opentargets Documentation

Release 3.1.15

Open Targets Core team

Feb 26, 2019

Contents

opentargets-py is the official python client for the Open Targets REST API.

This client allows you to query the API automatically handling all the calls and returning data in a pythonic way.

Main advantages of using the client versus querying the REST API directly:

- Include wrappers for all public methods, with query validation
- Tools for the most common calls (E.g. get data for a target gene symbol even if you do not know its Ensembl Gene Id)
- Supports automatic retrieval of paginated results with an iterator pattern
- Easily save query results as JSON, CSV or Excel file
- Handles Authentication
- Handles fair usage limits transparently
- Follows HTTP cache as set by the REST API
- Experimental HTTP2 support for better performance (beware the client library is in alpha)

This client is supported for Python 3.5 and higher. Works with lower versions (including python 2.7) on a best effort basis. Take a look at the *Tutorial* to get an idea of what you can do.

CHAPTER 1

Installation

```
pip install opentargets
```

or directly from github

```
pip install git+git://github.com/opentargets/opentargets-py.git
```

Get the source code (or make your own fork) on GitHub : opentargets/opentargets-py

CHAPTER 2

Documentation

2.1 Tutorial

2.1.1 Quick Start

import the high level client

```
from opentargets import OpenTargetsClient
ot = OpenTargetsClient()
```

or if you have an API key

```
from opentargets import OpenTargetsClient
ot = OpenTargetsClient(auth_app_name=<YOUR_APIKEY_APPNAME>, auth_secret=<YOUR_APIKEY_
↪SECRET>,)
```

search for a target:

```
search_result = ot.search('BRAF')
print(search_result[0])
```

search associations for a target:

```
a_for_target = ot.get_associations_for_target('BRAF')
for a in a_for_target:
    print(a['id'], a['association_score']['overall'])
```

search associations for a disease:

```
a_for_disease = ot.get_associations_for_disease('cancer')
```

get an association by id:

```
print(ot.get_association('ENSG00000157764-EFO_0005803')[0])
```

get evidence for a target:

```
e_for_target = ot.get_evidence_for_target('BRAF')
for evidence_json in e_for_target.to_json():
    print(evidence_json)
```

get evidence for a disease:

```
e_for_disease = ot.get_evidence_for_disease('medulloblastoma')
```

get an evidence by id:

```
print(ot.get_evidence('5cf863da265c32d112ff4fc3bfc25ab3')[0])
```

get stats about the release:

```
print(ot.get_stats().info)
```

use incremental filter:

```
>>> from opentargets import OpenTargetsClient
>>> client = OpenTargetsClient()
>>> response = client.filter_associations()
>>> response
<opentargets.conn.IterableResult object at 0x105c32d68>
>>> print(response)
2484000 Results found
>>> response.filter(target='ENSG00000157764')
>>> print(response)
865 Results found | parameters: {'target': 'ENSG00000157764'}
>>> response.filter(direct=True)
>>> print(response)
454 Results found | parameters: {'target': 'ENSG00000157764', 'direct': True}
>>> response.filter(scorevalue_min=0.2)
>>> print(response)
156 Results found | parameters: {'scorevalue_min': 0.2, 'target': 'ENSG00000157764',
↪'direct': True}
>>> response.filter(therapeutic_area='efo_0000701')
>>> print(response)
12 Results found | parameters: {'therapeutic_area': 'efo_0000701', 'scorevalue_min':␣
↪0.2, 'target': 'ENSG00000157764', 'direct': True}
>>> for i, r in enumerate(response):
...     print(i, r['id'], r['association_score']['overall'], r['disease']['efo_info'][
↪'label'])
...
0 ENSG00000157764-EFO_0000756 1.0 melanoma
1 ENSG00000157764-Orphanet_1340 1.0 Cardiofaciocutaneous syndrome
2 ENSG00000157764-Orphanet_648 1.0 Noonan syndrome
3 ENSG00000157764-Orphanet_500 1.0 LEOPARD syndrome
4 ENSG00000157764-EFO_0002617 1.0 metastatic melanoma
5 ENSG00000157764-EFO_0000389 0.9975053926198617 cutaneous melanoma
6 ENSG00000157764-EFO_0004199 0.6733333333333333 dysplastic nevus
7 ENSG00000157764-EFO_0002894 0.6638888888888889 amelanotic skin melanoma
8 ENSG00000157764-EFO_1000080 0.5609555555555555 Anal Melanoma
9 ENSG00000157764-EFO_0000558 0.5602555555555556 Kaposi's sarcoma
```

(continues on next page)

(continued from previous page)

```
10 ENSG00000157764-EFO_1000249 0.5555555555555556 Extramammary Paget Disease
11 ENSG00000157764-Orphanet_774 0.21793721666666668 Hereditary hemorrhagic␣
↪telangiectasia
```

export a table with association score for each datasource into an excel file:

```
>>> from opentargets import OpenTargetsClient
>>> client = OpenTargetsClient()
>>> response = client.get_associations_for_target('BRAF',
...     fields=['association_score.datasource*',
...             'association_score.overall',
...             'target.gene_info.symbol',
...             'disease.efo_info.*']
...     )
>>> response
865 Results found | parameters: {'target': 'ENSG00000157764', 'fields': ['association_
↪score.datasource*', 'association_score.overall', 'target.gene_info.symbol',
↪'disease.efo_info.label']}
>>> response.to_excel('BRAF_associated_diseases_by_datasource.xls')
>>>
```

If you want to change the way the associations are scored using just some datatype you might try something like this:

```
>>> from opentargets import OpenTargetsClient
>>> from opentargets.statistics import HarmonicSumScorer
>>> ot = OpenTargetsClient()
>>> r = ot.get_associations_for_target('BRAF')
>>> interesting_datatypes = ['genetic_association', 'known_drug', 'somatic_mutation']
>>> def score_with_datatype_subset(datatypes, results):
...     for i in results:
...         datatype_scores = i['association_score']['datatypes']
...         filtered_scores = [datatype_scores[dt] for dt in datatypes]
...         custom_score = HarmonicSumScorer.harmonic_sum(filtered_scores)
...         if custom_score:
...             yield (custom_score, i['disease']['id'], dict(zip(datatypes, filtered_
↪scores))) #return some useful data
>>> for i in score_with_datatype_subset(interesting_datatypes, r):
...     print(i)
(1.8333333333333333, 'EFO_0000701', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 1.0})
(1.8333333333333333, 'EFO_0000616', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 1.0})
(1.8333333333333333, 'EFO_0000311', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 1.0})
(1.8333333333333333, 'EFO_0001379', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 1.0})
(1.8333333333333333, 'EFO_0000313', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 1.0})
(1.8333333333333333, 'EFO_0005803', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 1.0})
(1.8333333333333333, 'EFO_0001642', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 1.0})
(1.587037037037037, 'EFO_0000319', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 0.2611111111111111})
(1.5949074074074074, 'EFO_0000508', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 0.2847222222222222})
1.5, 'Orphanet_183530', {'genetic_association': 1.0, 'known_drug': 1.0, 'somatic_
↪mutation': 0.0})
```

(continues on next page)

(continued from previous page)

```
(1.5, 'EFO_0003777', {'genetic_association': 1.0, 'known_drug': 1.0, 'somatic_mutation
↪': 0.0})
(1.5, 'Orphanet_98054', {'genetic_association': 1.0, 'known_drug': 1.0, 'somatic_
↪mutation': 0.0})
(1.5, 'Orphanet_99739', {'genetic_association': 1.0, 'known_drug': 1.0, 'somatic_
↪mutation': 0.0})
(1.5, 'Orphanet_217595', {'genetic_association': 1.0, 'known_drug': 1.0, 'somatic_
↪mutation': 0.0})
(1.5, 'Orphanet_183570', {'genetic_association': 1.0, 'known_drug': 1.0, 'somatic_
↪mutation': 0.0})
(1.5, 'Orphanet_98733', {'genetic_association': 1.0, 'known_drug': 1.0, 'somatic_
↪mutation': 0.0})
(1.6050444693876402, 'EFO_0000684', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 0.31513340816292035})
(1.6050444693876402, 'EFO_0003818', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 0.31513340816292035})
(1.6050444693876402, 'EFO_0003853', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 0.31513340816292035})
(1.6050444693876402, 'EFO_0001071', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 0.31513340816292035})
(1.5408333333333333, 'EFO_0000618', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 0.1225})
(1.6803112925534462, 'EFO_0000228', {'genetic_association': 1.0, 'known_drug': 0.
↪9357799925142999, 'somatic_mutation': 0.6372638888888889})
(1.6013073034769463, 'EFO_0000512', {'genetic_association': 1.0, 'known_drug': 1.0,
↪'somatic_mutation': 0.303921910430839})
```

Using insecure SSL? local certificate? an HTTP or SOCKS proxy?

```
   >>> from opentargets import OpenTargetsClient
   >>> from opentargets.statistics import HarmonicSumScorer
   >>> ot = OpenTargetsClient(verify = False) # SSL not verified
   >>> ot = OpenTargetsClient(verify = 'path to my certificate') # local certificate
   >>> ot = OpenTargetsClient(proxies = {
                                        'http': 'http://10.10.1.10:3128',
                                        'https': 'http://10.10.1.10:1080',
                                       }) # HTTP proxies
   >>> ot = OpenTargetsClient(proxies = {
                                        'http': 'socks5://user:pass@host:port',
                                        'https': 'socks5://user:pass@host:port'
                                       }) # HTTP proxies

verify and proxies options  works as in the (requests library)[http://docs.python-
↪requests.org/en/master/user/advanced/]
```

2.2 Utility

The opentargets client expose a set of helpers methods to mimic and extend the workflow of the webapp This includes:

- Searching for a target or a disease by common name
- Retrieve a specific target-disease evidence or association
- Dynamically apply filters to a dataset of evidence or associations
- Export data to JSON, csv, excel or directly into Pandas Dataframe

- Get statistics on the data served by the platform
- Support to a custom scoring scheme to a dataset

TODO

2.3 Connection

The opentargets client expose a *Connection* class that efficiently retireves data from the REST API, supporting input validation, authentication, caching, and fair usage limits. It is possible to get a list of endpoints available in the REST API, their documentation and call each of them directly. Each call is returned as an *IterableResult*. With *IterableResult* instance it is possible to stream directly content from the REST API by using an interator pattern. Data points are fetched dynamically and memory usage is optimised.

TODO

2.4 Code Documentation

2.4.1 Module contents

This module communicate with the Open Targets REST API with a simple client, and requires not knowledge of the API.

class opentargets.**OpenTargetsClient**(***kwargs*)

Bases: object

Main class to use to get data from the Open Targets REST API available at targetvalidation.org (or your private instance)

Init the client and start a connection

Keyword Arguments ****kwargs** – all params forwarded to opentargets.conn.Connection object

close()

filter_associations(***kwargs*)

Retrieve a set of associations by applying a set of filters

Keyword Arguments ****kwargs** – are passed as parameters to the /public/association/filterby method of the REST API

Returns Result of the query

Return type *IterableResult*

filter_evidence(***kwargs*)

Retrieve a set of evidence by applying a set of filters

Keyword Arguments ****kwargs** – are passed as parameters to the /public/evidence/filterby method of the REST API

Returns Result of the query

Return type *IterableResult*

get_association(*association_id*, ***kwargs*)

Retrieve a specific Association object from the REST API provided its ID

Parameters **association_id** (*str*) – Association ID

Keyword Arguments ****kwargs** – are passed as other parameters to the /public/association method of the REST API

Returns Result of the query

Return type *IterableResult*

get_associations_for_disease(*disease*, ***kwargs*)

Same as `OpenTargetsClient.filter_associations` but accept any string as *disease* parameter and fires a search if it is not a valid disease identifier

Parameters **disease** (*str*) – a disease identifier or a string to search for a disease mapping

Keyword Arguments ****kwargs** – are passed as parameters to the /public/association/filterby method of the REST API

Returns Result of the query

Return type *IterableResult*

get_associations_for_target(*target*, ***kwargs*)

Same as `OpenTargetsClient.filter_associations` but accept any string as *target* parameter and fires a search if it is not an Ensembl Gene identifier

Parameters **target** (*str*) – an Ensembl Gene identifier or a string to search for a gene mapping

Keyword Arguments ****kwargs** – are passed as parameters to the /public/association/filterby method of the REST API

Returns Result of the query

Return type *IterableResult*

get_disease(*disease_id*, ***kwargs*)

Retrieve a specific disease object from the REST API provided its ID

Parameters **evidence_id** – OT disease ID (EFO, Orphanet, ...)

Keyword Arguments ****kwargs** – are passed as other parameters to the /private/disease method of the REST API

Returns Result of the query

Return type *IterableResult*

get_evidence(*evidence_id*, ***kwargs*)

Retrieve a specific Evidence object from the REST API provided its ID

Parameters **evidence_id** –

Keyword Arguments ****kwargs** – are passed as other parameters to the /public/evidence method of the REST API

Returns Result of the query

Return type *IterableResult*

get_evidence_for_disease(*disease*, ***kwargs*)

Same as `OpenTargetsClient.filter_evidence` but accept any string as *disease* parameter and fires a search if it is not a valid disease identifier

Parameters **disease** (*str*) – a disease identifier or a string to search for a disease mapping

Keyword Arguments ****kwargs** – are passed as parameters to the /public/evidence/filterby method of the REST API

Returns Result of the query

Return type *IterableResult*

get_evidence_for_target(*target*, ***kwargs*)
Same as `OpenTargetsClient.filter_evidence` but accept any string as *target* parameter and fires a search if it is not an Ensembl Gene identifier

Parameters **target** (*str*) – an Ensembl Gene identifier or a string to search for a gene mapping

Keyword Arguments ****kwargs** – are passed as parameters to the /public/evidence/filterby method of the REST API

Returns Result of the query

Return type *IterableResult*

get_metrics(***kwargs*)
Returns metrics about the data served by the REST API

Returns Result of the query

Return type *IterableResult*

get_similar_disease(*disease*, ***kwargs*)
Return targets sharing a similar patter nof association to diseases Accepts any string as *disease* parameter and fires a search if nothing is retireved on a first attempt

Parameters **disease** (*str*) – a disease identifier or a string to search for a disease mapping

Keyword Arguments ****kwargs** – are passed as parameters to the /private/relation/disease method of the REST API

Returns Result of the query

Return type *IterableResult*

get_similar_target(*target*, ***kwargs*)
Return targets sharing a similar patter nof association to diseases Accepts any string as *target* parameter and fires a search if it is not an Ensembl Gene identifier

Parameters **target** (*str*) – an Ensembl Gene identifier or a string to search for a gene mapping

Keyword Arguments ****kwargs** – are passed as parameters to the /private/relation/target method of the REST API

Returns Result of the query

Return type *IterableResult*

get_stats(***kwargs*)
Returns statistics about the data served by the REST API

Returns Result of the query

Return type *IterableResult*

get_target(*target_id*, ***kwargs*)
Retrieve a specific target object from the REST API provided its ID

Parameters **target_id** – Ensembl ID

Keyword Arguments ****kwargs** – are passed as other parameters to the /private/target method of the REST API

Returns Result of the query

Return type *IterableResult*

search(*query*, ***kwargs*)
Search a string and return a list of objects form the search method of the REST API. E.g. A returned object could be a target or a disease

Parameters **query** (*str*) – string to search for

Keyword Arguments ****kwargs** – are passed as other parameters to the /public/search method of the REST API

Returns Result of the query

Return type *IterableResult*

2.4.2 Submodules

2.4.3 opentargets.conn module

This module abstracts the connection to the Open Targets REST API to simplify its usage. Can be used directly but requires some knowledge of the API.

class opentargets.conn.**Connection**(*host='https://platform-api.opentargets.io'*, *port=443*, *api_version='v3'*, *verify=True*, *proxies={}*, *auth=None*)
Bases: object

Handler for connection and calls to the Open Targets Validation Platform REST API

Parameters

- **host** (*str*) – host serving the API
- **port** (*int*) – port to use for connection to the API
- **api_version** (*str*) – api version to point to, default to 'latest'
- **verify** (*bool*) – sets SSL verification for Request session, accepts True, False or a path to a certificate
- **auth** (*AuthBase*) – sets the custom authentication object to use for requests made to the API. Should be one of the built in options provided by the reqests package, or a subclass of requests.auth.AuthBase.

api_endpoint_docs(*endpoint*)
Returns the documentation available for a given REST API endpoint

Parameters **endpoint** (*str*) – endpoint of the REST API

Returns documentation for the endpoint parsed from YAML docs

Return type dict

close()
Close connection to the REST API

get(*endpoint*, *params=None*)
makes a GET request :param endpoint: REST API endpoint to call :type endpoint: str :param params: request payload :type params: dict

Returns request response

Return type *Response*

get_api_endpoints()
Get a list of available endpoints

Returns available endpoints

Return type list

ping()
Pings the API as a live check :returns: True if pinging the raw response as a `str` if the API has a non standard name :rtype: bool

post(*endpoint*, *data=None*)
makes a POST request :param endpoint: REST API endpoint to call :type endpoint: str :param data: request payload :type data: dict

Returns request response

Return type *Response*

validate_parameter(*endpoint*, *filter_type*, *value*, *method='get'*)
Validate payload to send to the REST API based on info fetched from the API documentation

Parameters

- **endpoint** (*str*) – endpoint of the REST API
- **filter_type** (*str*) – the parameter sent for the request
- **value** – the value sent for the request
- **method** (HTTPMethods) – request method, either HTTPMethods.GET or HTTPMethods.POST. Defaults to HTTPMethods.GET

Raises AttributeError: if validation is not passed

class opentargets.conn.**HTTPMethods**
Bases: object

GET = 'get'

POST = 'post'

class opentargets.conn.**IterableResult**(*conn*, *method='get'*)
Bases: object

Proxy over the Connection class that allows to iterate over all the items returned from a quer. It will automatically handle making multiple calls for pagination if needed.

Requires a Connection :param conn: a Connection instance :type conn: Connection :param method: HTTP method to use for the calls :type method: HTTPMethods

filter(***kwargs*)
Applies a set of filters to the current query Keyword Args

**kwargs: passed to the REST API

Returns an IterableResult with applied filters

Return type *IterableResult*

to_csv(***kwargs*)
Create a csv file from a flattened version of the response.

Keyword Arguments ****kwargs** – forwarded to pandas.DataFrame.to_csv

Returns output of pandas.DataFrame.to_csv

Notes

Requires Pandas to be installed.

Raises `ImportError` – if Pandas is not available

to_dataframe(*compress_lists=False*, ***kwargs*)
Create a Pandas dataframe from a flattened version of the response.

Parameters **compress_lists** – if a value is a list, serialise it to a string with '|' as separator

Keyword Arguments ****kwargs** – forwarded to pandas.DataFrame.from_dict

Returns A DataFrame with all the data coming from the query in the REST API

Return type pandas.DataFrame

Notes

Requires Pandas to be installed.

Raises `ImportError` – if Pandas is not available

to_excel(*excel_writer*, ***kwargs*)
Create a excel (xls) file from a flattened version of the response.

Keyword Arguments ****kwargs** – forwarded to pandas.DataFrame.to_excel

Returns output of pandas.DataFrame.to_excel

Notes

Requires Pandas and xlwt to be installed.

Raises `ImportError` – if Pandas or xlwt are not available

to_file(*filename*, *compress=True*, *progress_bar=False*)

to_json(*iterable=True*, ***kwargs*)

Parameters **iterable** – If True will yield a json string for each result and convert them dinamically as they are fetched from the api. If False gets all the results and returns a singl json string.

Keyword Arguments ****kwargs** – forwarded to json.dumps

Returns an iterator of json strings or a single json string

to_object()

Converts dictionary in the data to an addict object. Useful for interactive data exploration on IPython and similar tools

Returns an iterator of addict.Dict

Return type iterator

class opentargets.conn.**IterableResultSimpleJSONEncoder**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: json.encoder.JSONEncoder

Constructor for JSONEncoder, with sensible defaults.

If skipkeys is false, then it is a TypeError to attempt encoding of keys that are not str, int, float or None. If skipkeys is True, such items are simply skipped.

If ensure_ascii is true, the output is guaranteed to be str objects with all incoming non-ASCII characters escaped. If ensure_ascii is false, the output can contain non-ASCII characters.

If check_circular is true, then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause an OverflowError). Otherwise, no such check takes place.

If allow_nan is true, then NaN, Infinity, and -Infinity will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a ValueError to encode such floats.

If sort_keys is true, then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If indent is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0 will only insert newlines. None is the most compact representation.

If specified, separators should be an (item_separator, key_separator) tuple. The default is (', ', ': ') if *indent* is None and (',', ': ') otherwise. To get the most compact JSON representation, you should specify (',', ':') to eliminate whitespace.

If specified, default is a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a TypeError.

default(*o*)

extends JsonEncoder to support IterableResult

class opentargets.conn.**Response**(*response*)

Bases: object

Handler for responses coming from the api

Parameters

- **response** – a response coming from a requests call
- **content_type** (*str*) – content type of the response

opentargets.conn.**compress_list_values**(*d*, *sep='|'*)

Parameters

- **d** (*dict*) – dictionary
- **sep** (*str*) – separator char used to join list element

Returns dictionary with compressed lists

Return type dict

opentargets.conn.**flatten**(*d*, *parent_key=''*, *separator='.'*)
Takes a nested dictionary as input and generate a flat one with keys separated by the separator

Parameters

- **d** (*dict*) – dictionary
- **parent_key** (*str*) – a prefix for all flattened keys
- **separator** (*str*) – separator between nested keys

Returns a flattened dictionary

Return type dict

2.4.4 opentargets.statistics module

class opentargets.statistics.**HarmonicSumScorer**(*buffer=100*)
Bases: object

An HarmonicSumScorer will ingest any number of numeric score, keep in memory the top max number defined by the buffer and calculate an harmonic sum of those :param buffer: number of element to keep in memory to compute the harmonic sum

add(*score*)
add a score to the pool of values :param score: a number to add to the pool ov values. is converted to float :type score: float

static harmonic_sum(*data*, *scale_factor=1*, *cap=None*)
Returns an harmonic sum for the data passed :param data: list of floats to compute the harmonic sum from :type data: list :param scale_factor: a scaling factor to multiply to each datapoint. Defaults to 1 :type scale_factor: float :param cap: if not None, never return an harmonic sum higher than the cap value. :type cap: float

Returns the harmonic sum of the data passed

Return type harmonic_sum (float)

refresh()
Store the minimum value of the pool

score(**args*, ***kwargs*)
Returns an harmonic sum for the pool of values :param *args: forwarded to HarmonicSumScorer.harmonic_sum

Keyword Args **kwargs: forwarded to HarmonicSumScorer.harmonic_sum

Returns the harmonic sum of the pool of values

Return type harmonic_sum (float)

2.5 Changelog

2.5.1 3.1.14

- two new endpoints through the client as get_target and get_disease

2.5.2 3.1.0

- added SSL ceritificate customisation
- added HTTP and SOCKS proxies support

2.5.3 3.0.0

Compatible with REST API release 3.0 - added *get_similar_target* and *get_similar_disease* methods - added ability to return json objects as addict Dictionaries - use next param to paginate when possible - retry on server side errors to make the connection more reliable

2.5.4 2.0.0

Compatible with REST API release 2.0 - added option to save the json query result to a local file - bugfixes

2.5.5 1.2.0

- added statistics module to allow score computation on subset of data
- improved fetching efficiency for big requests
- to_json method works with iterator pattern
- improved docs, and changed to Google Style
- set specific user agent for requests sent
- added methods to explore the REST API documentation and available endpoints
- added ping method to check for the api to be reachable

2.5.6 1.2.0b1

- Added option to export data as JSON, csv, excel or pandas dataframe. (optional dependency for pandas and xlwt)
- Fixed post calls and automatically switch to post for a large request

2.5.7 1.2.0a2

- Added filtering support to IterableResult
- Added parsing of swagger YAML for query validation

Minor:

- bugfixes
- improved tests
- improved docs

2.5.8 1.2.0a1

- First alpha release
- Compatible with Rest API 1.2.0

CHAPTER 3

Support

Open Targets Support (support@targetvalidation.org)

CHAPTER 4

Copyright

Python Module Index

o

Index

R

S

T

V