Aldcroft, T., . . . & Streicher, O. (2013). Astropy: A community Python package for astronomy. *Astronomy and Astrophysics*, *558* https://doi.org/10.1051/0004-6361/201322068

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling. In K. Huff, & J. Bergstra (Eds.), *Proceedings of the 14th python in science conference* (pp. 130–136).

Rush, A. (2020). Torch-struct: Deep structured prediction library. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 335–342). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.38

Sandt, S., Van De, Nielsen, L. H., Ioannidis, A., Muench, A., Henneken, E., Accomazzi, A., . . . & Dallmeier-Tiessen, S. (2019). *Practice meets principle: Tracking software and data citations to zenodo DOIs* Retrieved from. arXiv. https://arxiv.org/abs/1911.00295

Shah, S., Fernandez, R., & Drucker, S. (2019). A system for real-time interactive analysis of deep learning training. In *Proceedings of the {ACM} {SIGCHI} symposium on engineering interactive computing systems, {EICS} 2019* (pp. 1–6). https://doi.org/10.1145/3319499.3328231

Shao, H., Sun, D., Wu, J., Zhang, Z., Zhang, A., Yao, S., . . . & Abdelzaher, T. (2020). paper2repo: GitHub repository recommendation for academic papers. In *WWW' 20: Proceedings of the web conference 2020* (pp. 629–639).

Smith, A. M., Katz, D. S., Niemeyer, K. E., & FORCE11 Software Citation Working Group. (2016). Software citation principles. *PeerJ Computer Science*, *2*(e86), 1–31. https://doi.org/10.7717/peerj-cs.86

Sukumaran, J., & Holder, M. T. (2010). DendroPy: a Python library for phylogenetic computing. *Bioinformatics*, *26*(12), 1569–1571. https://doi.org/10.1093/bioinformatics/btq228

Sullivan, M. J., Petty, N. K., & Beatson, S. A. (2011). Easyfig: A genome comparison visualizer. *Bioinformatics*, *27*(7), 1009–1010. https://doi.org/10.1093/bioinformatics/btr039

Sun, S., Sia, S., & Duh, K. (2020). CLIReval: Evaluating machine translation as a cross-lingual information retrieval task. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 134–141). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.18

Tang, G., Peng, L., Baldwin, P. R., Mann, D. S., Jiang, W., Rees, I., & Ludtke, S. J. (2007). EMAN2: An extensible image processing suite for electron microscopy. *Journal of Structural Biology*, *157*(1), 38–46. https://doi.org/10.1016/j.jsb.2006.05.009

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, *72*(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

TensorFlow (n.d.). Retrieved November 7, 2020, from https://github.com/tensorflow/tensorflow.

Tiktinsky, A., Goldberg, Y., & Tsarfaty, R. (2020). pyBART: Evidence-based syntactic transformations for IE. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 47–55). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.7

Tokui, S., Okuta, R., Akiba, T., Niitani, Y., Ogawa, T., Saito, S., . . . & Yamazaki Vincent, H. (2019). Chainer: A deep learning framework for accelerating the research cycle. *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2002–2011.

Van Der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., . . . & Yu, T. (2014). Scikit-image: Image processing in python. *PeerJ*, *2014*(1) https://doi.org/10.7717/peerj.453

Van Griethuysen, J. J. M., Fedorov, A., Parmar, C., Hosny, A., Aucoin, N., Narayan, V., . . . & Aerts, H. J. W. L. (2017). Computational radiomics system to decode the radiographic phenotype. *Cancer Research*, *77*(21), e104–e107. https://doi.org/10.1158/0008-5472.CAN-17-0339

Vogel, T., Druskat, S., Scheidgen, M., Draxl, C., & Grunske, L. (2019). Challenges for verifying and validating scientific software in computational materials science. *SE4Science' 19: Proceedings of the 14th international workshop on software engineering for science*.

Weill, C., Gonzalvo, J., Kuznetsov, V., Yang, S., Yak, S., Mazzawi, H., . . . & Carolina, N. (2019). *AdaNet: A scalable and flexible framework for automatically learning ensembles*. pp. 1–11.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., . . . & Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*(1), 160018. https://doi.org/10.1038/sdata.2016.18

Yang, Z., Cui, Y., Chen, Z., Che, W., Liu, T., Wang, S., & Hu, G. (2020). TextBrewer: An open-source knowledge distillation toolkit for natural language processing. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System Demonstrations* (pp. 9–16). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.2

Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., . . . & Dolan, B. (2020). DIALOGPT: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 270–278). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.30

Zhao, Y., Nasrullah, Z., & Li, Z. (2019). PyOD: A Python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, *20*(96), 1–7. Retrieved from. http://jmlr.org/papers/v20/19-011.html

Zhu, Q., Zhang, Z., Fang, Y., Li, X., Takanobu, R., Li, J., . . . & Huang, M. (2020).). ConvLab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. In *Proceedings of the 58th annual meeting of the association for computational linguistics: System demonstrations* (pp. 142–149). Retrieved from. https://www.aclweb.org/anthology/2020.acl-demos.19

Zhu, X., Liu, X., Lei, Z., & Li, S. (2017). Face alignment in full pose range: A 3D total solution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1. https://doi.org/10.1109/TPAMI.2017.2778152

Zou, W., Xuan, J., Xie, X., Chen, Z., & Xu, B. (2019). How does code style inconsistency affect pull request integration? An exploratory study on 117 GitHub projects. *Empirical Software Engineering*, *24*, 3871–3903.