Quit Diff

Calculating the Delta Between RDF Datasets Under Version Control

Natanael Arndt
Universität Leipzig
Augustusplatz 10
04109 Leipzig, Germany
arndt@informatik.uni-leipzig.de

Norman Radtke
Universität Leipzig
Augustusplatz 10
04109 Leipzig, Germany
radtke@informatik.uni-leipzig.de

ABSTRACT

Distributed actors working on a common RDF dataset regularly encounter the issue to compare the status of one graph with another or generally to synchronize copies of a dataset. A versioning system helps to synchronize the copies of a dataset, combined with a difference calculation system it is also possible to compare versions in a log and to determine, in which version a certain statement was introduced or removed. In this demo we present *Quit Diff*[1], a tool to compare versions of a *Git* versioned quad store, while it is also applicable to simple unversioned RDF datasets. We are following an approach to abstract from differences on a syntactical level to differences on the level of the RDF data model, while we leave further semantic interpretation on the schema and instance level to specialized applications. *Quit Diff* can generate patches in various output formats and can be directly integrated in the distributed version control system *Git* which provides a foundation for a comprehensive co-evolution work flow on RDF datasets.

1. INTRODUCTION

The problem of co-evolution is the synchronization of distributed datasets under consideration of their independent evolution, this problem is further described in [2]. In software development this problem also exists for source code directories and is targeted by distributed version control systems such as *Git* or *Mercurial*. In this demo we present *Quit Diff*, which can be integrated into an existing *Git* system as *difftool*, which allows to compare different versions of an RDF dataset. In contrast to the syntactical changes, printed by `git diff`, *Quit Diff* is able to calculate the actual differences of the RDF data model and in turn displays them in various RDF formats and as SPARQL Update queries. *Quit Diff* also leads us towards a future implementation of *Quit Merge*, which should resolve merges of commits in an RDF compatible way independent of syntactical changes.

[1]Code repository: https://github.com/AKSW/QuitDiff

SEMANTiCS 2016 September 12–15, 2016, Leipzig, Germany
© 2016 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4752-5/16/09.
DOI: http://dx.doi.org/10.1145/2993318.2993349

In the distributed version control system *Git*, individual transactions are organized as commits, containing the status of the repository at that point in time and a reference to the direct ancestor commit. With *Quit* [2] we already work towards a quad store with complete versioning and collaboration support for RDF datasets in *Git* repositories. *Quit Diff* in addition allows the exploration of the versioning log by calculating the differences between arbitrary commits of the *Git* history and printing them in various formats. For expressing differences between RDF datasets multiple formats exist, such as the changeset vocabulary [12], the eccrev vocabulary [7] or TopBraid's diff vocabulary[2]. But also SPARQL 1.1 Update [8] queries can be used to specify a set of added and deleted triples. These diff-formats can help to transmit and apply only the changes of datasets as patches instead of copying the complete store. This role of diff as foundation for distributed versioning, co-evolution and collaboration on structured data is also discussed and proposed in [4] and [1].

```
@prefix changeset: <http://purl.org/vocab/changeset/schema#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix aksw: <http://aksw.org/> .
```

Listing 1: List of all prefixes used in this document

Throughout the paper we are referring to several RDF terms using simplified QNames[3]. The prefix definitions are as defined in listing 1.

1.1 State of the Art

Berners-Lee and Connolly [10] give a general overview on the problem of calculating delta on RDF graphs and the problem of synchronization. Franconi et al. [4] suggest a formalized model for differences of versions of an evolving knowledge base. Especially two requirements for such an approach are formulated, users need access to a specific version of the knowledge base and need to get the difference between versions. From these requirements two characteristics of the system are derived, i.e. that the system should not store all versions, but a *core* from which all versions can be reconstructed, and to determine the difference in

[2]http://topbraid.org/diff#
[3]https://www.w3.org/TR/2009/REC-xml-names-20091208/

meaning between versions in addition to the syntactical difference. In our eyes the first characteristic is unnecessary since it excludes systems which store all versions and use other means then diff to enable the storage of all versions (e.g. *Git*). Besides that the second characteristic formulates a core requirement for diff tools on knowledge bases.

Beside syntactical diffs there are important approaches of ontology evolution and versioning using description logic expressed with OWL [13]. Zaikin and Tuzovsky [14] describe two tools they have developed to target this problem. The first tool *owl2diff*[4] detects changes between different versions of OWL ontologies and the second tool *owl2merge* helps "to resolve conflicts and perform a three-way merge". Both tools can be integrated to be used with *Git*.

Various tools for comparing RDF graphs or datasets are further listed in the Semantic Web Activity Wiki[5]. The *SemDiff*[6] web service can compare individual Linked Data resources and output the result in an RDF file containing either all added or deleted statements, or summarized as RSS or RDF, wile the changed statements are encoded in comments in a custom string format. Also *rdf-utils*[7] outputs the result in two ZIP compressed RDF files containing all added respective deleted statements. *rdfdiff* from redland raptor[8] produces a human readable list of added and deleted triples. The *TopBraid Composer*[9] in the payed version, expresses patches using the diff ontology[2], while it also provides a graphical user interface for exploring the changes. The *GUO Graph Diff* web service is no longer functionally available[10], but it provided a custom RDF format for differences as well. None of these tools provides comparison of RDF datasets resp. multiple RDF models at once. A general option, which is also supported in [2], is to serialize the files to compare in a canonical form and use a line based diff tool.

The RDFLib Python API provides an *rdflib.compare*[11] module, which allows calculating the difference between two models. This module is used for implementing *Quit Diff*.

2. THE QUIT DIFF TOOL

The *Quit Diff* tool is designed as a stand alone diff tool for RDF datasets, as well as drop-in replacement for `git diff` respective for usage with `git difftool` in *Git* repositories containing RDF datasets. Due to the implementation as command line tool, it could also be integrated as back-end for compare operations in graphical RDF editing tools, such as protégé[12] or OntoWiki[6, 5]. The main features of *Quit Diff* are not only to compare triples contained in individual RDF graphs, but also RDF datasets in quad serialization formats, as well as in a directory of multiple files, integration in *Git* and support for various output formats: three patch formats and sparql update queries.

2.1 Diff

A diff utility on file systems is used to show the differences between two files (typically but not necessarily between two different versions of the same file) in a line-based approach. The diff tool shipped with Unix in the early 1970s was described by their programmers [9] and produced an output, called *diff* or *patch*, containing the lines that changed between the two versions. Those changes may be deletions or insertions of lines (changes of a line are expressed as a combination of deletion and insertion). The file which is given at the first position is interpreted as the old version of the file, while the second file is the new version.

A common usage of the diff tool is to compare two versions of a program code file in order to determine the actual changes or to generate a patch[13]. Transferring and applying this patch to an existing source code repository is easier and more flexible then reinitializing a complete new file structure.

Since for RDF several serialization formats are available, it is possible to use a similar workflow for RDF datasets, as it is common for program source code. There might also be use cases, such as versioning of RDF based data, where the syntactical diffs may be sufficient to reconstruct each version of an RDF file. Using delta versioning with canonical skolemized RDF serialization formats [3] with N-Triples or N-Quads produces line based diffs, which are equivalent to the triple based changes of the serialized RDF graphs. But this approach won't always give all necessary information e.g. for formats like RDF-XML or Turtle. In order to generate patches between different versions of RDF based data, we decided to take an approach to generate logical diffs.

When comparing RDF Datasets it is further relevant to find the correct graphs to compare. If both, the remote and the local, are RDF Datasets it is straight forward to compare each contained graph with the graph identified by the same URI respectively. While if one compares an RDF Graph serialization with an RDF dataset file, the Graph can only be compared with the default graph of the RDF dataset.

2.2 Integration in Git

Quit Diff is designed to be integrated in *Git* as a so called *difftool*. This enables the user to call it by executing the command `git difftool` or `git diff`. The differences between these two approaches and the advantages to couple *Quit Diff* with *Git* is to directly navigate on the complete *Git* versioning graph and comparing arbitrary commits. Further the `--dir-diff` option for `difftool` allows us to compare complete directories as RDF datasets with quad serialization formats.

Listing 2 shows a section of the `~/.gitconfig` file to integrate *Quit Diff* with *Git*. Customizing the `.gitattributes`[14] file in the repository also allows one to set *Quit Diff* as default diff tool for specific file types. This happens with lines like `*.nq diff=quitdiff`.

[4]https://github.com/utapyngo/owl2vcs

[5]How to diff RDF: https://www.w3.org/2001/sw/wiki/How_to_diff_RDF

[6]http://onto.rpi.edu/sw4j/diff.html

[7]http://sourceforge.net/projects/knobot/files/rdf-utils/

[8]http://librdf.org/raptor/

[9]http://www.topquadrant.com/tools/IDE-topbraid-composer-maestro-edition/

[10]https://web.archive.org/web/20100407030604/http://webr3.org/diff/

[11]https://github.com/RDFLib/rdflib/blob/master/rdflib/compare.py

[12]http://protege.stanford.edu/

[13]This can be achieved with `diff -Naur old new > changes.patch` which can then be applied with the `patch` utility

[14]The file might not exist, but can just be created in the repository. https://git-scm.com/book/en/v2/Customizing-Git-Git-Attributes

```
[diff "quitdiff"]
    command = quitdiff.py

[difftool "quitdiff"]
    cmd = quitdiff.py --diffFormat sparql --local=\"$LOCAL\" ↵
      --remote=\"$REMOTE\" --merged=\"$MERGES\" --base=\"↵
      $BASE\"
```

Listing 2: Configuring *Quit Diff* to be used with *Git*

3. THE QUIT DIFF APPROACH

The *Quit Diff* tool can calculate the difference between RDF datasets independent of the used serialization formats. For example *Quit Diff* is able to compare a Turtle serialized RDF graph with an RDF/XML serialized RDF graph or multiple RDF graphs in different files with a quad serialized RDF dataset.

The delta between the individual graphs is then determined in two sets each, added and removed triples per graph. For the output, multiple output formats are supported, three changeset ontologies and SPARQL 1.1 Update queries.

The diff tool expects two parameters, *left* and *right*, that may be a path of a file or a path of a directory. In our example let there be a directory *profiles* containing two files, *arndt.nq* (listing 3) and *radtke.ttl* (listing 4) describing the authors of this paper. In case of a directory the Diff Tool will analyze all containing files and distinguish between RDF serialized files and the rest. RDF serialized files will be detected by their suffix and will be loaded in a separate ad-hoc in-memory quad store for each *left* and *right*.

After the loading procedure is completed the two resulting stores are canonicalized using the `IsomorphicGraph()` class from the `rdflib.compare`[11] package. The algorithm is based on the work of Sayers & Karp [11]. To get two comparable canonicalized graphs the triples of each named graph and of the default graph (if no named graph is given e.g. for triple serializations) are collected for each directory *left* and *right*.

```
<http://aksw.org/NatanaelArndt> <http://www.w3.org↵
  /1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/↵
  Person> <http://aksw.org/> .
<http://aksw.org/NatanaelArndt> <http://xmlns.com/foaf/0.1/↵
  mbox> <mailto:arndt@informatik.uni-leipzig.de> <http://aksw↵
  .org/> .
```

Listing 3: A N-Quads file (in revision A) describing Natanael Arndt

```
<http://aksw.org/NormanRadtke> a  foaf:Person ;
    foaf:mbox <mailto:radtke@informatik.uni-leipzig.de> .
```

Listing 4: A Turtle file (in revision A) describing Norman Radtke

To create a new revision B on top of A we add information to the two files assuming the authors know each other. If we run *Quit Diff* to create a patch between these two revisions, one possible result would be like the one shown in listing 5

```
<urn:changeset:453a9f60> a changeset:ChangeSet ;
    changeset:addition [ a rdf:Statement ;
            rdf:object aksw:NormanRadtke ;
            rdf:predicate foaf:knows ;
            rdf:subject aksw:NormanRadtke ] .

<urn:changeset:453a9f61> a changeset:ChangeSet ;
    changeset:addition [ a rdf:Statement ;
            rdf:object aksw:NormanRadtke ;
            rdf:predicate foaf:knows ;
            rdf:subject aksw:NatanaelArndt ] ;
    changeset:subjectOfChange <http://aksw.org/> .
```

Listing 5: A diff between revision A and B serialized as a patch, expressed using the changeset ontology

Going on from revision B to revision C we add a triple to the N-Quads file of N. Arndt containing the phone number. The same information is added to the Turtle file of N. Radtke resulting in the files shown in listing 6 and listing 7. An example patch showing the difference between revision A and C as a SPARQL 1.1 Update query is given in listing 8

```
<http://aksw.org/NatanaelArndt> <http://www.w3.org↵
  /1999/02/22-rdf-syntax-ns#type> <http://xmlns.com/foaf/0.1/↵
  Person> <http://aksw.org/> .
<http://aksw.org/NatanaelArndt> <http://xmlns.com/foaf/0.1/↵
  knows> <http://aksw.org/NormanRadtke> <http://aksw.org/> .
<http://aksw.org/NatanaelArndt> <http://xmlns.com/foaf/0.1/↵
  mbox> <mailto:arndt@informatik.uni-leipzig.de> <http://aksw↵
  .org/> .
<http://aksw.org/NatanaelArndt> <http://xmlns.com/foaf/0.1/↵
  phone> <tel:+49-341-9732323> .
```

Listing 6: A N-Quads file (in revision C) describing Natanael Arndt

```
<http://aksw.org/NormanRadtke> a  foaf:Person ;
    foaf:knows <http://aksw.org/NormanRadtke> ;
    foaf:phone <tel:+49-341-9732323> ;
    foaf:mbox <mailto:radtke@informatik.uni-leipzig.de> .
```

Listing 7: A Turtle file (in revision C) describing Norman Radtke

```
insert data {
    aksw:NormanRadtke foaf:phone <tel:+49-341-9732323> .
    aksw:NormanRadtke foaf:knows aksw:NormanRadtke .
    aksw:NatanaelArndt foaf:phone <tel:+49-341-9732323> .

    graph <http://aksw.org/> {
        aksw:NatanaelArndt foaf:knows aksw:NormanRadtke .
    }
}
```

Listing 8: A diff generated as SPARQL Update

Since *Quit Diff* should allow its users to further exploit the information gained from a `git diff` operation, it offers multiple output formats. The following ways of expressing patches or changes between two RDF datasets are supported so far: SPARQL 1.1 Update query (see listing 8), Changeset Ontology (see listing 5), Topbraid and Eccenca Revision. The most straight forward format may be the SPARQL 1.1 Update query under the condition that a SPARQL endpoint with update functionality is available. The other formats need an implementation to apply the patches and they would allow querying and reasoning on the RDF model. The last point makes it a lot easier to identify commits that changed a discrete triple.

4. TEST OF THE APPROACH

To prove our concept we developed a test for our implementation that can be explained as a time machine that reverts all existing commits of a repository by executing SPARQL queries generated with *Quit Diff*. The files necessary for reproducing the test are included in the code repository of *Quit Diff*[1]. At first we took a *Git* repository that contains RDF data. With *Quit Diff* included in the *difftool* configuration of *Git*, we created SPARQL update patches of all pairs in the reverse order, i.e. of commits and their preceding commits. For each pair beginning with the pair *HEAD* and *HEAD~1*[15] going on with *HEAD~1* and *HEAD~2* we saved the resulting queries in the order they were created. In our case we chose n to run from 0 to 99 to create 100 queries.

After this step we had to load the data into a triple store that also provides a SPARQL update endpoint. We decided to use the *Quit Store* [2] which creates a commit for each update on the store, to apply the SPARQL Update queries. In a batch process we executed every SPARQL Update query we generated in the step before.

To validate the functionality of *Quit Diff* we had to compare the commits before *HEAD* and the commits after *HEAD*. Since we generated and executed 100 queries we can easily identify the commit that was *HEAD* before we started to apply our patches with *Git's* syntax *HEAD~100*. The commits *HEAD~99* and *HEAD~101* should result in an empty diff because commit *HEAD~101* led to commit *HEAD~100* and *HEAD~99* reverted the commit again, assuming *Quit Diff* generated the correct patch. The same assumption can be made for all pairs of commits *HEAD~100+n*, *HEAD~100-n* with $n \in \{0, 1, ..., 99\}$ because the patches were generated sequentially. In our case we tested all pairs with *Quit Diff* and got empty diffs.

5. CONCLUSION

In this demo paper we have presented *Quit Diff* a tool, which can be integrated into the distributed version control system *Git* to explore changes of RDF datasets under version control. In comparison to the diff tool provided by most systems, and the one coming with *Git*, *Quit Diff* interprets the RDF datasets and calculates the actual changes according to the RDF data model. The tool supports multiple output formats, which can be used in different use cases. Further the proof of concept was verified by reverse applying the changes, calculated by *Quit Diff*, on the same repository.

The work on *Quit Diff* is part of the aim for a full co-evolution aware store and tool stack. The co-evolution store is explained in detail in [2], while the implementation of *Quit Merge*, which should resolve merges of commits in an RDF compatible way independent of syntactical changes, is part of the future work.

6. ACKNOWLEDGEMENTS

This work was partly supported by the following grants from the German Federal Ministry of Education and Research (BMBF) for the LEDS Project under grant agreement No 03WKCG11C and the European Union's Horizon 2020 research and innovation programme for the SlideWiki Project under grant agreement No 688095.

[15] *HEAD~n* is the *n*th ancestor commit of *HEAD*, while *HEAD* is the currently checked out commit in a repository

7. REFERENCES

[1] N. Arndt, K. Junghanns, R. Meissner, P. Frischmuth, N. Radtke, M. Frommhold, and M. Martin. Structured feedback: A distributed protocol for feedback and patches on the web of data. In *Proceedings of the Workshop on Linked Data on the Web co-located with the 25th International World Wide Web Conference (WWW 2016)*, volume 1593 of *CEUR Workshop Proceedings*, Montréal, Canada, Apr. 2016.

[2] N. Arndt, N. Radtke, and M. Martin. Distributed collaboration on rdf datasets using git: Towards the quit store. In *12th International Conference on Semantic Systems Proceedings*, SEMANTiCS '16, Leipzig, Germany, Sept. 2016.

[3] R. Cyganiak, D. Wood, and M. Lanthaler. Rdf 1.1 concepts and abstract syntax. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, Feb. 2014.

[4] E. Franconi, T. Meyer, and I. Varzinczak. Semantic diff as the basis for knowledge base versioning. In *13th international workshop on Non-Monotonic Reasoning*, Toronto, Canada, May 2010.

[5] P. Frischmuth, N. Arndt, and M. Martin. OntoWiki 1.0: 10 years of development - what's new in OntoWiki. In *Proceedings of the Poster & Demos Track of the 12th International Conference on Semantic Systems*, CEUR Workshop Proceedings, Leipzig, Germany, Sept. 2016.

[6] P. Frischmuth, M. Martin, S. Tramp, T. Riechert, and S. Auer. OntoWiki—An Authoring, Publication and Visualization Interface for the Data Web. *Semantic Web Journal*, 6(3):215–240, 2015.

[7] M. Frommhold, R. N. Piris, N. Arndt, S. Tramp, N. Petersen, and M. Martin. Towards Versioning of Arbitrary RDF Data. In *12th International Conference on Semantic Systems Proceedings*, SEMANTiCS '16, Leipzig, Germany, Sept. 2016.

[8] P. Gearon, A. Passant, and A. Polleres. Sparql 1.1 update. https://www.w3.org/TR/2013/REC-sparql11-update-20130321/, Mar. 2013.

[9] J. W. Hunt and M. McIllroy. An algorithm for differential file comparison. Technical Report 41, AT&T Bell Laboratories Inc., 1976.

[10] T. B. Lee and D. Connolly. Delta: an ontology for the distribution of differences between rdf graphs. Technical report, W3C, 2001.

[11] C. Sayers and A. H. Karp. Computing the digest of an rdf graph. Technical report, HP Laboratories, Palo Alto, USA, Mar. 2004.

[12] S. Tunnicliffe and I. Davis. Changeset Vocabulary. http://purl.org/vocab/changeset/schema#, May 2009.

[13] W3C OWL Working Group. OWL 2 Web Ontology Language document overview. https://www.w3.org/TR/2012/REC-owl2-overview-20121211/, Dec. 2012.

[14] I. Zaikin and A. Tuzovsky. Owl2vcs: Tools for distributed ontology development. In *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) co-located with 10th Extended Semantic Web Conference (ESWC 2013)*, volume 1080 of *CEUR Workshop Proceedings*, Montpellier, France, May 2013.