wdmapper Documentation

Release 0.0.17

Jakob Voß

Sep 02, 2018

Contents

wdmapper is a tool to manage mappings between authority files in Wikidata. See https://github.com/gbv/wdmapper#readme for source code and a brief summary.

CHAPTER 1

Installation

wdmapper requires Python 2.7 or higher. Releases can be installed from the command line with pip:

```
$ pip install wdmapper              # either install global
$ pip install wdmapper --user       # or install to ~/.local
```

Add option *–user* to install as normal user to *~/.local* and option *–upgrade* to update an already installed version.

The latest developer version can be retrieved from the git repository:

```
$ git clone https://github.com/gbv/wdmapper.git
$ cd wdmapper
$ git checkout dev
$ pip install -r requirements.txt
$ ./wdmapper.py
```

CHAPTER 2

Usage

wdmapper can be used both, as *command line application* and as *Python library*. This document focuses on the command line client but its functionality can also used in pure Python code.

2.1 Get mappings and property information

The following call looks up three *mappings* from Wikidata to World Heritage Site identifiers as assigned by UNESCO:

```
$ wdmapper get P757 --limit 3
```

Wikidata *properties* can also be referenced by label, URI or URL. The call could also be written like this:

```
$ wdmapper get 'World Heritage Site ID' --limit 3
$ wdmapper get 'http://whc.unesco.org/en/list/' --limit 3
$ wdmapper get 'https://www.wikidata.org/wiki/Property:P757' --limit 3
```

To get the mappings in stable order, add option *sort*. The default output format BEACON includes mapping metadata before the actual mappings:

```
$ wdmapper get P757 --limit 3 --sort
#FORMAT: BEACON
#NAME: World Heritage Site ID
#DESCRIPTION: Mapping from Wikidata IDs to World Heritage Site IDs
#PREFIX: http://www.wikidata.org/entity/
#TARGET: http://whc.unesco.org/en/list/
#SOURCESET: http://www.wikidata.org/entity/Q2013
#TARGETSET: http://www.wikidata.org/entity/Q19832918

Q319841|Luxor Temple|087-002
Q38095|Galápagos Islands|1
Q6153869|Lower Valley of the Awash|10
```

Command *head* only emits metadata, for instance to quickly look up a Wikidata property. This command is assumed as default if additional arguments are given so the following calls are equivalent:

```
$ wdmapper head P757
$ wdmapper P757
```

Output formats such as CSV emit mappings without metadata:

```
$ wdmapper get P757 --limit 3 --sort --to csv
source, target, annotation
Q319841, 087-002, Luxor Temple
Q38095, 1, Galápagos Islands
Q6153869, 10, Lower Valley of the Awash
```

The second output line tells that Wikidata item with id `Q319841` (source column) is linked to World Heritage Site with id `087-002` (target column). The third column (annotation, put in the middle in BEACON format) gives the item label for better readability.

In NTriples output format, identifiers are expanded to full URIs. The expansion is based on URI templates of each property (see metadata fields PREFIX and TARGET above):

```
$ wdmapper get P757 --limit 3 --sort --to nt --relation owl:sameAs
<http://www.wikidata.org/entity/Q319841> <http://www.w3.org/2002/07/owl#sameAs>
↪<http://whc.unesco.org/en/list/087-002> .
<http://www.wikidata.org/entity/Q38095> <http://www.w3.org/2002/07/owl#sameAs> <http:/
↪/whc.unesco.org/en/list/1> .
<http://www.wikidata.org/entity/Q6153869> <http://www.w3.org/2002/07/owl#sameAs>
↪<http://whc.unesco.org/en/list/10> .
```

The last example shows how to connect multiple authority files. If two properties are given, wdmapper retrieves mappings from the first (source) to the second (target). The following call lists all Wikidata items that have both a TED speaker ID, and a Find a Grave grave ID):

```
$ wdmapper get P2611 P535
#FORMAT: BEACON
#NAME: Find a Grave grave ID
#DESCRIPTION: Mapping from TED speaker IDs to Find a Grave grave IDs
#PREFIX: https://www.ted.com/speakers/
#TARGET: http://www.findagrave.com/cgi-bin/fg.cgi?page=gr&GRid=
#TARGETSET: http://www.wikidata.org/entity/Q63056

viktor_e_frankl|Q154723|14540087
jimmy_carter|Q23685|6734
john_wooden|Q551032|53261713
douglas_adams|Q42|22814
roger_ebert|Q212173|107806860
denis_dutton|Q1187362|63438326
```

In the case of such "indirect links", the annotation field is used to give the Wikidata item identifier.

2.2 Check mappings and identifiers

Command *check* and command *diff* both compare mappings and/or identifiers provided from input file and mappings in Wikidata.

For instance check whether Q42 still has the Find a Grave ID 22814:

```
$ echo Q42,22814 | wdmapper --no-header check P535
~ Q42|Douglas Adams|22814
```

CHAPTER 3

Mappings

This introduction needs to be expanded to better explain authority files!

wdmapper is a tool to manage mappings between authority files. What does this mean?

The current version of wdmapper is limited to simple 1-to-1 mappings, also referred to as **Links**.

Each link consists of

- a source URI, specified in abbreviated form as source ID
- a target URI, specified in abbreviated form as target ID

The type of link ("relation") can optionally be configured with option *relation*.

Two kinds of links are supported:

3.1 direct links

Direct links are mappings from a Wikidata item to an entity from another authority file. The external authority file is identified by its target *property*. The entity within in authority file is identified by an external identifier.

3.2 indirect links

indirect links from an external source identifier, given by a source *property*, to external target identifier, given by a target *property*, The link is possible through a common Wikidata item that uses both source property and target property.

CHAPTER 4

Properties

Each Wikidata property has a unique identifier build of "P" followed by a natural number. For instance "P40" denotes the property "child" that is used to connect Wikidata items about parents with items about their childs. wdmapper requires properties to have an URL template and to be of datatype external identifier: this applies for instance to "P214" but not to "P40".

Properties in wdmapper can be referred to in different ways. The following examples execute the *command line client* with default command *head*:

- property by identifier

  ```
  $ wdmapper P214
  ```

- property by URI or URL

  ```
  $ wdmapper http://www.wikidata.org/entity/P214
  $ wdmapper https://www.wikidata.org/wiki/Property:P214
  ```

- property by URL template. The placeholder `$1` can be omitted at the end of an URL

  ```
  $ wdmapper 'https://viaf.org/viaf/$1'
  $ wdmapper https://viaf.org/viaf/
  ```

- property by label (in any language)

  ```
  $ wdmapper 'VIAF ID'
  ```

Properties can be specified as *arguments* and as part of mapping metadata in BEACON input format.

Each *mapping* has

- a target property for *direct links* from Wikidata to another authority file or
- a source property and a target property for *indirect links* between two authority files.

CHAPTER 5

Command Line

wdmapper comes with a command line client of same name. The general calling syntax is

```
$ wdmapper [OPTIONS] COMMAND TARGET           # for direct links
$ wdmapper [OPTIONS] COMMAND SOURCE TARGET    # for indirect links
```

where `COMMAND` is one of the wdmapper *commands* and `SOURCE` and `TARGET` are Wikidata *properties*. A list of commands and *options* is shown with option `-h` or `--help`:

```
$ wdmapper --help
```

CHAPTER 6

Commands

wdmapper provides several commands to perform different tasks.

6.1 get

Get mappings from Wikidata based on given *properties*. Examples:

```
$ wdmapper get P214 --limit 10
$ wdmapper get P214 P2428 --limit 10
```

Output format can be controlled with option `to` having BEACON format as default. Number and order of results can be influenced by options `limit` and `sort`. See Wikidata BEACON generator for a similar online tool.

6.2 head

Get information about given properties. This command works similar to command *get* but no mappings are retrieved. This is the default command if properties are specified as additional command line arguments. Examples:

```
$ wdmapper head P214
$ wdmapper P214
$ wdmapper "VIAF ID"
$ wdmapper https://viaf.org/viaf/
```

6.3 check

Check whether all input mappings are also in Wikidata.

```
$ wdmapper get P214 P2428 --sort --limit 10 --to csv > mappings.csv
$ # ...wait until Wikidata could have been modified...
$ wdmapper check P214 P2428 < mappings.csv
```

Each output line is preceded by a marker to indicate which input mappings have been found in Wikidata and how mappings in Wikidata differ from input mappings:

- If the same link was found in Wikidata, it is preceded by "="
- If a same link was found in Wikidata but with different annotation (different item or item label), it is preceded by "~"
- If the link was not found in Wikidata is is preceded by "+". Following link lines starting with "-" indicate that other links would have to be removed or merged to add the missing link to Wikidata.

Use command *diff* instead to compare full sets of mappings.

Use cases:

- Detect mapping changes in Wikidata
- Check whether mappings can be added to Wikidata
- Lookup mappings for given identifiers

Example:

```
echo ,114 | ./wdmapper.py --no-header check P757
```

6.4 diff

Compare input mappings and mappings at Wikidata. This can be used for instance to regularly check whether mappings at Wikidata have been changed:

```
$ wdmapper get P214 P2428 -t csv > mappings.csv
$ # ...wait until Wikidata could have been modified...
$ wdmapper diff P214 P2428 -t csv < mappings.csv
```

Each output line is preceded by "+ " if an input link is missing in Wikidata or "- " if a link from Wikidata is missing in the input.

The output is sorted by links. Option "limit" implies option "sort" to get stable results. Use command *check* instead to compare a limited set of mappings against mappings on Wikidata.

6.5 convert

Read input mappings to check or translate between mapping formats. This is the default command if no properties have been specified as command line arguments. Examples:

```
$ wdmapper convert -i mappings.csv -t beacon
$ wdmapper convert -f csv -t beacon < mappings.csv > mappings.txt
```

6.6 add

Add input mappings to mappings at Wikidata unless already there. Better first try command *check* and/or command “add” with option “dry” to find out what statements would be added to Wikidata.

not implemented yet

6.7 sync

Align Wikidata mappings and input mappings by adding and removing mappings in Wikidata: missing mappings are created and additional mappings are removed.

not implemented yet

CHAPTER 7

Options

wdmapper can be controlled by several parameters. Run the command line client with option `--help` to get a full list of command line arguments.

7.1 Input and output

Option **input** (`--input` or `-i`) and option **output** (`--output` or `-o`) can be used to select an input or output **file**. The special value `-` is used as default to denote the standard input or standard output, respectively. Input is always assumed to be Unicode in UTF-8 encoding.

Option **from** (`--from` or `-f`) and option **to** (`--to` or `-t`) can be used to select input or output **format**. Default input format is `csv` and default output format is `beacon`. If no input/output format has been specified, it is guessed from input/output filename extension, for instance `.csv` for CSV format and `.txt` for BEACON format.

7.1.1 Supported formats

input formats: csv

output formats: csv, beacon, nt

7.1.2 Examples

```
$ wdmapper convert -i mappings.csv -o mappings.txt
$ wdmapper convert < mappings.csv -t beacon > mappings.txt
```

7.2 Mapping retrieval

7.2.1 limit

Limit maximum number of mappings to process.

7.2.2 sort

Sort mappings (alphabetically) for stable output. This can slow down the query so only use if needed. set to `False` by default.

7.2.3 language

Specify language of labels. English ("en") is used by default.

7.2.4 type

Filter Wikidata items to instances of some class or its subclasses. For instance the value Q5 (human) will only include mappings with Wikidata items about humans. Keep in mind that not all Wikidata items have proper instance statements and the class hierarchy often contains errors and unexpected subclasses! See wdtaxonomy for another command line tool to examine the Wikidata class hierarchy.

This option is ignored for command "check"!

7.2.5 cache

Disable caching. Set to `False` by default.

7.2.6 sparql

Wikidata SPARQL endpoint, set to `http://query.wikidata.org/sparql` by default.

7.3 Wikidata editing

Changing Wikidata has not been implemented yet.

7.3.1 dry

Don't perform any edits on Wikidata

7.4 Additional output control

7.4.1 header

Read/write CSV/BEACON without header. *This option is experimental.*

7.4.2 relation

Mapping relation URI such as `skos:exactMatch` or `owl:sameAs`.

7.4.3 debug

Enable debugging mode.

CHAPTER 8

Python library

The Python API is in a very preliminary state so better not build on it yet!

8.1 Index

- genindex
- search

CHAPTER 9

Contributing

Feedback and contributions are very welcome!

This is my first Python project so I try to follow best-practice I could find. Please let me know if I either missed something or if I was to pedandic!

9.1 Running from source

You can manually execute wdmapper from source tree for testing:

```
$ ./wdmapper.py
```

9.2 Issue tracker

Please report bugs and feature requests at https://github.com/gbv/wdmapper/issues!

9.3 Development requirements

Additional requirements for development are listed in `dev-requirements.txt`. Install via `pip install -r dev-requirements.txt`.

9.4 Testing

Please test functionality with unit tests, located in directory `tests`.

```
$ python setup.py test
$ pytest
```

Run a single test file:

```
$ pytest tests/test\_whatever.py
```

Test plugins and default options are configured in setup.cfg.

To run all tests with multiple versions of Python use tox:

```
$ tox
```

It is important to check with tox to ensure compatibility with both Python 2.7 and Python 3.x. Which versions to test with and other options are configured in `tox.ini`.

Tests are also executed at https://travis-ci.org/gbv/wdmapper after pushing commits to GitHub.

CHAPTER 10

History

10.1 0.0.17 (2017-05-29)

- Fix lookup by URI Template

10.2 0.0.16 (2017-05-24)

- Prefer property P1921 (RDF URI Template) over P1630 (formatter URL)

10.3 0.0.15 (2017-05-07)

- Use empty string instead of Wikidata-ID as default annotation
- Allow to disable labels by setting language to empty string

10.4 0.0.13 (2017-04-19)

- Extend output formats
- Fix Unicode output when writing to files

10.5 0.0.9 (2017-04-13)

- Add ‘quicks’ output format for deltas to be used in QuickStatements tool
- Remove ‘help’ command

10.6 0.0.8 (2017-04-06)

- Add ‘json’ output format (JSKOS Concept Mappings, .ndjson)
- Include source and target concept scheme URI if available

10.7 0.0.7 (2017-01-17)

- Add option ‘language’ to select language of labels
- Add option ‘sparql’ to customize Wikidata SPARQL endpoint
- Add option ‘type’ to filter Wikidata items by class

10.8 0.0.6 (2017-01-13)

- Add basic support of BEACON input format
- Improve lookup of properties by label

10.9 0.0.5 (2017-01-05)

- Add ntriples output format
- Add option ‘relation’
- Fix command ‘check’ for indirect links
- Fix lookup of properties by URL template

10.10 0.0.4 (2017-01-03)

- Implement command ‘check’
- Rename command ‘about’ to ‘head’
- Include item label for one-way mappings from Wikidata

10.11 0.0.3 (2016-12-23)

- Rename command ‘echo’ to ‘convert’
- Use command ‘convert’ by default
- Fix and extend input/output from stdin/stdout and files

10.12 0.0.2 (2016-12-20)

- Add BEACON output format
- Implement command ‘diff’
- Add arguments –sort and –no-cache

10.13 0.0.1 (2016-12-16)

- First release at PyPI
- Implemented lookup of properties and mappings (commands ‘property’ and ‘get’)

10.14 0.0.0 (2016-12-08)

- Create repository boilerplate