Python-OpenTree Documentation

Release 0.1

OpenTreeOfLife

Jun 22, 2021

Contents:

CHAPTER 1

python-opentree

This package is a python library designed to make it easier to work with web services and data resources associated with the Open Tree of Life project. The git repo is at https://github.com/OpenTreeOfLife/python-opentree.

Please refer to this markdown document or opentree's documentation website https://opentree.readthedocs.io/en/latest/install.html for more details on installation instructions, function usage, running tutorials, real life examples, and tools for developers.

If you have questions or comments, submit a GitHub issue or email ejmctavish ucmerced.edu.

For examples scripts see: https://opentree.readthedocs.io/en/latest/running_examples.html

For example Jupyter notebooks see: https://opentree.readthedocs.io/en/latest/notebooks.html

SSB 2020 workshop curriculum is posted at: Open Tree workshop at the 2020 SSB Meeting in Gainesville, FL.

CHAPTER 2

Installation

If you don't need the latest version you, can simply use:

```
pip install opentree
```

If you are developer who wants to install multiple times, you probably want to clone the code locally, and install it in a virtual environment.

To do so, run:

```
git clone https://github.com/OpenTreeOfLife/python-opentree.git #Copies to code to␣
↪your machine
cd python-opentree
python3 -m venv env #Creates a virtual environment named env (you only need to ever␣
↪run this once)
source env/bin/activate # activates the virtual environment (you will wnat to do this␣
↪each time you are using the package)
pip install -r requirements.txt # Installs the requiremnets
pip install -e . # Installs the open tree package
```

You can deactivate the virtual environment by running

```
deactivate
```

If you plan to run the example jupyter notebooks, install jupyter within the virtual environment as well using:

Create and activate the virtual environment as shown above

```
source env/bin/activate
pip install ipykernel
python -m ipykernel install --user --name=opentree
pip install jupyter
jupyter notebook docs/notebooks/ # This will open the example notebooks
```

CHAPTER 3

Example scripts

opentree comes packaged with a set of example scripts, wrapping common function calls.

These wrap most of the the API calls described in https://github.com/OpenTreeOfLife/germinator/wiki/Open-Tree-of-Life-Web-APIs, and current API documentation is stored there.

3.1 About

An about call returns the current version of the OpenTree synthetic tree and taxonomy:

```
python examples/about.py
```

Response:

```
taxonomy_about
{
  "author": "open tree of life project",
  "name": "ott",
  "source": "ott3.2draft9",
  "version": "3.2",
  "weburl": "https://tree.opentreeoflife.org/about/taxonomy-version/ott3.2"
}

synth_tree_about
{
  "date_created": "2019-12-23 11:41:23",
  "filtered_flags": [
    "major_rank_conflict",
    "major_rank_conflict_inherited",
    "environmental",
    "viral",
    "barren",
    "not_otu",
```

(continues on next page)

(continued from previous page)


```
    "hidden",
    "was_container",
    "inconsistent",
    "hybrid",
    "merged"
  ],
  "num_source_studies": 1162,
  "num_source_trees": 1216,
  "root": {
    "node_id": "ott93302",
    "num_tips": 2391916,
    "taxon": {
      "name": "cellular organisms",
      "ott_id": 93302,
      "rank": "no rank",
      "tax_sources": [
        "ncbi:131567"
      ],
      "unique_name": "cellular organisms"
    }
  },
  "synth_id": "opentree12.3",
  "taxonomy_version": "3.2draft9"
}
```


3.2 Studies calls

3.2.1 study_properties

Get all searchable properties for trees and studies:

```
python examples/study_properties.py
```

3.2.2 Find studies

Search studies by property. Property can mb any of the above listed 'study properties', but the most common study search properties are:

"ot:studyPublicationReference", "ot:focalCladeOTTTaxonName", "ot:curatorName",

The default response just returns the study ID, adding the –verbose flag retuns the full publication references.

To find studies published in the journal Copeia:

```
python examples/find_studies.py --property "ot:studyPublicationReference" Copeia --
↪verbose
```

Property can be any of the above listed 'study properties', but the most common study search properties are:
"ot:studyPublicationReference", "ot:focalCladeOTTTaxonName", "ot:curatorName",

Response:

```
"matched_studies": [
{
  "ot:curatorName": [
    "Matt Girard"
  ],
  "ot:dataDeposit": "",
  "ot:focalClade": 814725,
  "ot:focalCladeOTTTaxonName": "Etheostoma",
  "ot:studyId": "ot_1930",
  "ot:studyPublication": "http://dx.doi.org/10.1643/ci-18-054",
  "ot:studyPublicationReference": "Matthews, William J., and Thomas F. Turner.␣
↪\u201cRedescription and Recognition of Etheostoma Cyanorum from Blue River,␣
↪Oklahoma.\u201d Copeia 107, no. 2 (April 11, 2019): 208. doi:10.1643/ci-18-054.",
  "ot:studyYear": 2019,
  "ot:tag": []
},
... cut off for length
```

3.2.3 Find trees

Search tress by property Property can be any of the above listed 'tree properties', but the most common tree search properties are:

> "ot:branchLengthTimeUnit", "ot:inGroupClade", "ot:ottTaxonName", "ot:branchLengthDescription", "ntips", "ot:ottId", "ot:branchLengthMode",

To find trees that contain lemurs:

```
python examples/find_trees.py --property ot:ottTaxonName Lemur
```

or to avoid spelling or typographical errors, you can use the ott id for lemurs, 913932 https://tree.opentreeoflife.org/taxonomy/browse?id=913932:

```
python examples/find_trees.py --property ot:ottId 913932
```

3.2.4 Get study

Get the full study as nexson from study id:

```
python examples/get_study.py pg_2067
```

3.2.5 Get tree

Get a tree from a study in Newick or Nexus format

For example, to get one of the lemur trees found above:

```
python examples/get_tree.py pg_2067 tree4259 --format newick
```

3.3 Taxonomy

3.3.1 TNRS

To get the taxonomic identifiers for a name:

```
python examples/tnrs_match_names.py Lemur
```

if you think you may have typos, add –do-approximate-matching:

```
python examples/tnrs_match_names.py Lemun --do-approximate-matching
```

To combine a genus and species, use quotation marks:

```
python examples/tnrs_match_names.py "Bos taurus"
```

Approximate name matching can be sped up by restricting the 'context' for the searches You can find out the possible contexts using:

```
python examples/tnrs_contexts.py
```

and then applying them:

```
python examples/tnrs_match_names.py Lemun --do-approximate-matching --context Mammals
```

3.3.2 Taxon information

To get more information for taxon which you have the ott id for:

```
python examples/taxon_info.py --ott-id 913932
```

Or the taxonomic subtree descending from a node:

```
python examples/taxon_info.py --ott-id 913932
```

3.3.3 Taxon mrca

To get the most recent common ancestor in the taxonomy for multiple taxa e.g. humans (ott:770309) and lemurs (ott:913932)(may differ from synth tree mrca):

```
python examples/taxon_mrca.py --ott-ids 770309,913932
```

You can pass in the ottids with or without 'ott' e.g. 'ott770309,ott913932', but there cannot be a space between taxa.

3.4 Synthetic tree

To get the most recent common ancestor in the synthetic tree for multiple taxa e.g. humans (ott:770309) and lemurs (ott:913932):

3.4.1 Synth mrca

python examples/synth_mrca.py –ott-ids 770309,913932

Response::

{

"mrca": { "node_id": "mrcaott786ott3428", "num_tips": 743, "partial_path_of": {

"ot_1366@Tr98763": "Tn14487470", "ot_722@tree1": "node47", "pg_1428@tree2855": "node610302", "pg_2812@tree6545": "node1135880"

}, "supported_by": {

"pg_2647@tree6169": "node1053665", "pg_2741@tree6645": "node1159651"

}, "terminal": {

"ot_508@tree2": "ott83926", "pg_2822@tree6569": "ott83926"

}

}, "nearest_taxon": {

"name": "Primates", "ott_id": 913935, "rank": "order", "tax_sources": [

"ncbi:9443", "gbif:798", "irmng:11338"

], "unique_name": "Primates"

}, "source_id_map": {

"ot_1366@Tr98763": { "git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "ot_1366", "tree_id": "Tr98763"

}, "ot_508@tree2": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "ot_508", "tree_id": "tree2"

}, "ot_722@tree1": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "ot_722", "tree_id": "tree1"

}, "pg_1428@tree2855": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "pg_1428", "tree_id": "tree2855"

}, "pg_2647@tree6169": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "pg_2647", "tree_id": "tree6169"

}, "pg_2741@tree6645": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "pg_2741", "tree_id": "tree6645"

}, "pg_2812@tree6545": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "pg_2812", "tree_id": "tree6545"

}, "pg_2822@tree6569": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "pg_2822", "tree_id": "tree6569"

}

}, "synth_id": "opentree12.3" }

3.4.2 Synth subtree

To get the full subtree descending from a node in the the synthetic tree:

```
python examples/synth_subtree.py --ott-id 913932 --outfile tmp.txt
```

By default this will write the tree to screen as an ascii plot, and write the newick to the file location listed in outfile:

```
python examples/synth_subtree.py --ott-id 913932 --outfile tmp.txt
```

You can specify the label format out the output tree using *–label-format* with options [id, name, name_and_id]:

```
python examples/synth_subtree.py --ott-id 913932 --label-format name --outfile tmp.txt
```

3.4.3 Synth induced subtree

To get the relationships among cows (Bos taurus ott490099), camels (Camelus dromedarius ott510752), and whales (Orcinus orca ott124215) By default this will write the tree to screen as an ascii plot, and write the newick to the file location listed in outfile:

```
python examples/synth_induced_subtree.py --ott-ids ott490099,ott510752,ott124215 --
↪outfile tmp.nwk
```

3.4.4 Synth node info

All nodes in the syntehtic tree are supported by published studies, taxonomy, or both.

To get more information the studies that are resolving a node in the syntehtic tree, you can get node information:

```
python examples/synth_node_info.py --node-id mrcaott354607ott374748
```

Response::

{

"node_id": "mrcaott354607ott374748", "num_tips": 21, "query": "mrcaott354607ott374748", "resolves": {

"pg_2812@tree6545": "node1135857"

}, "source_id_map": {

"ot_1344@Tr105486": { "git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "ot_1344", "tree_id": "Tr105486"

}, "pg_1217@tree2455": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "pg_1217", "tree_id": "tree2455"

}, "pg_1428@tree2855": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "pg_1428", "tree_id": "tree2855"

}, "pg_2647@tree6169": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "pg_2647", "tree_id": "tree6169"

}, "pg_2812@tree6545": {

"git_sha": "3008105691283414a18a6c8a728263b2aa8e7960", "study_id": "pg_2812", "tree_id": "tree6545"

}

}, "supported_by": {

"ot_1344@Tr105486": "Tn16531763"

}, "synth_id": "opentree12.3", "terminal": {

"pg_1217@tree2455": "node566205", "pg_1428@tree2855": "node610191", "pg_2647@tree6169": "node1053583"

3.5 Diagnosing subproblem solutions

To dig deeper into how different trees included in synthesis support or conflict with nodes in the inferred synthetic tree, you can examine what trees support and conflict with a given node's resolution, and experiment with alternate tree rankings.

For example, Drosophila (ott34907) is not found in the synthetic tree. Why not:??

```
python examples/diagnose_solution_for –ott-id 34907
```

You can then interactively view the subproblems. The subproblem synthesis algorithm described in depth in Redelings and Holder 2017 (https://peerj.com/articles/3058/).

CHAPTER 4

Tutorials and Interactive examples

We have developed Jupypter notebooks demonstrating how to use python-opentree interactively as part of a python workflow.

4.1 Example notebooks

To run these notebooks follow the installation instuctions in https://opentree.readthedocs.io/en/latest/readme.html#installation.

And start the notebook server by running:

```
jupyter notebook docs/notebooks/
```

For more information on Jupyter notebooks, see [https://jupyter.org/]

4.2 Workshop material

All the notes from a workshop presented in winter 2020 are in http://opentreeoflife.github.io/SSBworkshop/, git repo with code and data at https://github.com/snacktavish/OpenTree_SSB2020.

4.2.1 Getting a tree for a list of species names

https://github.com/snacktavish/OpenTree_SSB2020/blob/master/notebooks/DEMO_OpenTree.ipynb

4.2.2 Getting a tree for taxa found in GBIF

https://github.com/snacktavish/OpenTree_SSB2020/blob/master/notebooks/DEMO_GBIF_OpenTree.ipynb

CHAPTER 5

API Documentation

OT object. High level wrapper for OpenTree calls

class opentree.ot_object.**FilesServerWrapper**(*api_endpoint='files'*, *run_mode=<WebServiceRunMode.RUN: 1>*)

This class provides a mid-level wrapper for interaction with OT web services and data.

class opentree.ot_object.**OpenTree**(*api_endpoint='production'*, *run_mode=<WebServiceRunMode.RUN: 1>*)

This class provides a high-level wrapper for interaction with OT web services and data. The method names are intended to be clear to a wide variety of users, rather than necessarily matching the API calls directly.

about()

Get information about the Open Tree of Life taxonomy and the synthetic tree.

conflict_info(*study_id*, *tree_id*, *compare_to='synth'*)

Get node status data from any tree in the Open Tree of Life Phylesystem.

study_id [single character value] The study id from Open Tree of Life.

tree_id [single character value] The tree id of a tree within the study id provided.

compare_to [a single character value] Usually, you want this to be 'synth', to compare to the synthetic tree. Alternatively, you can compare your tree to any other tree in phylesystem.

conflict_str(*tree_str*, *compare_to='synth'*)

Get node status data from a newick string tree with ott_ids as labels, following the rough format: "(('_nd1_ott770315','newick_nd2_ott417950')'_nd3_','_nd4_ott158484')'_nd5';".

tree_str: a tree in 'conflict formatted' newick string compare_to : a single character value

Usually, you want this to be 'synth', to compare to the synthetic tree. Alternatively, you can compare your tree to any other tree in phylesystem.

find_studies(*value*, *search_property*, *exact=False*, *verbose=False*)

Get study ids that match a certain value of a given search property.

value [single character value] The study id from Open Tree of Life.

search_property [single character value] Any value from studies_properties.

exact : boolean

verbose : boolean

find_trees(*value*, *search_property*, *exact=False*, *verbose=False*)
Get trees that match a certain value of a given search property.

value [single character value] The study id from Open Tree of Life.

search_property [single character value] Any value from studies_properties.

exact : boolean

verbose : boolean

get_citations(*studies*)
Returns study citations from a list of study or tree ids

get_matchdict_from_taxlist(*list_of_taxa*)
Input: a list of taxon names Returns: matches - a dictionary of name:ott_id and failed - a set of the names that were not found.

get_ottid_from_gbifid(*gbif_id*)
Returns an ott id for a gbif id ott_id is set to 'None' if the gbif id is not found in the Open Tree Taxanomy

get_ottid_from_name(*spp_name*)
Returns an ott id for a string - requires exact match. ott_id is set to 'None' if the name is not found in the Open Tree Txanomy

get_otus(*study_id*)
Get OTUs from a study in the Open Tree of Life Phylesystem.

study_id [single character value] The study id from Open Tree of Life.

get_study(*study_id*)
Get a study and its associated metadata.

study_id [single character value] The study id from Open Tree of Life.

get_tree(*study_id*, *tree_id*, *tree_format='nexson'*, *label_format='ot:originallabel'*, *demand_success=False*)
Get a source tree from phylesystem and its associated metadata.

study_id [single character value] The study id from Open Tree of Life.

tree_id [single character value] The tree id of a tree within the study id provided.

tree_format [single character value] Must be one of "newick", "nexson", "nexus", or "object" If tree format is newick or nexus, returns tree as string in that format. If "nexson", returns semi-useless tree nexson w/o OTUS.

label_format [single character value] Must be one of "ot:originallabel", "ot:ottid", or "ot:otttaxonname". "ot:originallabel" returns the tree with tip labels as it was originally

submitted to phylesystem by a curator.

"ot:ottid" returns a tree with tip labels corresponding to the matching ott id.

"ot:otttaxonname" returns a tree with tip labels corresponding to the matching ott taxon name.

demand_success [boolean] Whether to return an error or return a somewhat failed output silently.

studies_properties()
Get properties that can be used to search across studies and trees in phylesystem.

synth_induced_tree(*node_ids=None*, *ott_ids=None*, *label_format='name_and_id'*, *ignore_unknown_ids=False*)
Get an induced subtree

synth_mrca(*node_ids=None*, *ott_ids=None*, *ignore_unknown_ids=True*)
Get the most recent common ancestor of a group of taxa on the synthetic Open Tree of Life

synth_node_info(*node_ids=None*, *node_id=None*, *ott_id=None*, *include_lineage=False*)
Get information of a node

synth_subtree(*node_id=None*, *ott_id=None*, *tree_format='newick'*, *label_format='name_and_id'*, *height_limit=None*)
Get a subtree

taxon_info(*ott_id=None*, *source_id=None*, *include_lineage=False*, *include_children=False*, *include_terminal_descendants=False*)
Get taxonomic information for a given taxon in the Open Tree taxonomy.

ott_id [single character value] The OTT id of a taxon.

source_id : maybe single character value

include_lineage : boolean

include_children : boolean

include_terminal_descendant : boolean

taxon_mrca(*ott_ids=None*)

Get the node corresponding to the most recent commom ancestor (mrca) of a taxon in the synthetic Open Tree of Life tree.

Notes from Luna: Does it work with just one id? Since it is not always a taxon mrca, should it be called get_mrca?

ott_ids : maybe single character value

taxon_subtree(*ott_id=None*, *label_format='name_and_id'*)
Get a subtree of a particular taxon

tnrs_autocomplete(*name*, *context_name=None*, *include_suppressed=False*)
Taxonomic name resolution service autocomplete

tnrs_contexts()
Get a list of taxonomic contexts that can be used to constraint a TNRS match.

tnrs_infer_context(*names*)
Infer taxonomic context for names via a TNRS (Taxonomic Name Resolution Service) match.

tnrs_match(*names*, *context_name=None*, *do_approximate_matching=False*, *include_suppressed=False*)
Match taxon names to Open Tree Taxonomy using TNRS (Taxonomic Name Resolution Service).

class opentree.object_conversion.**DendropyConvert**
Class to convert newicks to dendropy objects

class opentree.ot_command_line_tool.**OTCommandLineTool**(*usage*, *name=None*, *common_args=None*)
Helper class for writing a script that uses a common set of Open Tree command line options.

parse_cli(*arg_list=None*)
Parses *arg_list* or sys.argv (if None), handles basic options, returns OpenTree and args.

May call sys.exit - if the user requested an option like –version to display info and exit.

Returns an OpenTree instance configured with the specified api_endpoint and the args object returned by the argparse object's parse_args method

class opentree.ot_ws_wrapper.**OTWebServiceWrapper**(*api_endpoint*, *run_mode=<WebServiceRunMode.RUN: 1>*)

This class provides a wrapper to the Open Tree of Life web service methods. Actual HTTP calls are handled by methods implemented in the base class for clarity of this code. API method calls will be mappable to methods in this class. The methods implemented here do argument checking and conversion of the returned JSON to more usable objects.

Miscellaneous light-weight functions for common operations when working with Open Tree data

opentree.util.**get_suppressed_taxon_flag_expl_url**()

Returns the current URL describing taxon flags that lead to suppression

opentree.util.**ott_str_as_int**(*o*)

Returns the OTT Id *o* as an integer if *o* is an integer or a string starting with ott (case-insensitive).

Raises a ValueError if the string does not match ^(OTT)?[0-9]+$

opentree.util.**write_node_info_links_to_input_trees**(*blob*, *out=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*)

Writes a summary of the support/conflict info from a ToL/node_info call response *blob* to stream *out*

exception opentree.ws_wrapper.**OTClientError**(*message*, *call_record=None*)

This type of error is raised when the calling code does not make a legitimate request based on the Open Tree of Life API's (see https://opentreeoflife.github.io/develop/api).

exception opentree.ws_wrapper.**OTWebServicesError**(*message*, *call_record=None*)

This type of error is raised when a web-service call fails for a reason that is impossible or difficult to diagnose. The string representation of the error should contain some helpful information.

class opentree.ws_wrapper.**WebServiceCallRecord**(*service_wrapper*, *url*, *http_method*, *headers*, *data*)

Wrapper around a web-service call, returned by WebServiceWrapper methods.

The main client methods to call are:

- __bool__ (check if status code was 200)
- __str__ (explanation of the call status
- *write_response* (writes call explanation and response, if there was one).

The most commonly used properties:

- url: string
- response: a requests response object
- status_codeL: None or the HTTP status code as an integer
- response_dict: None, decoding of a JSON response or {'content' : raw_content} (for non-JSON methods)

If the API call returns some encoding of a tree, then the *tree* property of the WebServiceCallRecord can be used to decode the response.

curl_call

Returns a string that is a curl representation of the call

class opentree.ws_wrapper.**WebServiceRunMode**

An enumeration.

CHAPTER 6

Indices and tables

- search

Python Module Index

o

Index

W