

---

# **Skosify Documentation**

*Release 2.0.1*

**Osma Suominen  
Jakob Voß**

**May 15, 2018**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Usage</b>	<b>5</b>
<b>3</b>	<b>Additional scripts</b>	<b>7</b>
<b>4</b>	<b>Author and Contributors</b>	<b>9</b>
<b>5</b>	<b>See also</b>	<b>11</b>
	<b>Python Module Index</b>	<b>17</b>



Python script for converting to **SKOS**.

This program accepts a thesaurus-like vocabulary expressed as RDFS, OWL or SKOS as input. It produces a clean SKOS representation, which attempts to represent the input data losslessly using SKOS best practices. When given SKOS as input, it will be cleaned up, validated and enriched to follow the SKOS specification and related best practices.



# CHAPTER 1

---

## Installation

---

Skosify requires Python 2.6 or newer.

```
pip install --upgrade skosify
```





As command line script:

```
skosify myvoc.owl -o myvoc-skos.ttl --label "My Ontology"
```

This will read the file `myvoc.owl` in RDF/XML format and write SKOS file `myvoc-skos.ttl` in Turtle format, setting the name of the Concept Scheme to `My Ontology`.

Run `skosify --help` for more usage information.

As Python library:

```
import skosify # contains skosify, config, and infer

voc = skosify.skosify('myontology.owl', label='My Ontology')
voc.serialize(destination='myontology-skos.rdf', format='xml')

rdf = Graph()
rdf.parse('myontology.owl')
config = skosify.config('owl2skos.cfg')
voc = skosify.skosify(rdf, **config)

skosify.infer.skos_related(rdf)
skosify.infer.skos_topConcept(rdf):
skosify.infer.skos_hierarchical(rdf, narrower=True)
skosify.infer.skos_transitive(rdf, narrower=True)

skosify.infer.rdfs_classes(rdf)
skosify.infer.rdfs_properties(rdf)
```

See the [API Reference](#) for documentation of the public API of this module. Everything not listed there might change in a future version.

Additional documentation can be found in the [GitHub project wiki](#)



## CHAPTER 3

---

### Additional scripts

---

The *scripts* directory contains two additional scripts to be used together with Skosify:

- *skosify.cgi* a web application to use Skosify
- *sparqldump.py* a command line client to download RDF via a SPARQL endpoint



## CHAPTER 4

---

### Author and Contributors

---

- Osma Suominen
- Jakob Voß
- Dan Michael O. Heggø
- Sini Pessala



---

See also

---

See *background* for history, related works, publications etc.

## 5.1 Background

Skosify has originally been developed in the FinnONTO projects at the [Semantic Computing Research Group \(SeCo\)](#) at Aalto University and University of Helsinki, Finland. The code is open source and available under a permissive MIT-style license. Current development is ongoing at the [National Library of Finland](#) and external contributors.

See also the [Skosify homepage](#) at SeCo.

### 5.1.1 Related works

- [mc2skos](#) can convert MARC21 Classification and Authority records to SKOS
- [SKOS Play!](#) can visualize SKOS and convert Excel spreadsheets to SKOS

### 5.1.2 Publications

- Osma Suominen and Christian Mader: Assessing and Improving the Quality of SKOS Vocabularies. *Journal on Data Semantics*, vol. 3, no. 1, pp. 47-73, June, 2014 ([PDF](#))
- Osma Suominen and Eero Hyvönen: Improving the Quality of SKOS Vocabularies with Skosify. *Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2012)*, Springer-Verlag, Galway, Ireland, October, 2012 ([PDF](#))

## 5.2 Skosify API Reference

```
skosify.skosify(*sources, **config)
```

Convert, extend, and check SKOS vocabulary.

`skosify.config` (*file=None*)

Get default configuration and optional settings from config file.

- **file**: can be a filename or a file object

Checks/fixes are bundled in one namespace.

`skosify.check.disjoint_relations` (*rdf, fix=False*)

Check if the graph contains concepts connected by both of the semantically disjoint semantic `skos:related` and `skos:broaderTransitive` (S27), and optionally remove the involved `skos:related` relations.

### Parameters

- **rdf** (*Graph*) – An `rdflib.graph.Graph` object.
- **fix** (*bool*) – Fix the problem by removing `skos:related` relations that overlap with `skos:broaderTransitive`.

`skosify.check.hierarchical_redundancy` (*rdf, fix=False*)

Check for and optionally remove extraneous `skos:broader` relations.

### Parameters

- **rdf** (*Graph*) – An `rdflib.graph.Graph` object.
- **fix** (*bool*) – Fix the problem by removing `skos:broader` relations between concepts that are otherwise connected by `skos:broaderTransitive`.

`skosify.check.hierarchy_cycles` (*rdf, fix=False*)

Check if the graph contains `skos:broader` cycles and optionally break these.

### Parameters

- **rdf** (*Graph*) – An `rdflib.graph.Graph` object.
- **fix** (*bool*) – Fix the problem by removing any `skos:broader` that overlaps with `skos:broaderTransitive`.

Inference rules are bundled in one namespace.

`skosify.infer.rdfs_classes` (*rdf*)

Perform RDFS subclass inference.

Mark all resources with a subclass type with the upper class.

`skosify.infer.rdfs_properties` (*rdf*)

Perform RDFS subproperty inference.

Add superproperties where subproperties have been used.

`skosify.infer.skos_hierarchical` (*rdf, narrower=True*)

Infer `skos:broader/skos:narrower` (S25) but only keep `skos:narrower` on request.

**Parameters** **narrower** (*bool*) – If set to `False`, `skos:narrower` will not be added, but rather removed.

`skosify.infer.skos_hierarchical_mappings` (*rdf, narrower=True*)

Infer `skos:broadMatch/skos:narrowMatch` (S43) and add the super-properties `skos:broader/skos:narrower` (S41).

**Parameters** **narrower** (*bool*) – If set to `False`, `skos:narrowMatch` will not be added, but rather removed.

`skosify.infer.skos_related` (*rdf*)

Make sure that `skos:related` is stated in both directions (S23).



`skosify.infer.skos_symmetric_mappings` (*rdf, related=True*)

Ensure that the symmetric mapping properties (`skos:relatedMatch`, `skos:closeMatch` and `skos:exactMatch`) are stated in both directions (S44).

**Parameters** `related` (*bool*) – Add the `skos:related` super-property for all `skos:relatedMatch` relations (S41).

`skosify.infer.skos_topConcept` (*rdf*)

Infer `skos:topConceptOf/skos:hasTopConcept` (S8) and `skos:inScheme` (S7).

`skosify.infer.skos_transitive` (*rdf, narrower=True*)

Perform transitive closure inference (S22, S24).

## 5.3 Skosify Release Notes

### 5.3.1 v2.0.1 - 2017-11-20

This release does not change functionality of the command line client (except support of options `output`, `to_format`, `log`, and `debug` in config files). It comes with a major refactoring of internal source code and add unit tests.

- Make parts of the functionality available as module
- Drop internal object `Skosify`
- Drop support of `rdflib < 3.0.0`
- Separate command line client and core library
- Add unit tests and documentation
- Implemented SPARQL hooks for input data (#46)
- Add `--set-modified` option to set `dct:modified` timestamp on `ConceptScheme` (#42)
- Add option to eliminate hierarchical redundancy (#40)
- Generated `ConceptScheme` URI now defaults to namespace, not `ns:conceptscheme`

### 5.3.2 v1.0 - 2014-01-14

See also the [release announcement](#).

- Support both Python 2.x and 3.x with the same code (#35)
- PEP 8 coding style conformance (#39)
- Enable mapping of relations to inverse properties (#32)
- Enable mapping of types, literals and relations to multiple values (#34)
- Remove bundled `SetStore` in favor of `rdflib 4.x`, where it's included (#33)
- Add `enrich_mappings` option, making mapping inferences optional (#29)
- Add `mark_top_concepts` option, making top concept marking optional (#31)
- Fixes for conversion of FinnONTO ontologies (e.g. #30)
- More deterministic RDF and logging output

### 5.3.3 v0.6 - 2013-01-29

- Release with mostly minor improvements to the code to improve robustness and correct more problems found in SKOS vocabularies in the wild.
- Deterministic cycle breaking, also for top level cycles (#26)
- Correct SKOS S14 (no >1 prefLabels per language) for all resources (#24)
- Don't break on literal inScheme values (#25)
- Correct missing language tag and extra whitespace for rdfs:label (#27)
- Better support for multiple concept schemes (#3)
- SKOS inferences for mapping relations (#28)
- SKOS inferences for hasTopConcept/topConceptOf and inScheme (#23)
- Make some cleanups optional and off by default (#22)
- Easier to read -help output with grouped options (r140)
- Added sparqldump utility script (r137, r138)

### 5.3.4 v0.5 - 2012-05-04

- Online version (#16)
- Optionally correct unlabeled concept schemes (#13)
- Detect referred concept schemes and add rdf:type (#14)
- Clean up transitive/narrower relationships (#15)
- Make cycle removal optional (#17)
- Make cleanup of related relationships optional (#18)
- Support different prefLabel policies (#20)
- Switch meaning of -i/-I options (r106)
- Use standard Python logging module (r97)
- Support -log option to specify a log file (r107)
- Catch parsing errors properly (r108)

### 5.3.5 v0.4

- Intermediate version. Not really released.

### 5.3.6 v0.3 - 2012-02-16

New features (with issue numbers):

- Automatically detect vocabulary namespace (#8).
- Setting language for labels and documentary notes without language tag (#10).
- Better sanitizing of Concept-specific properties on Collections (#11)

- Support multiple input files (issue #12)
- Support .nt file extension (n-triples format)
- Slightly better usage message for `-help` (show input file parameters)

### 5.3.7 v0.2 - 2011-11-17

\* Added support for property `skosext:candidateLabel`. Label properties mapped to this will be converted to `skos:prefLabel` if one doesn't exist, but `skos:altLabel` if a `prefLabel` already exists (for the same concept, in the same language). This makes it possible to prefer some kind of labels, or labels from a particular source, when there may be many sources of labels. This is useful for combined ontologies such as TERO and LIITO, where the "same" concept may have different `prefLabels` from the different source vocabularies.

### 5.3.8 v0.1 - 2011-05-25

- First public version.



**S**

skosify, [11](#)  
skosify.check, [12](#)  
skosify.infer, [12](#)



## C

`config()` (in module `skosify`), 12

## D

`disjoint_relations()` (in module `skosify.check`), 12

## H

`hierarchical_redundancy()` (in module `skosify.check`), 12

`hierarchy_cycles()` (in module `skosify.check`), 12

## R

`rdfs_classes()` (in module `skosify.infer`), 12

`rdfs_properties()` (in module `skosify.infer`), 12

## S

`skos_hierarchical()` (in module `skosify.infer`), 12

`skos_hierarchical_mappings()` (in module `skosify.infer`),  
12

`skos_related()` (in module `skosify.infer`), 12

`skos_symmetric_mappings()` (in module `skosify.infer`), 12

`skos_topConcept()` (in module `skosify.infer`), 13

`skos_transitive()` (in module `skosify.infer`), 13

`skosify` (module), 11

`skosify()` (in module `skosify`), 11

`skosify.check` (module), 12

`skosify.infer` (module), 12