German Medical Data Sciences: Shaping Change – Creative Solutions for Innovative Medicine
R. Röhrig et al. (Eds.)

doi:10.3233/SHTI190832

Provenance for Biomedical Ontologies with RDF and Git

Mark R. STÖHR [a,1], Andreas GÜNTHER [a] and Raphael W. MAJEED [a]
[a] *UGMLC, German Center for Lung Research (DZL), Justus-Liebig-University, Giessen, Germany*

Abstract. The German Center for Lung Research (DZL) is a research network with the aim of researching respiratory diseases. In order to enable consortium-wide retrospective research and prospective patient recruitment, we perform data integration into a central data warehouse. The enhancements of the underlying ontology is an ongoing process for which we developed the Collaborative Metadata Repository (CoMetaR) tool. Its technical infrastructure is based on the Resource Description Framework (RDF) for ontology representation and the distributed version control system Git for storage and versioning. Ontology development involves a considerable amount of data curation. Data provenance improves its feasibility and quality. Especially in collaborative metadata development, a comprehensive annotation about "who contributed what, when and why" is essential. Although RDF and Git versioning repositories are commonly used, no existing solution captures metadata provenance information in sufficient detail. We propose an enhanced composition of standardized RDF statements for detailed provenance representation. Additionally, we developed an algorithm that extracts and translates provenance data from the repository into the proposed RDF statements.

Keywords. Biological ontologies, metadata, data curation, automatic data processing, quality improvement.

1. Introduction

1.1. Background

The German Center for Lung Research (DZL) is a consortium of multiple lung research institutions. We pursue the goal to find ways of preventing and curing respiratory diseases. The DZL divides into several disease areas collecting differing data depending on their studies' focus. From a technical point of view, we are confronted with a variety of historically grown site-specific software systems like Excel, Access, CentraXX, Filemaker, SecuTrial, etc. This circumstance hinders researchers from accessing the complete consortium-wide data inventory. That is why we use a central data warehouse (i2b2) to store data from all local databases and information systems, offering one single interface to query all patient related data. The underlying metadata ontology covers concepts related to the lung research domain. Our metadata development is ongoing and technically supported by our Collaborative Metadata

[1] Corresponding Author, Mark R. Stöhr, UGMLC, Justus-Liebig-University, Klinikstraße 36, 35392 Gießen, Germany, E-Mail: mark.stoehr@innere.med.uni-giessen.de

Repository (CoMetaR) tool. It is based on the Resource Description Framework (RDF) for ontology representation and the distributed version control system "Git" for version control [1,2].

We designed the system to satisfy the FAIR principles [3]. Although we assess the principles "Findable", "Accessible" and "Interoperable" as fulfilled, it still misses an important aspect of "Reusability": Provenance, a record of information that enables researchers to track the range of, participants in and reasons for changes that were made to the ontology. The benefits of provenance are not limited to a specific domain, but can be broadly applied to any kind of data [4,5]. It has been shown to be an absolute requirement for data curation processes, since it offers the ability of tracing and reproducing changes by making them auditable, verifiable and reproducible [6,7].

Although Git is often used for knowledge resource versioning, there are only few attempts to extract provenance data.

1.2. Requirements

Through literature research and our own ongoing data curation process, we identified several use cases: (1) during the data curation process, a user needs to inquire with a concept's author and involved people for clarification. (2) For measuring the ontology's progression, we need to calculate its size and number of changes broken down into time intervals. (3) To minimize curation effort, we do not only want to record what concepts have been modified, but exactly what attributes have been added/removed/changed. (4) There are cases in which we need to change a concept's identifier. Thereby, we lose the link between the concept and its precursors, thus cannot track the concept's entire history. To solve this, the respective link needs to be documented in the system. (5) Besides the annotations introduced by Auer et al. [8], we need the information of who else was involved in specific changes. For example, a medical documentalist may act on behalf of a medical investigator. (6) All information must be extracted from the existing GIT repository. (7) The resulting provenance files should contain all relevant information while also being as compact as possible.

2. State of the art

In the field of informatics, the requirement for provenance has been identified and applied to productive systems. There are solutions to provide provenance data based on Git: The Git4Voc project makes use of "hooks", which are script interfaces provided by Git, that execute at certain points during the data upload process [9]. The Git2PROV project extracts basic information about repository versions such as the author and a description [10]. The Quit Store is a multi-layered system, that uses Git as a backend tool for versioning [11].

We found several publications for analyzing and applying provenance principles in the domain of medical informatics, e.g. by McGovern [12] and Sahoo [13]. They usually refer to provenance of clinical data analysis, study design, workflow management, etc., but none of them addresses the development of metadata itself.

Gonçalves et al. presented an algorithm for detecting and presenting changes between OWL ontologies [14]. Although they identify additions and removals down to an atomic level, they do not serialize them in a standardized way. Auer et al. present one approach to record RDF Knowledge Base data evolution on an atomic level [8].

For enhancing human change review, they annotate changes with information about what changes have been done, the editing user, timestamp, documentation and the bundle to which a change belongs. They introduce the "log" namespace, which is one of multiple frameworks dealing with the task of annotating changes of data in a standardized way. Other examples are EvoRDF [15], the Quit Store [11], Delta [16] and ChangeSet [17].

The latest standard by the World Wide Web Consortium (W3C) to record provenance data is the PROV document, including the sub-document PROV-O for ontologies [18]. This framework is increasingly applied in the international research community for medical informatics [13,19,20]. Its advantage is the adaptiveness to many fields through its generic design. By itself, it is not specific enough to describe (meta-) data changes on an atomic level, but PROV relations are designed to be qualifyable with additional information. The three main classes are Agents, Actions and Entities. In our case, we deal with an ontology containing concepts (entities) on which the changes (actions) are performed by medical staff (agents).

3. Concept

The following figure illustrates a typical workflow, in which three users simultaneously edit the RDF ontology. To gather provenance information about an ontology state and the changes made, the states' precursors have to be identified.

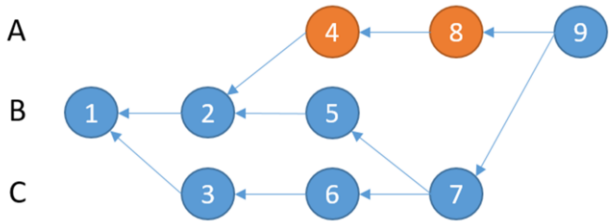

Figure 1 – Example of ontology versioning and merging. Blue circles: successfully uploaded ontology versions; red circles: failed uploads; arrows: succession.

Explanation: User B is author of ontology state 1. User C fetches this version and both users make changes to state 1, resulting in state 2 and 3. User B immediately pushes his changes to the repository and User A fetches those (4). After users B and C made changes to their local copies (5,6), User B pushes his state to the server again. When User C does the same, he first has to merge his state with the repository's state, resulting in state 7. Then User A pushes his changes (8), first having to merge with the repository's state as well, resulting in the comprehensive state 9. State 4 and 8 contain syntactic errors. Our Git server, which performs consistency checks on every submission, will not propagate these states as new ontology versions. The following algorithm shows how we extract the necessary information from the Git repository.

3.1. The Algorithm

Our algorithm is meant to create a provenance file for either the latest ontology file upload or a specific time interval. The following steps are taken for all ontology stages uploaded in the given time interval. In the first case, only the latest stage is processed.

3.1.1. Identifying Precursors for Comparison

To determine the exact changes made to RDF triples from one state to the next, we need to identify the state's precursors. Figure 1 shows all relevant cases: To get the changes made in state 2, we need to compare it to state 1. In order to get the changes in state 7, we have to compare it to state 5 as well as state 6. In case of an incorrect state, comparisons have to be made to their precursor. Thus, in order to get the changes in state 9, it is compared to state 2 and 7.

3.1.2. Extracting the Current and Precursors' State

To gather all content of the selected ontology states, we parse and translate all RDF data of one state into an EAV schema and save it in a delimited text file format.

3.1.3. Generating the Delta File

In order to compare two or more ontology states, we filter all text lines that occur in the new state but in neither of the precursors. This way, we collect all additions and, vice versa, we find removals by looking at what occurred in all precursors but not in the new state. Besides the delta files we also collect information about the ontology states (Git commit ID, Git parent commit IDs, timestamp, author's name, Git commit message) and the authors (author's name and author's e-mail-address).

3.1.4. Translation into Standardized Serialization

We decided to combine the highly interoperable PROV standard with the ChangeSet standard offering finer granularity. Both standards are published by the W3C [17,18]. The delta files, as well as the files containing authors and ontology states information, are translated into PROV-O files in turtle syntax.

The introduced algorithm extracts RDF statements that have been added or removed from one ontology version to the next. From this information, one may infer statements about what concepts have been added to, removed from or modified in the ontology. A concept's provenance information may be inferred without difficulty, as long as the RDF subject – in other words the concept's identifier – stays the same. If an identifier changed from one version to another, this information has to be stated explicitly in the RDF files.

4. Implementation

We perform the concept algorithm on a Debian Linux server with shell scripts and a java-based RDF parser: For extracting an ontology state, we parse its RDF files with Apache Jena Fuseki and save all available tuples through Fuseki's SPARQL interface

with the tool “curl”. Furthermore, we use standard tools like “grep” for delta file generation and “awk” to translate the delta files into provenance RDF.

4.1. Provenance File Content

Figure 2 shows example provenance statements we generated. The provenance information are partly extracted from the Git commit metadata via “git log”-command and partly from the delta files previously described. Git by itself offers the author username, e-mail-address, a 40-letter-ID for every commit, a timestamp and the previous commits’ IDs. The delta files contain the information whether a triple was added or removed, the affected concept (subject), attribute (predicate) and characteristic (object).

```
:ontology a prov:Entity .
:Stöhr a prov:Agent, foaf:Person ;
        foaf:mbox "mark.stoehr@innere.med.uni-giessen.de" .
:Majeed a prov:Agent, foaf:Person ;
        foaf:mbox "raphael.majeed@chiru.med.uni-giessen.de" .
:commit_6c365b51ef866b09f0a10e1e6b16546ffe746295 a prov:Activity ;
        prov:wasAssociatedWith :Stöhr ;
        prov:endedAtTime "2018-10-01T12:00:00+02:00"^^xsd:dateTime ;
        prov:label "Added concept for emphysema index." ;
        prov:actedOnBehalfOf :Majeed ;
        prov:wasInfluencedBy :commit_5af9416541b64f2471b54f5f40fe1d7cebafbe5f ;
        prov:generated [ a prov:Entity; prov:specializationOf :ontology ] ;
        prov:qualifiedUsage [
                a prov:Usage, cs:ChangeSet ;
                cs:addition [
                        a rdf:Statement;
                        rdf:subject :emphysemaIndex;
                        rdf:predicate rdf:type ;
                        rdf:object skos:Concept
                ] ;
        ] ;
        prov:used :emphysemaIndex .
:emphysemaIndex a prov:Entity .
```

Figure 2 – Example provenance statements in turtle N3 syntax. Git metadata in upper blue box, delta file data in lower green box.

Figure 2 illustrates most of the attributes we use in our provenance files. Additionally, we link two concept identifiers through the “prov:wasDerivedFrom” attribute with the new identifier as subject and the old identifier as literal value. Since this information cannot be extracted from the Git repository, it must be entered manually in the RDF files. Sometimes users edit the ontology on behalf of someone else. There is no dedicated field in Git that allows naming additional authors, but there are established conventions in the development community: the “commit message” is a field to describe the changes that were made. Co-authors may be appended to this message in a structured way.

4.2. Performance

In our DZL ontology’s first state at the end of 2016, it contained 637 statements. At the end of 2017, it contained 4069 statements. In November 2018, it contains 6081 statements. In total, 11866 additions and 5536 removals were performed. The overall size of our provenance files is approximated 4 megabytes. In its current state, one ontology version uses around 400 kilobytes of space.

The program for generating the provenance file from 2017-11-01 until 2018-11-01 takes 9 minutes and 19 seconds on our Debian Linux server. This includes loading into the triple store, exporting and comparing the versions, serializing and saving all extracted information. For the current single commit, this procedure takes 3.84 seconds.

4.3. Tracking Identifier Changes

When we implemented the algorithm, we retrospectively identified 67 concept identifier changes, which had to be annotated. We found them by searching through the Git repository's history tool, which lets users search for strings in added and removed lines. After implementation, changes in identifiers are annotated simultaneously during continuous ontology development. So far, 143 identifier changes have been annotated.

5. Lessons learned

In this article, we proposed a composition of standardized statements for provenance data representation. When looking at the W3 PROV documentation we find that a "prov:Activity" which modified a "prov:Entity" is usually meant to have "prov:used" an existing "prov:Entity" and "prov:generated" a new one. For our setup that would mean we had to keep a copy of every ontology version and its concepts. Since this information is stored in the repository, we decided to have only one "Entity" for every concept, which is referenced by the commit "Activity". Otherwise, we would create redundancy and file sizes would get out of hand.

In order to further simplify the curation process and make modifications even clearer, we consider classifying changes, e.g. structural, semantic and literal.

Our current work was focused on the algorithmic extraction, translation and standardized representation of provenance data. Its visualization is the next step to make information available for target users. There are several tools for visual representation of provenance. We will investigate in how far they can further support the curation process.

Since the program takes only few seconds for a single commit, it is reasonable to append it to Git's "after-push-hook" which already includes loading the data into our data warehouse. This way, all ontology data is always available together with complete provenance information.

Like other researchers, we found that blank nodes need a special treatment [11,8]. In our case, blank nodes would impede the comparison of ontology states. Thus, as a limitation, our algorithm demands that the underlying RDF data contains no blank nodes.

Concept identifier changes may lead to misinterpretation of the data delivered by our algorithm. Not only does the annotation of identifier changes need manual effort, but also new challenges may occur, e.g. if an identifier is reintroduced later for a different concept. Smart algorithms may identify derivations and reintroductions automatically with help of metric definitions and temporal relations.

In the introduction, we named several existing software solutions, of which we consider the Quit Store being the most advanced. Compared to other solutions, it provides detailed information about modifications down to the atomic RDF triple level, but to achieve this they introduced a proprietary namespace. Additionally, due to its

architecture, it is not applicable to a raw Git setup. All operations like merging and synchronization of data are done through the Quit API and the Repository Manager.

6. Conclusion

We proposed an enhanced composition of standard RDF statements for provenance data representation. Additionally, we developed an algorithm to extract and store provenance data from a Git repository accordingly. This algorithm works implementation-independent on top of ontology management system using Git and RDF. It can be applied to ontology versioning systems that are already in place, since the algorithm can backtrack changes up to the first ontology state. Because we use standard terminologies, the resulting data is reusable for any application with an appropriate standard interface. From the ontology-editing user's point of view, the development process remains the same except for additional steps in case of changes in identifiers. The software runs unnoticeably in the background. All source code is publicly accessible under Github: https://github.com/dzl-dm/cometar

Conflict of Interest

The authors state that they have no conflict of interests.

References

[1] M.R. Stöhr, et al., CoMetaR: A Collaborative Metadata Repository for Biomedical Research Networks, *Studies in health technology and informatics* **245** (2017), 1337.
[2] M.R. Stöhr, R.W. Majeed, A. Günther, Using RDF and Git to Realize a Collaborative Metadata Repository, *Studies in health technology and informatics* **247** (2018), 556–560.
[3] M.D. Wilkinson, et al., The FAIR Guiding Principles for scientific data management and stewardship, *Scientific data* **3** (2016), DOI: 10.1038/sdata.2016.18.
[4] P. Groth, et al., Requirements for Provenance on the Web, *International Journal of Digital Curation* **7(1)** (2012), 39–56.
[5] E.D. Ragan, et al., Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purposes, *IEEE transactions on visualization and computer graphics* **22(1)** (2016), 31–40.
[6] V. Curcin, et al., Implementing interoperable provenance in biomedical research, *Future Generation Computer Systems* **34** (2014), 1–16.
[7] B. Baum, et al., Opinion paper: Data provenance challenges in biomedical research, *it - Information Technology* **59(4)** (2017), DOI: 10.1515/itit-2016-0031.
[8] S. Auer, et al., A Versioning and Evolution Framework for RDF Knowledge Bases, *Perspectives of Systems Informatics* (2007), 55–69.
[9] L. Halilaj, et al., Git4Voc: Collaborative Vocabulary Development Based on Git, *International Journal of Semantic Computing* **10(02)** (2016), 167–191.
[10] T. de Nies, et al., Git2PROV: Exposing Version Control System Content as W3C PROV, *Poster and Demo Proceedings of the 12th International Semantic Web Conference* (1035) (2013), 125–128.
[11] N. Arndt, et al., Decentralized Collaborative Knowledge Management using Git, *Journal of Web Semantics* (2018), DOI: 10.1016/j.websem.2018.08.002.
[12] A.P. McGovern, et al., Glucose test provenance recording in UK primary care: was that fasted or random?, *Diabetic medicine : a journal of the British Diabetic Association* **34(1)** (2017), 93–98, DOI: 10.1111/dme.13067.

[13] S.S. Sahoo, J. Valdez, M. Rueschman, Scientific Reproducibility in Biomedical Research: Provenance Metadata Ontology for Semantic Annotation of Study Description, *AMIA ... Annual Symposium proceedings. AMIA Symposium* (2016), 1070–1079.
[14] R.S. Gonçalves, B. Parsia, U. Sattler, Ecco: A Hybrid Diff Tool for OWL 2 ontologies, *CEUR Workshop Proceedings* **849** (2012).
[15] E. Blomqvist, et al., EvoRDF: A Framework for Exploring Ontology Evolution, *The Semantic Web: ESWC 2017 Satellite Events* (2017), 104–108.
[16] Tim Berners-Lee, Dan Connolly, Delta: an ontology for the distribution of differences between RDF graphs, 2001 [Last accessed: 05/11/2018], https://www.w3.org/DesignIssues/Diff.
[17] W3C Semantic Web, DatasetDynamics/ChangeDescriptionVocabulary, 2011 [Last accessed: 05/11/2018], https://www.w3.org/wiki/DatasetDynamics/ChangeDescriptionVocabulary.
[18] W3C Semantic Web, PROV-O: The PROV Ontology [Last accessed: 05/11/2018], https://www.w3.org/TR/2013/REC-prov-o-20130430/.
[19] A.-K. Kock-Schoppenhauer, et al., Practical Extension of Provenance to Healthcare Data Based on the W3C PROV Standard, *Studies in health technology and informatics* **253** (2018), 28–32.
[20] P. Ciccarese, et al., PAV ontology: provenance, authoring and versioning, *Journal of biomedical semantics* **4(37)** (2013), DOI: 10.1186/2041-1480-4-37.