jpredapi Documentation

Release 1.5.0

Andrey Smelter

Apr 14, 2017

Contents

CHAPTER 1

jpredapi

The *jpredapi* package provides a simple Python interface for submitting and retrieving jobs from JPRED: A Protein Secondary Structure Prediction Server (JPRED).

This is unofficial Python port of *jpredapi* perl script that can be found here: http://www.compbio.dundee.ac.uk/jpred4/api.shtml

Links

- jpredapi @ GitHub
- jpredapi @ PyPI
- Documentation @ ReadTheDocs

Installation

Install on Linux, Mac OS X

```
python3 -m pip install jpredapi
```

Install on Windows

```
py -3 -m pip install jpredapi
```

Note: Read the User Guide and The jpredapi Tutorial on ReadTheDocs to learn more and to see code examples on using the *nmrstarlib* as a library and as a command-line tool.

Contents:

User Guide

Description

The *jpredapi* package provides a simple Python interface for submitting and retrieving jobs from JPRED: A Protein Secondary Structure Prediction Server (JPRED).

Installation

The *jpredapi* package runs under Python 2.7 and Python 3.4+. Starting with Python 3.4 pip is included by default. To install system-wide with pip run the following:

Install on Linux, Mac OS X

```
python3 -m pip install jpredapi
```

Install on Windows

```
py -3 -m pip install jpredapi
```

Install inside virtualenv

For an isolated install, you can run the same inside a virtualenv.

```
$ virtualenv -p /usr/bin/python3 venv  # create virtual environment, use python3␣
↪interpreter

$ source venv/bin/activate             # activate virtual environment

$ python3 -m pip install jpredapi      # install jpredapi as usually

$ deactivate                           # if you are done working in the virtual␣
↪environment
```

Dependencies

jpredapi depends on several Python libraries, it will install its dependencies automatically, but if you wish to install them manually, then run commands below:

- **docopt for creating *jpredapi* command-line interface.**
  - To install docopt run the following:

```
python3 -m pip install docopt  # On Linux, Mac OS X
py -3 -m pip install docopt    # On Windows
```

- **requests for sending HTTP/1.1 requesrs to JPRED server.**
  - To install requests Python library run the following:

```
python3 -m pip install requests  # On Linux, Mac OS X
py -3 -m pip install requests    # On Windows
```

- **retrying for controlling status requests.**
  - To install retrying Python library run the following:

```
python3 -m pip install retrying  # On Linux, Mac OS X
py -3 -m pip install retrying    # On Windows
```

Basic usage

jpredapi can be used in several ways:

- As a library within interactive Python shell or Python script and as a command-line tool to:
  - Submit JPRED job.
  - Check status of JPRED job.
  - Retrieve results of JPRED job.

Note: Read *The jpredapi Tutorial* to learn more and see code examples on using `jpred`.

The jpredapi Tutorial

The *jpredapi* package provides functions to submit, check status, and retrieve results from JPRED: A Secondary Structure Prediction Server.

Command Line Interface

```
jpredapi command-line interface

The RESTful API allows JPred users to submit jobs from the command-line.

Usage:
    jpredapi -h | --help
    jpredapi --version
    jpredapi submit (--mode=<mode> --format=<format>) (--file=<filename> | --seq=
↪<sequence>)
                    [--email=<name@domain.com>] [--name=<job_name>] [--skipPDB=<value>
↪]
                    [--rest=<address>] [--jpred4=<address>] [--silent]
    jpredapi status (--job_id=<id>) [--results_dir=<path>]
                    [--wait_interval=<interval>] [--extract] [--silent]
    jpredapi get_results (--job_id=<id>) [--results_dir=<path>]
                         [--wait_interval=<interval>] [--extract] [--silent]
    jpredapi quota (--email=<name@domain.com>)

Options:
```

```
    -h, --help                    Show this help message.
    --version                     Show jpredapi version.
    --silent                      Do not print messages.
    --extract                     Extract results tar.gz archive into folder.
    --mode=<mode>                 Submission mode, possible values: single, batch, msa.
    --format=<format>             Submission format, possible values: raw, fasta, msf,␣
↪blc.
    --file=<filename>             Filename of a file with the job input (sequence(s)).
    --seq=<sequence>              Instead of passing input file, for single-sequence␣
↪submission.
    --email=<name@domain.com>     E-mail address where job report will be sent
                                  (optional for all but batch submissions).
    --name=<job_name>             Job name.
    --job_id=<job_id>             Job id.
    --skipPDB=<value>             PDB check, possible values: True, False [default:␣
↪True].
    --results_dir=<path>          Path where to save archive with results.
    --rest=<address>              REST address of server
                                  [default: http://www.compbio.dundee.ac.uk/jpred4/cgi-
↪bin/rest].
    --jpred4=<address>            Address of Jpred4 server
                                  [default: http://www.compbio.dundee.ac.uk/jpred4].
    --wait_interval=<interval>    Wait interval before retrying to check job status in␣
↪seconds
                                  [default: 60].
```

Print jpredapi help message

```
$ python3 -m jpredapi --help
```

Print jpredapi version

```
$ python3 -m jpredapi --version
```

Submit jobs to JPRED server

Submit single sequence in `raw` format using `--seq` parameter:

```
python3 -m jpredapi submit --mode=single --format=raw --
↪seq=MQVWPIEGIKKFETLSYLPP
```

Submit single sequence in `raw` format using `--file` parameter:

```
python3 -m jpredapi submit --mode=single --format=raw --file=tests/example_
↪data/single_raw.example
```

Content of `single_raw.example` file:

```
MQVWPIEGIKKFETLSYLPPLTVEDLLKQIEYLLRSKWVPCLEFSKVGFVYRENHRSPGYYDGRYWTMWKLPMFGCTDATQVLKELEEAKKAYPDA
```

Submit single sequence in `fasta` format using `--file` parameter:

```
python3 -m jpredapi submit --mode=single --format=fasta --file=tests/example_
↪data/single_raw.example
```

Content of `single_fasta.example` file:

```
>my test sequence
MQVWPIEGIKKFETLSYLPPLTVEDLLKQIEYLLRSKWVPCLEFSKVGFVYRENHRSPGYYDGRYWTMWKLPMFGCTDATQVLKELEEAKKAYPDA
```

Submit multiple sequences in `fasta` format using `--file` parameter:

```
python3 -m jpredapi submit --mode=batch --format=fasta --file=tests/example_
↪data/batch_fasta.example --email=name@domain.com
```

Content of `batch_fasta.example` file:

```
>my_seq1
MKFLVLLFNILCLFPILGADELVMSPIPTTDVQPKVTFDINSEVSSGPLYLNPVEMAGVK
YLQLQRQPGVQVHKVVEGDIVIWENEEMPLYTCAIVTQNEVPYMAYVELLEDPDLIFFLK
EGDQWAPIPEDQYLARLQQLRQQIHTESFFSLNLSFQHENYKYEMVSSFQHSIKMVVFTP
KNGHICKMVYDKNIRIFKALYNEYVTSVIGFFRGLKLLLLNIFVIDDRGMIGNKYFQLLD
DKYAPISVQGYVATIPKLKDFAEPYHPIILDISDIDYVNFYLGDATYHDPGFKIVPKTPQ
CITKVVDGNEVIYESSNPSVECVYKVTYYDKKNESMLRLDLNHSPPSYTSYYAKREGVWV
TSTYIDLEEKIEELQDHRSTELDVMFMSDKDLNVVPLTNGNLEYFMVTPKPHRDIIIVFD
GSEVLWYYEGLENHLVCTWIYVTEGAPRLVHLRVKDRIPQNTDIYMVKFGEYWVRISKTQ
>my_seq2
MASVKSSSSSSSSSFISLLLLILLVIVLQSQVIECQPQQSCTASLTGLNVCAPFLVPGSP
TASTECCNAVQSINHDCMCNTMRIAAQIPAQCNLPPLSCSAN
>my_seq3
MEKKSIAGLCFLFLVLFVAQEVVVQSEAKTCENLVDTYRGPCFTTGSCDDHCKNKEHLLS
GRCRDDVRCWCTRNC
```

Submit multiple sequence alignment files in `fasta` format:

```
python3 -m jpredapi submit --mode=msa --format=fasta --file=tests/example_
↪data/msa_fasta.example --email=name@domain.com
```

Content of `msa_fasta.example` file:

```
>QUERY_1
MQVWPIEGIKKFETLSYLPPLTVEDLLKQIEYLLRSKWVPCLEFSKVGFVYRENHRSPGYYDGRYWTMWKLP
MFGCTDATQVLKELEEAKKAYPDAFVRIIGFDNVRQVQLISFIAYKPPGC
>UniRef90_Q40250_2
MKVWPPIGLKKYETLSYLPPLSDEALSKEIDYLIRNKWIPCLEFEEHGFVYREHHHSPGYYDGRYWTMWKLP
MFGCTDSAQVMKEVGECKKEYPNAFIRVIGFDNIRQVQCISFIVAKPPGV
>UniRef90_A7YVW5_3
MQVWPPLGKRKFETLSYLPPLPVDALLKQIDYLIRSGWIPCIEFTVEGFVYREHHHSPGYYDGRYWTMWKLP
MYGCTDSTQVLAEVEANKKEYPNSYIRIIGFDNKRQVQCVSFIVHTPPS-
```

```
>UniRef90_P04714_4
MQVWPPYGKKKYETLSYLPDLTDEQLLKEIEYLLNKGWVPCLEFTEHGFVYREYHASPRYYDGRYWTMWKLP
MFGCTDATQVLGELQEAKKAYPNAWIRIIGFDNVRQVQCISFIAYKPPG-
>UniRef90_W9RUU9_5
MQVWPPRGKLKFETLSYLPDLTDEQLLKEIDYLLRSNWIPCLEFEVKAHIYRENNRSPGYYDGRYWTMWKLP
MFGCTDATQVLAEVQETKKAYPDAHVRIIGFDNNRQVQCISFIAYKPPA-
```

Submit multiple sequence alignment files in `msf` format:

```
python3 -m jpredapi submit --mode=msa --format=msf --file=tests/example_data/
↪msa_msf.example --email=name@domain.com
```

Content of `msa_msf.example` file:

```
/tmp/filelPdICy  MSF: 108  Type: N  January 01, 1776  12:00  Check: 2741 ..

Name: 0_1a             Len:   108  Check:  4063  Weight:  1.00
Name: 1_MA             Len:   108  Check:  4875  Weight:  1.00
Name: 2_KE             Len:   108  Check:   449  Weight:  1.00
Name: 3_NC             Len:   108  Check:  3354  Weight:  1.00

//

           0_1a  APAFSVSPAS GASDGQSVSV SVAAAGETYY IAQCAPVGGQ DACNPATATS
           1_MA  APGVTVTPAT GLSNGQTVTV SATTPGTVYH VGQCAVVEGV IGCDATTSTD
           2_KE  SAAVSVSPAT GLADGATVTV SASATSTSAT ALQCAILAGR GACNVAEFHD
           3_NC  APTATVTPSS GLSDGTVVKV AGAQAGTAYD VGQCAWVDGV LACNPADFSS

           0_1a  FTTDASGAAS FSFTVRKSYA GQTPSGTPVG SVDCATDACN LGAGNSGLNL
           1_MA  VTADAAGKIT AQLKVHSSFQ AVVANGTPWG TVNCKVVSCS AGLGSDSGEG
           2_KE  FSLSG.GEGT TSVVVRRSFT GYVPDGPEVG AVDCDTAPCE IVVGGNTGEY
           3_NC  VTADANGSAS TSLTVRRSFE GFLFDGTRWG TVDCTTAACQ VGLSDAAGNG

           0_1a  GHVALTFG
           1_MA  AAQAITFA
           2_KE  GNAAISFG
           3_NC  PGVAISFN
```

Submit multiple sequence alignment files in `blc` format:

```
python3 -m jpredapi submit --mode=msa --format=blc --file=tests/example_data/
↪msa_blc.example --email=name@domain.com
```

Content of `msa_blc.example` file:

```
>0_1a  Name
>1_MA  Name
>2_KE  Name
>3_NC  Name
* iteration 1
AASA
PPAP
AGAT
```

```
FVVA
STST
VVVV
STST
PPPP
AAAS
*
```

Check job status on JPRED server

Check single job status using `job_id`:

```
python3 -m jpredapi status --job_id=jp_K46D05A
```

Check single job status using `job_id` and retrieve results:

```
python3 -m jpredapi status --job_id=jp_K46D05A --results_dir=jpred_sspred/
↪results
```

Check single job status using `job_id`, retrieve results, and decompress archive:

```
python3 -m jpredapi status --job_id=jp_K46D05A --results_dir=jpred_sspred/
↪results --extract
```

Retrieve results from JPRED server

Retrieve results using `job_id`:

```
python3 -m jpredapi get_results --job_id=jp_K46D05A --results_dir=jpred_
↪sspred/results
```

Retrieve results using `job_id` and decompress archive:

```
python3 -m jpredapi get_results --job_id=jp_K46D05A --results_dir=jpred_
↪sspred/results --extract
```

Check how many jobs you have already submitted on a given day:

```
python3 -m jpredapi quota --email=name@domain.com
```

Using jpredapi as a library

Importing jpredapi module

If *jpredapi* package is installed on the system, it can be imported:

```
>>> import jpredapi
```

Submit jobs to JPRED server

Submit single sequence in `raw` format using `seq` parameter:

```
>>> import jpredapi
>>>
>>> jpredapi.submit(mode="single", user_format="raw", seq="MQVWPIEGIKKFETLSYLPP")
>>>
```

Submit single sequence in `raw` format using `file` parameter:

```
>>> jpredapi.submit(mode="single", user_format="raw", file="tests/example_data/single_
↪raw.example")
>>>
```

Submit single sequence in `fasta` format using `file` parameter:

```
>>> jpredapi.submit(mode="single", user_format="fasta", file="tests/example_data/
↪single_fasta.example")
>>>
```

Submit multiple sequences in `fasta` format using `file` parameter:

```
>>> jpredapi.submit(mode="batch", user_format="fasta", file="tests/example_data/batch_
↪fasta.example", email="name@domain.com")
>>>
```

Submit multiple sequence alignment files in `fasta` format:

```
>>> jpredapi.submit(mode="msa", user_format="fasta", file="tests/example_data/msa_
↪fasta.example", email="name@domain.com")
>>>
```

Submit multiple sequence alignment files in `msf` format:

```
>>> jpredapi.submit(mode="msa", user_format="msf", file="tests/example_data/msa_msf.
↪example", email="name@domain.com")
>>>
```

Submit multiple sequence alignment files in `blc` format:

```
>>> jpredapi.submit(mode="msa", user_format="blc", file="tests/example_data/msa_blc.
↪example", email="name@domain.com")
>>>
```

Check job status on JPRED server

Check single job status using `job_id`:

```
>>> import jpredapi
>>>
>>> jpredapi.status(job_id="jp_K46D05A")
>>>
```

Check single job status using `job_id` and retrieve results:

```
>>> jpredapi.status(job_id="jp_K46D05A", results_dir_path="jpred_sspred/results")
>>>
```

Check single job status using `job_id`, retrieve results, and decompress archive:

```
>>> jpredapi.status(job_id="jp_K46D05A", results_dir_path="jpred_sspred/results",␣
↪extract=True)
>>>
```

Retrieve results from JPRED server

Retrieve results using `job_id`:

```
>>> import jpredapi
>>>
>>> jpredapi.get_results(job_id="jp_K46D05A", results_dir_path="jpred_sspred/results")
>>>
```

Retrieve results using `job_id` and decompress archive:

```
>>> jpredapi.get_results(job_id="jp_K46D05A", results_dir_path="jpred_sspred/results",
↪ extract=True)
>>>
```

Check how many jobs you have already submitted on a given day:

```
>>> import jpredapi
>>>
>>> jpredapi.quota(email="name@domain.com")
>>>
```

The jpredapi API Reference

jpredapi Python library

The RESTful API allows JPred users to submit jobs from the command-line.

Usage example for command-line:

```
python3 -m jpredapi --help
python3 -m jpredapi --version
python3 -m jpredapi submit --mode=single --format=raw --
↪seq=MQVWPIEGIKKFETLSYLPP
python3 -m jpredapi status --job_id=jp_K46D05A
python3 -m jpredapi get_results --job_id=jp_K46D05A --results_dir=jpred_
↪sspred/results
python3 -m jpredapi quota --email=name@domain.com
```

Usage example for interactive Python shell:

```
>>> import jpredapi
>>>
>>> jpredapi.submit(mode="single", user_format="raw", seq="MQVWPIEGIKKFETLSYLPP")
>>>
>>> jpredapi.status(job_id="jp_K46D05A")
>>>
>>> jpredapi.get_results(job_id="jp_K46D05A", results_dir_path="jpred_sspred/results")
>>>
>>> jpredapi.quota(email="name@domain.com")
>>>
```

jpredapi.api.**submit**(*mode*, *user_format*, *file=None*, *seq=None*, *skipPDB=True*, *email=None*, *name=None*, *silent=False*)

Submit job to Jpred server.

Parameters

- **mode** (*str*) – Submission mode, possible values: *single*, *batch*, *msa*.
- **user_format** (*str*) – Submission format, possible values: *raw*, *fasta*, *msf*, *blc*.
- **file** (*str*) – Filename of a file with the job input (sequence or msa).

- **seq** (*str*) – Amino acid sequence passed as string of single-letter code without spaces, e.g. –seq=ATWFGTHY
- **skipPDB** (`True` or `False`) – PDB check will not be performed (True), otherwise perform PDB check (False).
- **email** (*str*) – E-mail address.
- **name** (*str*) – Job name.
- **silent** (`True` or `False`) – Print information about job submission.

Returns None

Return type `None`

`jpredapi.api.`**`status`**(**args*, ***kw*)
Check status of submitted job.

Parameters

- **job_id** (*str*) – Job id.
- **results_dir_path** (*str*) – Directory path where to save results if job is finished.
- **extract** (`True` or `False`) – Extract (True) or not (False) results into directory.
- **silent** (`True` or `False`) – Print information about job status.

Returns None

Return type `None`

`jpredapi.api.`**`get_results`**(*job_id*, *results_dir_path=None*, *extract=False*, *silent=False*)
Download results from Jpred server.

Parameters

- **job_id** (*str*) – Job id.
- **results_dir_path** (*str*) – Path where to save results.
- **extract** (`True` or `False`) – Extract from tar.gz archive.
- **silent** (`True` or `False`) – Print information.

Returns None

Return type `None`

`jpredapi.api.`**`quota`**(*email*, *host='http://www.compbio.dundee.ac.uk/jpred4/cgi-bin/rest'*, *suffix='quota'*)
Check how many jobs you have already submitted on a given day (out of 1000 maximum allowed jobs per user per day).

Parameters

- **email** (*str*) – E-mail address.
- **host** (*str*) – Jpred host address.
- **suffix** (*str*) – Host address suffix.

Returns None

Return type `None`

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

j

Index