reference-sequence-fetcher Documentation

Release beta

Somesh Chaturvedi

Aug 06, 2018

Contents

reference-sequence-fetcher is a python client to retrieve sequence and metadata from reference servers in a user friendly and easy to use manner. It has requests library at its heart and allows you to fetch sequences using a Fetcher object.

An example would be like:

```
>>from reference_sequence_fetcher import fetcher
>>fethcer = Fetcher('yourserver.com')
>>fetcher.fetch_sequence('<checksum>')
>>ATGCATCGACTG.......ATGCATCGACTG
```

Fetcher object typically has two methods:

- *fetch_info* to fetch the service info of the server used in that particular Fetcher object
- *fetch_sequence* to fetch the sequence metadata
- *fetch_metadata* to fetch the metadata of the sequence

and three class methods *info*, *sequenece* and *metadata*

Note: reference-sequence-fetcher is stable only for Python3

CHAPTER 1

Installation

The recommended method for installing reference-sequence-fetcher is by using pip3.

1.1 Installing from PyPi

If you are just getting started with reference-sequence-fetcher, it is recommended that you start by installing the latest version from the Python Package Index (PyPi). To install reference-sequence-fetcher from PyPi using pip run the following command in your terminal.

```
pip install reference_sequence_fetcher
```

1.2 Installing from Source

You can install the latest development version of reference-sequence-fetcher directly from GitHub.

```
git clone https://github.com/someshchaturvedi/reference-sequence-fetcher.git
cd reference-sequence-fetcher
python setup.py sdist bdist_wininst upload
```

CHAPTER 2

Getting Started

The first thing you'll need to do to get started is install ChatterBot.

```
pip install reference_sequence_fetcher
```

See Installation for options for alternative installation methods.

2.1 Create a new Fetcher instance

```
from reference_sequence_fetcher import Fetcher
fetcher = Fetcher('<server_base_url>')
```

The only required parameter for the *Fetcher* is a server url. This will be the server which will be queried for fetching data

2.2 Fetch service info

After creating a new Fetcher instance you can use `fetch_info` to get the information regarding the server being qeried such as algorithms support, circular support, subsequence limit, supported API versions etc

```
print(fetcher.fetch_info()
```

2.3 Fetch sequence data

After creating a new Fetcher instance you can use `fetch_sequence` to get the sequence which has only one required parameter i.e. checksum. You can query for subsequence using start and end parameters. A detailed description will be given in the API documentaion section.

```
print(fetcher.fetch_sequence('<checksum>'))

print(fetcher.fetch_sequence('<checksum>', start=0, end=10))
```

2.4 Fetch metadata

You can use `fetch_metdata` to get the metadata associated with a sequence which has only one required parameter i.e. checksum. A detailed description will be given in the API documentaion section.

```
print(fetcher.fetch_metadata('<checksum>'))
```

CHAPTER 3

Retrieving sequences

3.1 Complete sequence query

Complete sequence can be retrieved using only the required parameter checksum in `fetch_sequence` function or `Fetcher.sequence` class method.

```
from reference_sequence_fetcher import fetcher
fetcher = Fetcher(<server_base_url>)
print(fetcher.fetch_sequence('<checksum>'))

print(Fetcher.sequence(<server_base_url>, <checksum>))
```

3.2 Sub-sequence query

User can also retrieve sub-sequences using start and end optional parameters

start and end used to retrieve sub-subsequence of specified bytes. Start is 0-start inclusive and end is exclusive. 0 <= start < length of sequence ; 1 <= end <= length of sequence

For ex sequence A is given as `ATGCATGCATGC` Length is 12

```
>>fetcher = Fetcher(<server_base_url>)
>>print(fetcher.fetch_sequence('<checksum>'), start=0, end=5)
>>ATGCA
```

If a server supports circular chromosomes, the client supports crossing the origin using the start and end parameters

```
>>fetcher = Fetcher(<server_base_url>)
>>print(fetcher.fetch_sequence('<checksum>'), start=5, end=2)
>>TGCATGCATG
```

so TGCATGC + ATG is the retrieved sequence.

Note: start and end can be used alone. If only start is given, end is assumed to be equal to length of the sequences. When only end is given start is assumed to be equal to 0.

CHAPTER 4

API Documentation

class Fetcher(*base_url*)

Act as a factory class for sequence and metadata

Parameters **base_url** (*string*) – Base url of the server from which data to be fetched

get_base_url()

Return base_url of the Fetcher object

Return type string

set_base_url(*base_url*)

Set base_url of the Fetcher object. Used to change the base url of already instantiated object on the fly.

Parameters **base_url** (*string*) – Base url of the server from which data to be fetched

Return type void

fetch_info()

Returns the service information of the server being queried

Return type dict

fetch_sequence(*checksum*[, *start=None*, *end=None*])

Act as factory method for retrieving sequences

Parameters

- **checksum** (*string*) – Checksum identifier of the sequence to be retrieved
- **start** (*integer*) – Used to define start location of the sequence to be retrieved (inclusive)
- **end** (*integer*) – Used to define end location of the sequence to be retrieved (exclusive)

Return type string

fetch_metadata(*checksum*)

Act as factory method for retrieving sequences

param checksum: Checksum identifier of the sequence to be retrieved :type checksum: string

Return type dict

classmethod info(*base_url*)
A class method for easily fetching single service info without creating Fetcher object.

Return type dict

classmethod sequence(*base_url*, *checksum*[, *start=None*, *end=None*])
A class method for easily fetching single sequence without creating Fetcher object. Parameters definitions as per defined above

Return type string

classmethod metdata(*base_url*, *checksum*)
A class method for easily fetching single metadata without creating Fetcher object. Parameters definitions as per defined above

Return type dict

CHAPTER 5

Examples

Under development :)

CHAPTER 6

Command Line Interface

To use command line interface, first install the package using pip3. You can then access Fetcher functions using `ref-seq-fetcher`

```
ref-seq-fetcher --help

ref-seq-fetcher [OPTIONS] COMMAND [ARGS]...

Options:
--help  Show this message and exit.

Commands:
metadata  retrieve metadata using base_url and checksum
sequence  retrieve sequence using base_url and checksum
```

6.1 To retrieve service-info

```
ref-seq-fetcher info --help

info [OPTIONS] BASE_URL

retrieve service info using base_url

Options:
--help  Show this message and exit.
```

6.1.1 Examples

```
ref-seq-fetcher info https://www.ebi.ac.uk/ena/cram/

{"service": {"circular_supported": "true", "subsequence_limit": 4000000, "algorithms
↪": ["md5", "trunc512"], "supported_api_versions": ["1.0"]}}
```

6.2 To retrieve a sequence

```
ref-seq-fetcher sequence --help

sequence [OPTIONS] BASE_URL CHECKSUM

retrieve sequence using base_url and checksum

Options:
 -s, --start INTEGER   first byte of the checksum. 0-start inclusive
 -e, --end INTEGER     last byte of the checksum. 0-start exclusive
 --help                Show this message and exit.
```

server_base_url and checksum are required arguments while rest are optional. It internally invokes classmethod sequence of Fetcher.

6.2.1 Examples

```
ref-seq-fetcher sequence https://www.ebi.ac.uk/ena/cram/␣
↪6681ac2f62509cfc220d78751b8dc524
CCACA........GTGGG
```

```
ref-seq-fetcher sequence https://www.ebi.ac.uk/ena/cram/␣
↪6681ac2f62509cfc220d78751b8dc524 --start 10 --end 20
CCCACACACC
```

```
ref-seq-fetcher sequence https://www.ebi.ac.uk/ena/cram/␣
↪3332ed720ac7eaa9b3655c06f6b9e196 -s 5374 -e 5
ATCCAACCTGCAGAGTT
```

6.3 To retrieve a metadata

```
ref-seq-fetcher metadata --help

retrieve metadata using base_url and checksum

Options:
  --help                Show this message and exit.
```

server_base_url and checksum are required arguments while rest are optional. It internally invokes classmethod metadata of Fetcher.

6.3.1 Examples

```
ref-seq-fetcher metadata https://www.ebi.ac.uk/ena/cram/␣
↪3332ed720ac7eaa9b3655c06f6b9e196 > metadata.json
```

CHAPTER 7

Development Guide

7.1 Contributing to reference-sequence-fetcherdoc

There are numerous ways to contriubte to reference-sequence-fetcher. All of which are highly encouraged.

- Contributing bug reports and feature requests
- Contributing documentation
- Contributing code for new features
- Contributing fixes for bugs

Every bit of help received on this project is highly appreciated.

7.1.1 Setting Up a Development Environment

To contribute to reference-sequence-fetcher's development, you simply need:

- Python3
- pip3
- A few python packages:

```
pip3 install requirements.txt
```

- A text editor

7.1.2 Reporting a Bug

If you discover a bug in reference-sequence-fetcher and wish to report it, please be sure that you adhere to the following when you report it on GitHub.

1. Before creating a new bug report, please search to see if an open or closed report matching yours already exists.

2. Please include a description that will allow others to recreate the problem you encountered.

7.1.3 Contributing Documentation

reference-sequence-fetcher's documentation is written in reStructuredText and is compiled by Sphinx. The reStructuredText source of the documentation is located in `docs/`.

To build the documentation yourself, run:

```
cd docs
make html
```

You can then open the index.html file that will be created in the build directory.

7.1.4 Contributing Code

The development of reference-sequence-fetcher happens on GitHub. Code contributions should be submitted there in the form of pull requests.

Pull requests should meet the following criteria.

1. Fix one issue and fix it well.
2. Do not include extraneous changes that do not relate to the issue being fixed.
3. Include a descriptive title and description for the pull request.
4. Have descriptive commit messages.

7.2 Testing

To run the tests clone the repository and run the following command

```
py.test
```

CHAPTER 8

Report an Issue

Please direct all bug reports and feature requests to the project's issue tracker on GitHub.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Index