Provit Documentation

Release 1.0.2

Universitätsbibliothek Leipzig

Nov 27, 2019

Contents:

This is the documentation for the provenance integration tools (provit). It consinsts of a command line interface, a browser frontend and a python package which you can include into your project to track your provenance from within your software. Feel free to look around and contact us, if you have any questions or find mistakes.

CHAPTER 1

provit - Provenance Integration Tools

provit is a data provenance annotation and documentation tool. It provides various feature for creation and retrieval of provenance information for data stored in files. The tracking of sources, modifications and merges allows the user to keep a log of all modifications a dataset was subject to. This is especially useful for dataset which are accessed intermittently or part of a long running workflow (e.g. for a scientific thesis). Furthermore, provenance data stored next to the data in an archive can help others to identify quality, value and acutality of the data.

provit does not require any external infrastructure. All information is stored in *.prov* files right next to the data files as a JSON-LD graph. This makes it the perfect tool for small teams or individual researchers.

To allow interoperatibility, a small subset of the W3C PROV-O vocabulary is implemented. Therefore, the provenance information can easily be merge in a linked data graph if necessary, at a later stage of the project.

provit aims to provided an easy to use interface for users who have never worked with provenance tracking before. You can operate the tool using the

If you feel limited by PROVIT you should have a look at more extensive implementations, e.g.: prov.

Full documentation is available under: provit.readthedocs.io.

1.1 Quick Installation

Note: *provit* requires a working installation of Python 3.7, furthermore the use of a virtualenv is strongly encouraged. If you need help to set this up, please follow the Installation section in the documentation.

provit is availabe via the Python Package Index (PyPI) and can be installed by using pip pip. Simply create a virtualenvironment with your preferred method a run the *pip install* command:

```
$ mkvirtualenv provit
$ pip install provit
```

1.2 Quickstart

provit provides three modes of interaction:

- command line interface
- graphical user interface
- python package

All of them allow you to track provenance, but the *provit browser* additionally lets you explore tracked provenance.

1.2.1 provit browser

You can start provit browser directly from your terminal:

```
$ provit browser
```

1.2.2 provit cli

Simply *cd* to the directory, where your data is located, create (or append to an already existing) provenance file.

```
$ provit add FILEPATH [OPTIONS]
```

The –help command shows you the full list of available options and arguments.

```
$ provit --help
```

1.2.3 provit package

Using provit in your ETL pipeline is easy. simply import the Proveance class and start using it (e.g. as displayed below).

```
from provit import Provenance

# load prov data for a file, or create new prov for file
prov = Provenance(<filepath>)

# add provenance metadata
prov.add(agents=[ "agent" ], activity="activity", description="...")
prov.add_primary_source("primary_source")
prov.add_sources([ "filepath1", "filepath2" ])

# return provenance as json tree
prov_dict = prov.tree()

# save provenance metadata into "<filename>.prov" file
prov.save()
```

1.3 Roadmap

We have a small roadmap, which we will make transparent below:

- Increase test coverage (currently 81%)
- Windows support (all devs are on Linux)
- Agent management in PROVIT Browser

1.4 Overview

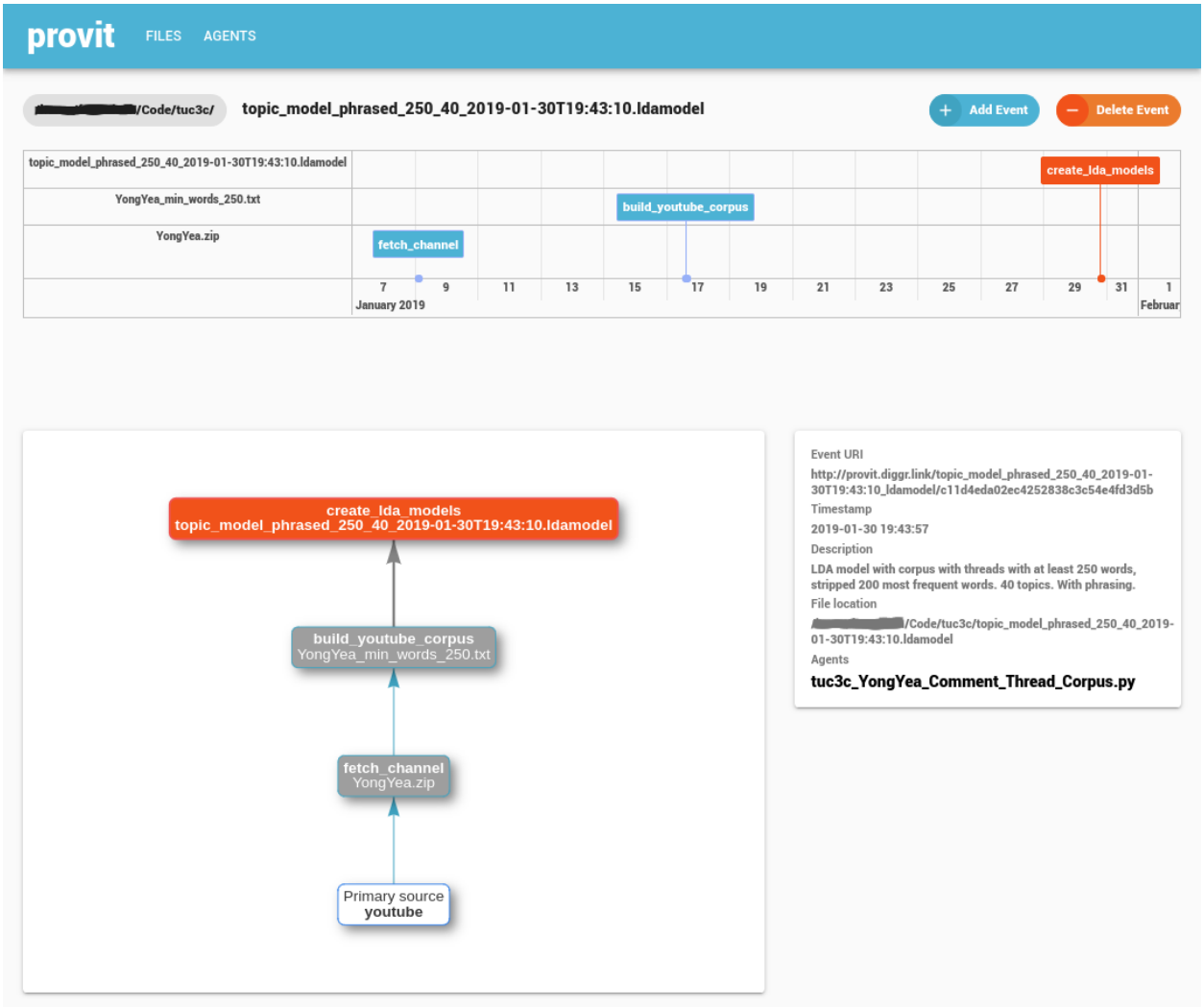
Authors P. Mühleder muehleder@ub.uni-leipzig.de, F. Rämisch raemisch@ub.uni-leipzig.de

CHAPTER 2

provit browser

provit browser is the graphical user interface of provit. It can be used to track provenance, as well as intuitive interface to explore existing provenance.

If a data file, is selected in the browser a detail view of the accompanied provenance file is opened. It consinsts of a timeline, a provenance graph, and a detail window, where additional information about the currently selected node in the provenance file is displayed.



2.1 Start

You start the *provit browser* by simply invoking this in your terminal.

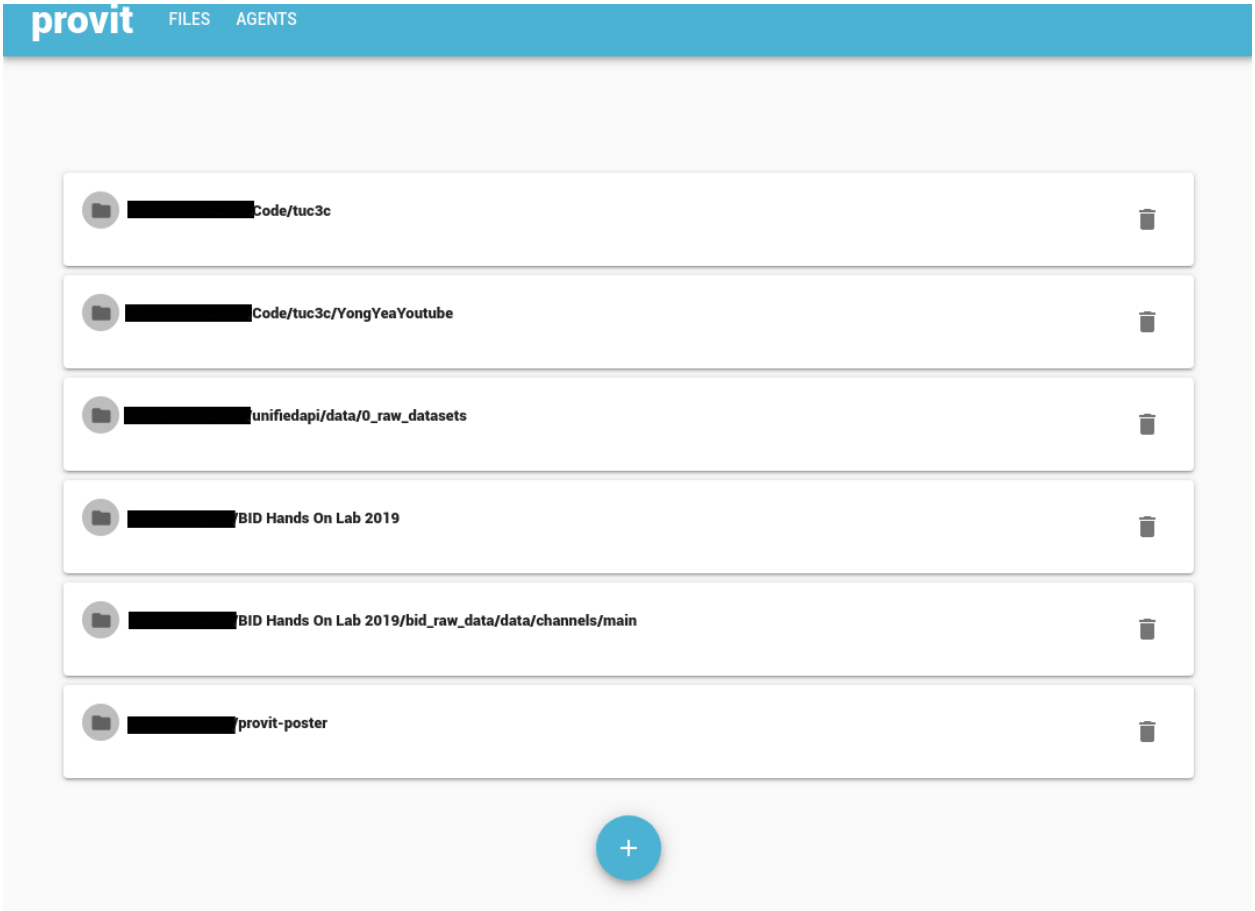
```
% ~ (provit) ➜  provit browser
 * Serving Flask app "provit.browser.backend" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5555/ (Press CTRL+C to quit)
```

The warning can safely be ignored, as the server will only be started on your machine.

2.2 Usage

After starting *provit browser* you are displayed all your data directories.

New directories can be added by using the plus sign at the bottom of the page.



By clicking on a directory, the contents of the directory are displayed. All files without provenance information have a red background. Inspectable/Explorable files are displayed with a white background.

Clicking on a file with disply the detailed provenance information of the file as shown above.

CHAPTER 3

provit command line interface

Provenance information can be inspected and created using the command line interface. A short demonstration of the capabilities is given below.

3.1 Show Provenance

To show the provenance information for a file *cd* into the directory and enter the show command.

```
$ provit show source_file
{
    "uri": "http://vocab.ub.uni-leipzig.de/provit/source_file/
↪0cab498f56e3417da1120dcaa6f48354",
    "agent": [
        "http://vocab.ub.uni-leipzig.de/provit/enrich_0.1.0"
    ],
    "activity": "http://vocab.ub.uni-leipzig.de/provit/data_download/
↪632c7e1fb1814cda86d7c727f1b1a4ed",
    "ended_at": "2019-11-27 15:01:48",
    "activity_desc": "The data was downloaded",
    "location": "/home/fraemisch/tmp/provit_demo/source_file",
    "primary_sources": [
        "http://vocab.ub.uni-leipzig.de/provit/https:/diggr.link"
    ],
    "sources": []
}
```

3.2 Create Provenance (interactively)

Provenance information can be appended to existing provenance information or created for files, which do not yet have provenance information attached.

The following command will take you to the interactive prompt which guides you through the provenance creation process.

```
$ provit create FILENAME
```

Note: Agents can not be created properly using this interface. Please see *Vocabulary* for information on the types of agents and how to create them.

3.3 Create Provenance (non-interactive)

Sometimes you want to create provenance by just giving the information as command line arguments. This is also possible, but should only be used by advanced users or in scripts.

```
provit add \
        --agent testagent \
        --comment test \
        --activity testing \
        FILENAME
```

This will create a *FILENAME.prov* file right next to your existing data file, or append the provenance information given as options to the provenance graph, in case the file already exists.

3.4 ~/.provit directory

provit stores information about agents and its configuration in the home directory. You can

Have a look at the *Vocabulary* section to see example agent files.

CHAPTER 4

Vocabulary

Provit uses a small subset of the W3C PROV-O vocabulary.

4.1 Agents

Provit implements three different agents:

- Organization
- Person
- SoftwareAgent

The names are specified in PROV-O.

4.1.1 Organization

An organizations is a social or legal institution that can be responsible for creation or modification of data (in our case).

A valid entry for *~/.provit/agents/wikidata.yaml* would be:

```
homepage:
- https://www.wikidata.org
name:
- Wikidata
slug: wikidata
type: Organization
```

4.1.2 Person

A persons is a human actor. If a person manually corrects a data set, it is the agent responsible for this modification step.

A valid entry for *~/.provit/agents/johndoe.yaml* would be:

```
email:
- john.doe@uni-leipzig.de
- doe.john@ub.uni-leipzig.de
homepage:
- https://ub.uni-leipzig.de
- https://diggr.link
institution:
- ubleipzig
name:
- "John Doe"
- "J. Doe"
- "John Dö"
slug: johndoe
type: Person
```

4.1.3 SoftwareAgent

A SoftwareAgent is running software (e.g. a scraper for data retrieval or bulk downloader).

A valid entry for *~/.provit/agents/gephi_0.9.2.yaml* would be:

```
homepage:
- https://gephi.org/
name:
- Gephi
slug: gephi_0.9.2
type: SoftwareAgent
version:
- 0.9.2
```

CHAPTER 5

provit

5.1 provit package

5.1.1 Subpackages

provit.browser package

Submodules

provit.browser.backend module

`provit.browser.backend.`**`create_app`**(*cfg=None*)

`provit.browser.backend.`**`start_backend`**(*debug=False*)

`provit.browser.backend.`**`start_provit_browser`**(*debug=False*)

Start provit backend and open webbrowser

`provit.browser.backend.`**`start_webbrowser`**()

Module contents

provit.tests package

Submodules

provit.tests.test_agent module

provit.tests.test_cli module

provit.tests.test_config module

provit.tests.test_config.**test_get_config**(*tmp_path*)

provit.tests.test_config.**test_load_provit_dir**(*tmp_path*)

provit.tests.test_config.**test_load_provit_dir_from_config**(*tmp_path*)

provit.tests.test_home module

provit.tests.test_prov module

provit.tests.test_utils module

Module contents

5.1.2 Submodules

5.1.3 provit.agent module

Everything agent profile related

1. Classes

- Person
- SoftwareAgent
- Organization

Each class contains :to_json(): and :graph(): functions, returning the data of the classes either as json-dict or rdflib Graph.

2. Helper functions

- load_agent_profile(slug) loads the yaml file of agent :slug: and initiates the respective agent class (depending on type)
- load_agent_profiles() loads all agent yaml files and returns a list of agent classes
- agent_factory(slug, **type_**) initializes an agent class of type :**type_**: with id/uri :slug:

class provit.agent.**OrganizationAgent**(*slug*, *name=[]*, *homepage=[]*, *uri=”*)

Bases: object

graph()

to_json()

update(*data*)

class provit.agent.**PersonAgent**(*slug*, *name=[]*, *institution=[]*, *homepage=[]*, *email=[]*, *uri=”*)

Bases: object

graph()

to_json()

update(*data*)

class provit.agent.**SoftwareAgent**(*slug*, *name=[]*, *version=[]*, *homepage=[]*, *uri=”*)

Bases: object

graph()

to_json()

update(*data*)

provit.agent.**agent_factory**(*slug*, *type_*)

return “empty” agent class instance of the specified type

provit.agent.**load_agent_profile**(*slug*)

loads agent yaml profile (if available) and initiates agent class with the values obtained from the yaml file

provit.agent.**load_agent_profiles**()

5.1.4 provit.cli module

5.1.5 provit.config module

provit configuration module

provides the Config-class as well as its factory method get_config.

By default, $HOME/.provit is assumed to be provits default config directory. This can be customized.

class provit.config.**Config**(*provit_dir: pathlib.Path*, *person: str = ’Person’*, *software: str = ’SoftwareAgent’*, *organization: str = ’Organization’*, *base_uri: str = ’http://vocab.ub.uni-leipzig.de/provit/{}’*)

Bases: object

agent_profile(*slug*)

agent_profile_exists(*slug*)

agents_dir

base_uri = 'http://vocab.ub.uni-leipzig.de/provit/{}'

directories_file

get_agent_profile(*slug*)

organization = 'Organization'

person = 'Person'

software = 'SoftwareAgent'

provit.config.**get_config**(*provit_dir=None*)

factory method for Config class. can be given a custom provit dir. If no directory is given, the default directory ~/.provit will be chosen.

5.1.6 provit.home module

Helper functions for accessing data in the provit home directory

provit.home.**add_directory**(*directory*, *directories_file=None*)

Add directory to project directories list

provit.home.**load_directories**(*directories_file=None*)

load the list of directories from the directories yaml file

`provit.home.remove_directories(directory)`
Remove directories from project directory list

5.1.7 provit.namespaces module

5.1.8 provit.prov module

Provenance class handles provenance metadata information.

Use:

from pit.prov import Provenance

#load prov data for a file, or create new prov for file prov = Provenance(<filepath>)

#add provenance metadata prov.add(agent="agent", activity="activity", description="...") prov.add_primary_source("primary_source", url="http://...", comment="...") prov.add_sources(["filepath1", "filepath2"])

#return provenance as json tree prov_dict = prov.tree()

#save provenance metadata into "<filename>.prov" file prov.save()

class `provit.prov.Provenance(filepath, namespace=None, overwrite=False)`
Bases: `object`

Provenance class handles the provenance metadata graph

add(*agents*, *activity*, *description*, *started_at=''*, *ended_at=''*)
Add new basic provenance information (agent, activity) to file

add_graph(*graph*)

add_primary_source(*primary_source*, *url=None*, *comment=None*)
Adds primary source (+ url and comment) to provenance information

add_sources(*filepaths*, *add_prov_to_source=True*)
Add provenance information from source file (wasDerivedFrom) to provenance graph If source file does not have valid provenance data, a prov graph for the source file is initialized

get_agents(*include_primary_sources=False*)
Returns agent profiles from prov graph

get_current_location()
Returs the file location of root element

get_primary_sources(*root_entity=None*)
Returns the URIs of all primary sources in prov graph

iter_remove(*root_uri*)

remove_last_event()
removes the last provenance event and all sources, if they do not belong to the same file

save()
Serializes prov graph as json-ld and saves it to file

tree()
Returns of dict tree with provenance information

`provit.prov.load_prov(filepath, namespace=Namespace('http://vocab.ub.uni-leipzig.de/provit/'))`
Loads a Provenance Object from the given file path or returns None if no (valid) provenance file was found. :param filepath: :return:

`provit.prov.load_prov_files(directory)`

5.1.9 provit.utils module

`provit.utils.load_jsonld(filepath)`
Reads json-ld file and returns (rdfslib) graph and context

`provit.utils.provit_uri(slug)`

`provit.utils.walk_up(start_dir)`
Walks up directory tree from :start_dir: and returns directory paths

5.1.10 Module contents

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

T

U

W