Using Provenance for Generating Automatic Citations

Dai Hai Ton That, Andrew Youngdahl, Alexander Rasin, Tanu Malik
School of Computing, DePaul University, Chicago, IL, 60604, USA
{dtonthat, ayoungdahl, arasin, tmalik1}@depaul.edu

Abstract

When computational experiments include only datasets, they could be shared through the Uniform Resource Identifiers (URIs) or Digital Object Identifiers (DOIs) which point to these resources. However, experiments seldom include only datasets, but most often also include software, execution results, provenance, and other associated documentation. The Research Object has recently emerged as a comprehensive and systematic method for aggregation and identification of diverse elements of computational experiments. While an entire Research Object may be citable using a URI or a DOI, it is often desirable to cite specific sub-components of a research object to help identify, authorize, date, and retrieve the published sub-components of these objects. In this paper, we present an approach to automatically generate citations for sub-components of research objects by using the object's recorded provenance traces. The generated citations can be used "as is" or taken as suggestions that can be grouped and combined to produce higher level citations.

Keywords Research Object, Provenance Graph, Automatic Citations, sub-component Citations

1. Introduction

Scientific research relies on the embedded citation of scholarly artifacts within publications. As computational experiments become a scholarly artifact that can be shared, reused, and published, supporting easy and efficient methods for publishing and citing relevant sub-components (files, computer code, outputs, etc.) of experiments becomes crucial. Computational experiments are rarely conducted in isolation; an easy and efficient method to cite sub-components of an experiment can permit authors to give credit where credit is due, and facilitate re-use of the whole, or parts of the computational experiment in a more comprehensive way.

TaPP 2018 July 11 - 12, 2018, London, UK

The current practice is to cite computational experiments using a Digital Object Identifier (DOI). Often DOIs are manually created and assigned. Creating and assigning DOIs for all sub-components of an experiment can be cumbersome. Recent research (Buneman et al. 2016) has focused on automatically generating citations at a more granular level, particularly for large databases—users often use parts of the database for research and so citing the entire database is not helpful. However unlike databases in a computational experiment not all parts are worthy of being cited, and often there is lack of a query framework to identify relevant sub-components. For example, a computational experiment may consist of a data preprocessing step and depending upon the quality of pre-processing a user may or may not want to cite this part of the experiment.

In this paper, we explore how large computational experiments can be automatically cited. In particular, we explore if provenance associated with an experiment can be used to generate automatic citations for the experiment. To obtain provenance we use tools such as Sciunit (Ton That et al.) and Reprozip (Chirigati et al. 2016) that make it easy to package computational experiments, share and repeat them. These tools generate a provenance trace for the experiment, and provide easy and efficient methods to share experiments on scholarly exchange websites such as Figshare (Figshare.com 2017).

Using provenance from these traces can be challenging. In general, associated provenance is fine-grained and too replete for comprehension. Recent work has shown that this fine-grained provenance can be effectively summarized to understand application execution behavior (Yuan et al. 2018). In this paper we assess if available summaries are an effective method to generate citations. The objective of this paper is neither to propose a new method to capture execution traces, nor present a novel summarization algorithm, but to use available methods to determine if a quality citation framework can be supported. Such a citation framework can be useful for tools such as Sciunit used for publishing workflows.

The rest of this paper is presented as follows. Section 2 presents a framework for automatic generation of citations for a computational experiment. As part of this framework, we describe how associated provenance can be summa-

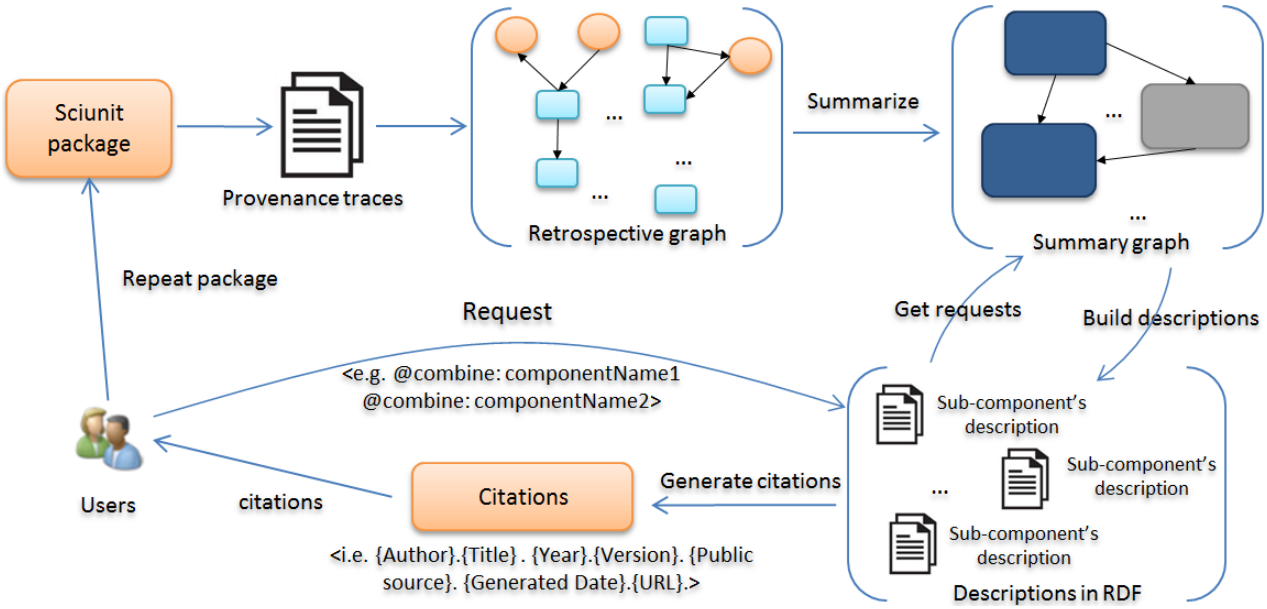

Figure 1. Automatic Generating Citation Framework

rized to generate citations. We compare generated citations with two real-world use cases in which authors have painstakingly assigned citations to their relevant parts of the experiment in Section 4. Finally, we conclude the paper with Section 5.

2. A Citation Generation Framework For Computational Experiments

We define a computational experiment in terms of a *research object*—aggregations of digital artifacts such as code, data, scripts, and temporary experiment results. Research objects are typically created by a user either manually with explicit commands such as those used in RO-Manager (RO- 2016) (a tool that uses the RO-Bundle specification (Soiland-Reyes et al. 2014)), or automatically by using an application virtualization tool such as Sciunit (Ton That et al.) or Reprozip (Chirigati et al. 2016). The advantage of using Sciunit or Reprozip is that a reusable research object is automatically created by simply executing the experiment. In addition, a provenance trace of the computational experiment is immediately available.

This provenance trace is a retrospective trace describing actions that lead to interactions between processes and files, and documenting the flow of data. In general it is fine-grained and too replete for human consumption. Figure 1 describes the architecture of an automatic citation generation framework that uses this available provenance to generate citations. The architecture consists of the following:

1. Record a provenance traces in the research object of the application execution and build a retrospective graph.
2. Summarize the available retrospective graph; this step provides potential citable sub-component groupings.
3. Use the descriptions of the package sub-component groupings that records all the information about the sub-component to create a citation.
4. Adjust the suggested citations.

While the architecture is depicted in context of the Sciunit package, an automatic citation framework may use other methods to generate execution traces.

3. Summarizing Provenance for Citations

A good citation cites resources critical to the experiment as specifically as possible, while avoiding redundancy and also keeping the number of citations to a human readable amount. To achieve this goal, we use the following set of rules to summarize provenance:

We model the generated provenance as a graph $G = (V, E)$ where V and E are sets of nodes and edges respectively. Graph nodes are classified into two types of node: `Activity` node and `Entity` node according to processes or files in the provenance graph.There are three types of edges (i.e., $used$, $wasGeneratedBy$ and $wasInformedBy$) with respect to three types used in W3C PROV standard.

Create provenance summaries according to the following rules:

Rule 1. Common type *(Every node in a group should share the same type).* Given a grouping $\Phi = \{g_1, g_2..., g_k\}$ in G, $\forall u, v \in g_i$, then $Type(u) = Type(v)$.

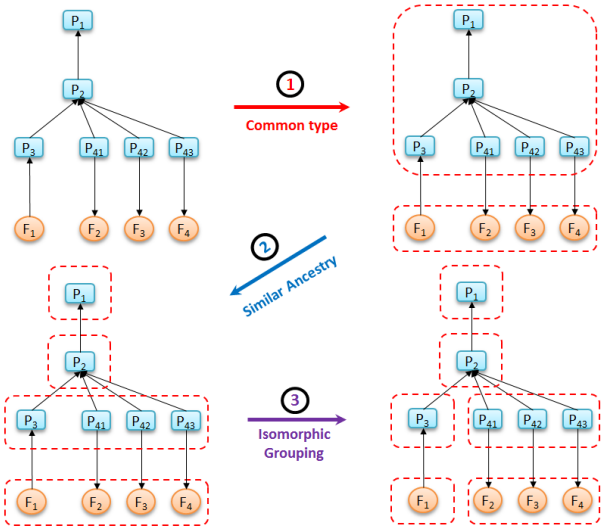

Figure 2. Applying rules on a replete provenance graph to produce summaries

Rule 2. Similar ancestry *(All units (i.e., groups or nodes) in a group have to share the same ancestry).* Given a group g in G, $\forall u_1, u_2 \in g$, then $Ancestor(e_1) = Ancestor(e_2)$.

Rule 3. Isomorphism grouping *(All units in a group should be isomorphic).* Given a group g in G, $\forall u_1, u_2 \in g$, if there is an edge e_1: $\exists e_1 \in E, e_1 = (u_1, x_1)$ *or* (x_1, u_1), then there is a similar edge e_2: $\exists e_2 \in E, e_2 = (u_2, x_2)$ *or* (x_2, u_2), where nodes: x_1, x_2 are in the same group in G and $Label(e_1) = Label(e_2)$.

Figure 2 shows the effect of applying these rules on a replete provenance graph. The net effect is that grouping based on these rules often generates summary graphs that resemble human drawn application workflows. In other words summarizing can be used to obtain prospective provenance from retrospective provenance. These groupings can also provide useful means for publication and citation.

Each summary group can contain many elements which may not be organized hierarchically. To adequately describe these elements we build a single RDF format [1] document for each subgroup. These descriptions are then used in citation generation. For a detailed example of sub-component's description, see our technical report (Ton That et al. 2018).

We create human-readable citations in the following format:

```
{Author}.{Title}.{Year}.{Version}.
{Public source}.{Generated Date}.{URL}.
```

The individual fields (i.e., snippets) of the citation (such as author, title, date, url) are built via operations on the RDF description for the sub-component. Particularly, the snippet information of a citation could be built by using the following rules:

[1] https://www.w3.org/RDF

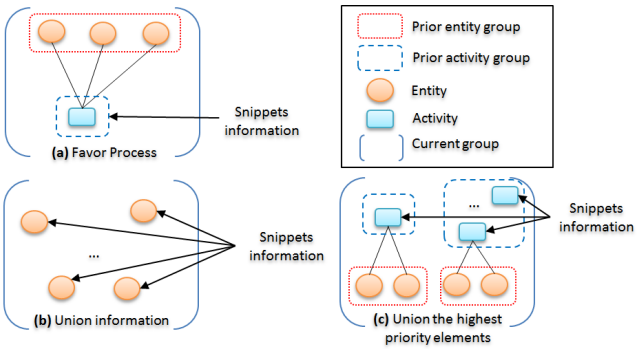

Figure 3. Generating citation examples: (a) Favor process, (b) union information and (c) union the highest priority elements

Rule 4. Favor Activity. The `activity` element has higher priority than `entity` element. It's citation will be presented.

Rule 5. Union information. The information for citation to a component is the union of the highest priority elements within it.

Note that our current graph summarization techniques produce a graph where `activity` node and `entity` node are always separated into different groups. As this may not be ultimately desirable and precludes application of our "favor activity" rule, a user can send requests (as presented in the next part) to combine groups, and thus create a sub-component containing differing types of elements. Figure 3 shows three examples of how the information to build the citation is collected from the elements in a sub-component. An example of a generated citation to a sub-component for a research object (City of Chicago 2017) is shown in Table 1.

3.1 Adjusting the citations

Given available provenance, citations can be automatically generated whenever a research object is published or re-published. However, the generated citations may not be to the author's liking. To address this possibility our system allows the user to either split a sub-component or to combine multiple sub-components by adding a meta description file with the following syntax:

```
To combine: @combine <componentName1> ...
                     <componentNameN>
To split: @split <componentName>
To rename: @rename <oldName> <newName>
```

The citations will be updated when the user re-publishes the container using `Sciunit`.

Furthermore, as mentioned earlier, our current implementation does not automatically create sub-component groupings consisting of both `activity` and `entity` types. If the author desires to have groupings of this nature the author may use the combine functionality to create such groupings.

Table 1. Citation examples

FIE	suggested	`Schenk T. and Leynes G.. Burglary.Rds, Garbage.Rds, Sanitation.Rds, Violation_dat.Rds. 2018. Version 1.0. Hydroshare.com. Generated 03-21-2018. https://www.hydroshare.org/resource/5482e7db02dd4be6b73b177b7caeb8b5/package/5446.`
	manual	`Schenk T. and Leynes G.. Heat map data: Burglary.Rds, Garbage.Rds, Sanitation.Rds, Violation_dat.Rds. 2018. Version 1.0. Hydroshare.com. Generated 03-21-2018. https://www.hydroshare.org/resource/5482e7db02dd4be6b73b177b7caeb8b5/package/heat_map_data.`
DSF	suggested	`Sadler J.. Prepare flood events table. 2018. Version 1.0. Hydroshare.com. Generated 03-21-2018. https://www.hydroshare.org/resource/8db60cf6c8394a0fa24777c8b936f32d/package/3514.`
	manual	`Sadler J.. Script for processing street flood reports. Version 1.0. Hydroshare.com. Generated 03-21-2018. http://dx.doi.org/10.4211/hs.38a4ce62960942b4ad8398ee58a777cf.`

Table 2. Preliminary results in two use cases

	FIE	DSF
# of citations (manually)	11	10
# of suggested citations	9	5
# of matched citations	8 (∼73%)	4 (∼40%)

4. Preliminary evaluation in two research object use cases

We applied our method on two research object use cases: FIE (City of Chicago 2017) and Data-driven street flood severity modeling in Norfolk (DSF) (Sadler 2018). In both these studies, citations were generated a-priori by the respective authors of the study. We used the available computational experiment accompanying these citations to generate provenance traces, and summarized them using our tools to generate sample citations. As shown in Table 2, there are some differences between citations generated by our framework and those manually made by users. The results are quite close in FIE (a 73% match), however in DSF only 4 citations out of 10 match the citations that were built by users. Some of the citation examples are shown in Table 1. The automatically generated citations are sightly different from those manually generated in the `Name` and `URL` snippets. The differences in `Name` are a result of obtaining names directly from sub-component element names as opposed to human authored names. As a correction a user may supply desired names via a meta data file and re-publish the package. The differences in `URL` are due to the differences of publishing methods between our system (i.e., using `Sciunit`) and user (i.e, manually publishing).

5. Conclusion

We have presented our framework for automatic generation of citations of a container in `Sciunit`. Our construction method relies on a summarization technique that decomposes a retrospective graph into sub-components. Experiments show that the suggested citations for the evaluated projects are reasonable. However, the full generation process can still require some manual author intervention to alter citation content in some cases.

Acknowledgments

The authors would like to thank Tom Schenk and Gene Leynes for the FIE use case and Jonathan Goodall and Jeff Sadler for the DFS use case. This work is supported by the National Science Foundation under grants ICER-1639759, ICER-1661918, ICER-1440327 and ICER-1343816.

References

`https://github.com/wf4ever/ro-manager`, 2016. [Accessed 02-May-2017].

P. Buneman, S. Davidson, and J. Frew. Why data citation is a computational problem. *Commun. ACM*, 59(9):50–57, 2016.

F. Chirigati, R. Rampin, D. Shasha, and J. Freire. ReproZip: Computational reproducibility with ease. In *SIGMOD*, 2016.

City of Chicago. Food Inspection Evaluation. `https://chicago.github.io/food-inspections-evaluation/`, 2017. [Online; accessed 05-2017].

Figshare.com. Figshare. `https://figshare.com/`, 2017. [Online; accessed 2-May-2017].

J. Sadler. Data-driven street flood severity modeling in Norfolk, Virginia USA 2010-2016. HydroShare. `http://www.hydroshare.org/resource/9db60cf6c8394a0fa24777c8b9363a9b`, 2018. [Online; accessed 21-Mar-2018].

S. Soiland-Reyes, M. Gamble, and R. Haines. Research object bundle 1.0, 2014. [Online; accessed 2-May-2017].

D. H. Ton That, G. Fils, Z. Yuan, and T. Malik. Sciunits: Reusable research objects. In *eScience 2017*, Auckland, New Zealand.

D. H. Ton That, A. Youngdahl, T. Malik, and A. Rasin. Tech. report. `https://sciunit.run/papers/TechReport.pdf`, 2018.

Z. Yuan, D. H. Ton That, S. Kothari, G. Fils, and T. Malik. Utilizing provenance in reusable research objects. *Informatics*, 5(1), 2018. ISSN 2227-9709. doi: 10.3390/informatics5010014.