wikidata-taxonomy Documentation

Release 0.6.7

Jakob Voß

Jun 19, 2019

Contents

1 Installation **3**

2 Usage **5**
2.1 Examples 6

3 Options **9**
3.1 Query options 9
3.2 Output options 11

4 Output formats **13**
4.1 Text format 13
4.2 JSON format 14

5 NDJSON format **17**
5.1 CSV and TSV format 17

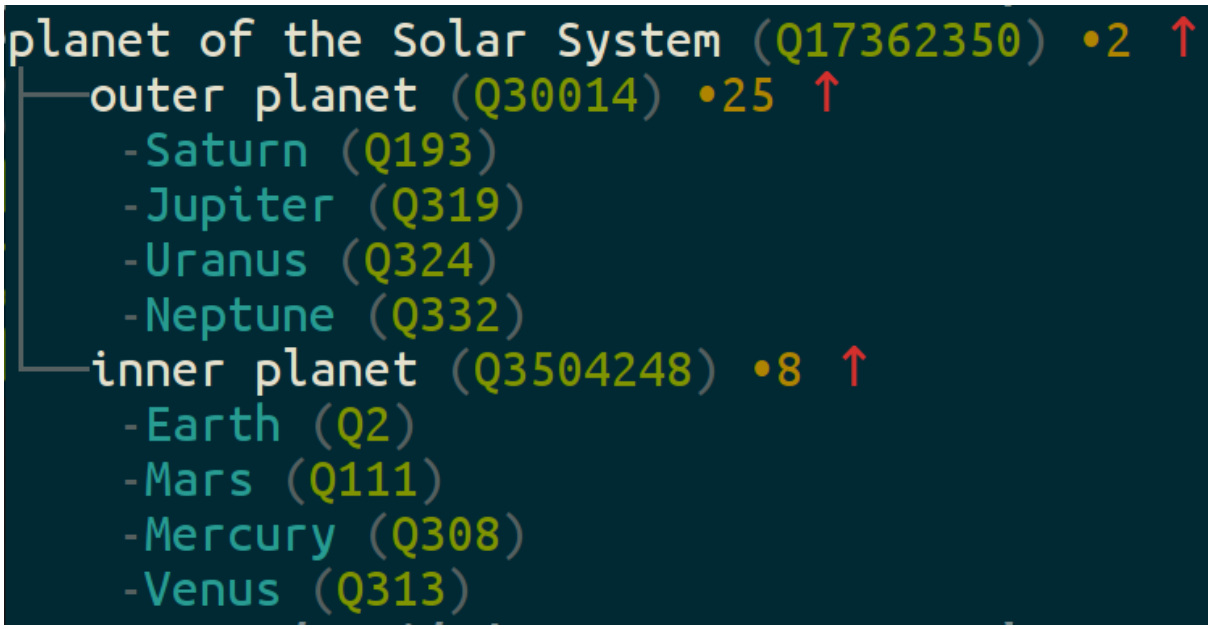
6 Usage as module **19**

7 Usage in web applications **21**

8 Release notes **23**

9 See Also **25**

Command-line tool and library to extract taxonomies from Wikidata.



CHAPTER 1

Installation

wikidata-taxonomy requires at least NodeJs version 6.

Install globally to make command `wdtaxonomy` accessible from your shell `$PATH`:

```
$ npm install -g wikidata-taxonomy
```

Installation and usage *as module* and *in web applications* is described below.

CHAPTER 2

Usage

This module provides the command `wdtaxonomy`. By default, a usage help is printed:

```
$ wdtaxonomy


  Usage: wdtaxonomy [options] <id>

  extract taxonomies from Wikidata


  Options:

    -V, --version                           output the version number
    -b, --brief                             omit counting instances and sites
    -c, --children                          get direct subclasses only
    -C, --color                             enforce color output
    -d, --descr                             include item descriptions
    -e, --sparql-endpoint <url>             customize the SPARQL endpoint
    -f, --format <txt|csv|tsv|json|ndjson>  output format
    -i, --instances                         include instances
    -I, --no-instancecount                  omit counting instances
    -j, --json                              use JSON output format
    -l, --lang <lang>                       specify the language to use
    -L, --no-labels                         omit all labels
    -m, --mappings <ids>                    mapping properties (e.g. P1709)
    -n, --no-colors                         disable color output
    -o, --output <file>                     write result to a file
    -P, --property <id>                     hierarchy property (e.g. P279)
    -R, --prune <criteria>                  prune hierarchies (e.g. mappings)
    -p, --post                              use HTTP POST to disable caching
    -r, --reverse                           get superclasses instead
    -s, --sparql                            print SPARQL query and exit
    -S, --no-sitecount                      omit counting sites
    -t, --total                             count total number of instances
    -u, --user <name>                       user to the SPARQL endpoint
    -U, --uris                              show full URIs in output formats
    -v, --verbose                           make the output more verbose
    -w, --password <string>                 password to the SPARQL endpoint
    -h, --help                              output usage information
```

The first arguments needs to be a Wikidata identifier to be used as root of the taxonomy. For instance extract a taxonomy of planets (Q634):

```
$ wdtaxonomy Q634
```

To look up by label, use wikidata-cli (e.g `wd id planet` or `wd f planet`).

The extracted taxonomy by default is based on statements using the property "subclass of" (P279) or "subproperty of" (P1647). Taxonomy extraction and output can be controlled by several *options*. Option `--sparql` (or `-s`) prints the underlying SPARQL queries instead of executing them.

2.1 Examples

Direct subclasses of planet (Q634) with description and mappings:

```
$ wdtaxonomy Q634 -c -d -m =
```

The hierarchy properties P279 ("subclass of") and P31 ("instance of") to build taxonomies from can be changed with option `property` (`-P`).

Members of (P463) the European Union (Q458):

```
$ wdtaxonomy Q458 -P P463
```

Members of (P463) the European Union (Q458) and number of its citizens in Wikidata (P27):

```
$ wdtaxonomy Q458 -P 463/27
```

Wikiversity (Q370) editions mapped to their homepage URL (P856):

```
$ wdtaxonomy Q370 -i -m P856
```

Biological taxonomy of mammals (Q7377):

```
$ wdtaxonomy Q7377 -P P171 --brief
```

Property constraints (Q21502402) with number of properties that have each constraint:

```
$ wdtaxonomy Q21502402 -P 279,2302
```

As Wikidata is no strict ontology, subproperties are not factored in. For instance this query does not include members of the European Union although P463 is a subproperty of P361.

Parts of (P361) the European Union (Q458):

```
$ wdtaxonomy Q458 -P P361
```

A taxonomy of subproperties can be queried like taxonomies of items. The hierarchy property is set to P1647 ("subproperty of") by default:

```
$ wdtaxonomy P361
$ wdtaxonomy P361 -P P1647  # equivalent
```

Subproperties of "part of" (P361) and which of them have an inverse property (P1696):

```
$ wdtaxonomy P361 -P P1647/P1696
```

Inverse properties are neither factored in so queries like these do not necessarily return the same results:

What hand (Q33767) is part of (P361):

```
$ wdtaxonomy Q33767 -P 361 -r
```

What parts the hand (Q33767) has (P527):

```
$ wdtaxonomy Q33767 -P 527
```

CHAPTER 3

Options

3.1 Query options

3.1.1 brief (`-b`)

Don't count instance and sites. Same as `-S/--no-sitecount` and `-I/--no-instancecount`.

3.1.2 children (`-c`)

Get direct subclasses only

3.1.3 descr (`-d`)

Include item descriptions

3.1.4 sparql-endpoint (`-e`)

SPARQL endpoint to query (default: https://query.wikidata.org/sparql)

3.1.5 instances (`-i`)

Include instances

3.1.6 no-instancecount (`-I`)

Don't count number of instances

3.1.7 lang (`-l`)

Language to get labels in (default: `en`)

3.1.8 no-labels (`-L`)

Omit all labels. This allows for querying larger taxonomies (several thousands of classes), especially if combined with option `--brief`.

3.1.9 mappings (`-m`)

Lookup mappings based on given comma-separated properties such as P1709 (equivalent class). The following keywords can be used as shortcuts:

- `equal` or =: equivalent property (P1628), equivalent class (P1709), and exact match (P2888)
- `broader`: external superproperty (P2235)
- `narrower`: narrower external class (P3950), external subproperty (P2236)
- `class`: properties for mapping classes
- `property`: properties for mapping properties
- `all` all properties for ontology mapping (instances of Q30249126)

3.1.10 reverse (`-r`)

Get superclasses instead of subclasses up to the root

3.1.11 no-sitecount (`-I`)

Don't count number of sites

3.1.12 total (`-t`)

Count total (transitive) number of instances, including instances of subclasses

3.1.13 post (`-p`)

Use HTTP POST to disable caching

3.1.14 sparql (`-s`)

Don't actually perform a query but print SPARQL query and exit

3.1.15 user (`-u`)

User to the SPARQL endpoint

3.1.16 password (`-w`)

Password to the SPARQL endpoint

3.2 Output options

3.2.1 color (`-C`)

enable color output if it's disabled (e.g. when output is piped or written to a file)

3.2.2 format (`-f`)

Output format

3.2.3 json (`-j`)

Use JSON output format. Same as `--format json` but shorter.

3.2.4 no-colors (`-n`)

disable color output

3.2.5 output (`-o`)

write result to a file given by name

3.2.6 prune (`-R`)

prune hierarchy to all entries with any of a given criteria plus their broader concepts and all top concepts:

- `mappings`: has mappings
- `sites`: has sites
- `instances`: has instances
- `occurences` : has sites or instances

Multiple criteria can be combined as alternatives with comma.

3.2.7 uris (`-U`)

Show full URIs in output formats, e.g. http://www.wikidata.org/entity/Q1 instead of `Q1`

3.2.8 verbose (`-v`)

Show verbose error messages

CHAPTER 4

Output formats

4.1 Text format

By default, the taxonomy is printed in "`text`" format with colored Unicode characters:

```
$ wdtaxonomy Q17362350
```

```
planet of the Solar System (Q17362350) •2 ↑
├──outer planet (Q30014) •25 ×4 ↑↑
└──inner planets (Q3504248) •8 ×4 ↑↑
```

The output contains item labels, Wikidata identifiers, the number of Wikimedia sites connected to each item (indicated by bullet character "•"), the number of instances (property P31), indicated by a multiplication sign "×"), and an upwards arrow ("↑") as indicator for additional superclasses.

Option "`--instances`" (or "`-i`") explicitly includes instances:

```
$ wdtaxonomy -i Q17362350
```

```
planet of the Solar System (Q17362350) •2 ↑
├──outer planet (Q30014) •25 ↑↑
|   -Saturn (Q193)
|   -Jupiter (Q319)
|   -Uranus (Q324)
|   -Neptune (Q332)
└──inner planets (Q3504248) •8 ↑↑
    -Earth (Q2)
    -Mars (Q111)
    -Mercury (Q308)
    -Venus (Q313)
```

Classes that occur at multiple places in the taxonomy (multihierarchy) are marked like in the following example:

```
$ wdtaxonomy Q634
```

```
planet (Q634) •202 ×7 ↑
├──extrasolar planet (Q44559) •88 ×833 ↑
```

(continues on next page)

(continued from previous page)

```
|   ├──circumbinary planet (Q205901) •15 ×10
|   ├──super-Earth (Q327757) •32 ×46 ↑
...
├──terrestrial planet (Q128207) •70 ×7
|  super-Earth (Q327757) •32 ×46 ↑ ...
...
```

4.2 JSON format

Option `--format json` serializes the taxonomy as JSON object. The format follows specification of JSKOS Concept Schemes:

```
{
  "type": [ "http://www.w3.org/2004/02/skos/core#ConceptScheme" ],
  "modified": "2017-11-06T10:25:54.966Z",
  "license": [
    {
      "uri": "http://creativecommons.org/publicdomain/zero/1.0/",
      "notation": [ "CC0" ]
    }
  ],
  "languages": [ "en" ],
  "topConcepts": [
    { "uri": "http://www.wikidata.org/entity/Q17362350" }
  ],
  "concepts": [ ]
}
```

Field `concepts` contains an array of all extracted Wikidata entities (usually classes and instances) as JSKOS Concepts:

```
{
  "uri": "http://www.wikidata.org/entity/Q17362350",
  "notation": [ "Q17362350" ],
  "prefLabel": {
    "en": "planet of the Solar System"
  },
  "scopeNote": {
    "en": [ "inner and outer planets of our solar system" ]
  },
  "broader": [
    { "uri": "http://www.wikidata.org/entity/Q634" }
  ],
  "narrower": [
    { "uri": "http://www.wikidata.org/entity/Q30014" },
    { "uri": "http://www.wikidata.org/entity/Q3504248" }
  ]
}
```

Instances (option `--instances`) are linked via field `subjectOf` the same way as field `broader` and `narrower`.

The number of instances and sites, if counted is given as array of JSKOS Concept Occurrences in field `occurrences`, each identified by subfield `relation`:

```
{
  "uri": "http://www.wikidata.org/entity/Q30014",
  "notation": [ "Q30014" ],
```

(continues on next page)

(continued from previous page)

```
  "prefLabel": {
    "en": "outer planet of the Solar system"
  },
  "occurrences": [
    {
      "relation": "http://www.wikidata.org/entity/P31",
      "count": 4
    },
    {
      "relation": "http://schema.org/about",
      "count": 25
    }
  ]
}
```

Mappings (option `--mappings`) are stored in field `mappings` as array of JSKOS Concept Mappings:

```
[
  {
    "from": {
      "memberSet": [
        { "uri": "http://www.wikidata.org/entity/Q634" }
      ]
    },
    "to": {
      "memberSet": [
        { "uri": "http://dbpedia.org/ontology/Planet" }
      ]
    },
    "type": [
      "http://www.w3.org/2004/02/skos/core#exactMatch",
      "http://www.w3.org/2002/07/owl#equivalentClass",
      "http://www.wikidata.org/entity/P1709"
    ]
  }
]
```

The mapping type is given in field `type` with the Wikidata property URI as last array element and the SKOS mapping relation URI as first.

CHAPTER 5

NDJSON format

Option `--format ndjson` serializes JSON field `concepts` with one record per line. The order if records is same as in txt, json, and csv format but each concept is only included once.

5.1 CSV and TSV format

CSV and TSV format are optimized for comparing differences in time. Each output row consists of five fields:

- **level** in the hierarchy indicated by zero or more "–" (default) or "=" characters (multihierarchy).
- **id** of the item. Items on the same level are sorted by their id.
- **label** of the item. Language can be selected with option `--language`. The label in csv format is quoted.
- **sites**: number of connected sites (Wikipedia and related project editions). Larger numbers may indicate more established concepts.
- **parents** outside of the hierarchy, indicated by zero or more "^" characters.

For instance the CSV output for Q634 would be like this:

```
$ wdtaxonomy -f csv Q634
```

```
level,id,label,sites,instances,parents
,Q634,"planet",196,7,^
-,Q44559,"extrasolar planet",81,833,^
--,Q205901,"circumbinary planet",14,10,
--,Q327757,"super-Earth",32,46,
...
-,Q128207,"terrestrial planet",67,7,
==,Q327757,"super-Earth",32,46,
...
```

In this example there are 196 Wikipedia editions or other sites with an article about planets and seven Wikidata items are direct instance of a planet. At the end of the line "^" indicates that "planet" has one superclass. In the next rows "extrasolar planet" (Q44559) is a subclass of planet with another superclass indicated by "^". Both "circumbinary planet" and "super-Earth" are subclasses of "extrasolar planet". The latter also occurs as subclass of "terrestrial planet" where it is marked by "==" instead of "--".

CHAPTER 6

Usage as module

Add `wikidata-taxonomy` as dependency to you `package.json`:

```
$ npm install wikidata-taxonomy --save
```

The library provides:

- `queryTaxonomy(id, options)` returns a promise with a taxonomy extracted from Wikidata as JSKOS Concept Scheme. See *JSON format* of the command line client for documentation.

  ```
  const { queryTaxonomy } = require('wikidata-taxonomy')

  var options = { lang: 'fr', brief: true }
  queryTaxonomy('Q634', lang)
  .then(taxonomy => {
    taxonomy.concepts.forEach(concept => {
      var qid = concept.notation[0]
      var label = (concept.prefLabel || {}).fr || '???'
      console.log('%s %s', qid, label)
    })
  })
  .catch(error => console.error("E",error))
  ```

  Options roughly equivalent command line *query options*:

  - boolean flags `brief`, `children`, `description`, `labels`, `total`, `instances`, `instancecount`, `sitecount`, `reverse`, `post`
  - SPARQL endpoint configuration with `endpoint`, `user`, `password`
  - language tag `language` or `lang`
  - array `property` (set to `['P279', 'P31']` by default)
  - array or string `mappings`

- `serializeTaxonomy` contains serializers to be called with a taxonomy, an output stream, and optional configuration:

  ```
  const { serializeTaxonomy } = require('wikidata-taxonomy')

  // serialize taxonomy to stream
  ```

(continues on next page)

(continued from previous page)

```
serializeTaxonomy.csv(taxonomy, process.stdout)
serializeTaxonomy.txt(taxonomy, process.stdout, {colors: true}) // FIXME
serializeTaxonomy.json(taxonomy, process.stdout)
serializeTaxonomy.ndjson(taxonomy, process.stdout)
```

CHAPTER 7

Usage in web applications

Experimental support of this library in web application is given with file `wikidata-taxonomy.js` in directoy `dist`. The `gh-pages` branch contains a sample application, also available at http://jakobvoss.de/wikidata-taxonomy/.

Requires wikidata-sdk and a HTTP client library. The latter can be attached to `window.requestPromise` (before wikidata-taxonomy is loaded). Axios is detected by default.

```
<html>
  <head>
    <script src="https://unpkg.com/wikidata-sdk/dist/wikidata-sdk.min.js"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <script src="https://unpkg.com/wikidata-taxonomy/dist/wikidata-taxonomy.min.js
↪"></script>
  </head>
  <body>
    ...
  </body>
</html>
```

CHAPTER 8

Release notes

Release notes are listed in file CHANGES.md in the source code repository.

CHAPTER 9

See Also

build status npm version Documentation Status

This document

- at GitHub
- at npmjs
- at Read the Docs

Related tools

- wikidata-sdk is used by this module
- wikidata-cli provide more generic command line tools for Wikidata
- taxonomy browser is a web application based on Wikidata dumps
- Wikidata Graph Builder another web application
- wdmapper Wikidata authority file mapping tool
- Wikidata ontology explorer

Publications

- Jakob Voß (2016): *Classification of Knowledge Organization Systems with Wikidata*. CEUR Workshop Proceedings 1676. http://ceur-ws.org/Vol-1676/paper2.pdf (=Q27261879)