
pyjulius Documentation

Release 0.3

Antoine Bertin

October 17, 2014

1	Example	3
2	API Documentation	5
2.1	States	5
2.2	Client	5
2.3	Models	6
2.4	Exceptions	7
	Python Module Index	9

Release v0.3

pyjulius provides a simple interface to connect to julius module server

Example

First you will need to run `julius` with the `-module` option (documentation [here](#) or `man julius`). Julius will wait for a client to connect, this is what `Client` does in a threaded way.

Let's just write a simple program that will print whatever the `julius` server sends until you press CTRL+C:

```
#!/usr/bin/env python
import sys
import pyjulius
import Queue

# Initialize and try to connect
client = pyjulius.Client('localhost', 10500)
try:
    client.connect()
except pyjulius.ConnectionError:
    print 'Start julius as module first!'
    sys.exit(1)

# Start listening to the server
client.start()
try:
    while 1:
        try:
            result = client.results.get(False)
        except Queue.Empty:
            continue
        print repr(result)
except KeyboardInterrupt:
    print 'Exiting...'
    client.stop() # send the stop signal
    client.join() # wait for the thread to die
    client.disconnect() # disconnect from julius
```

If you are only interested in recognitions, wait for an instance of `Sentence` objects in the queue:

```
if isinstance(result, pyjulius.Sentence):
    print 'Sentence "%s" recognized with score %.2f' % (result, result.score)
```

If you do not want `Client` to interpret the raw xml `Element`, you can set `modelize` attribute to `False`

If you encounter any encoding issues, have a look at the `-charconv` option of `julius` and set the `Client.encoding` to the right value

API Documentation

More details about the use of the module can be found here

2.1 States

`pyjulius.core.CONNECTED = 1`
Connected client state

`pyjulius.core.DISCONNECTED = 2`
Disconnected client state

2.2 Client

class `pyjulius.core.Client` (*host='localhost', port=10500, encoding='utf-8', modelize=True*)
Threaded Client to connect to a julius module server

Parameters

- **host** (*string*) – host of the server
- **port** (*integer*) – port of the server
- **encoding** (*string*) – encoding to use to decode socket's output
- **modelize** (*boolean*) – try to interpret raw `xml Element` as `models` if `True`

host
Host of the server

port
Port of the server

encoding
Encoding to use to decode socket's output

modelize
Try to interpret raw `xml Element` as `models` if `True`

results
Results received when listening to the server. This `Queue` is filled with raw `xml Element` objects and `models` (if `modelize`)

sock
The socket used

state
Current state. State can be:

- `CONNECTED`
- `DISCONNECTED`

connect ()
Connect to the server

Raises ConnectionError If socket cannot establish a connection

disconnect ()
Disconnect from the server

run ()
Start listening to the server

send (command, timeout=5)
Send a command to the server

Parameters command (string) – command to send

stop ()
Stop the thread

2.3 Models

Models are designed in order to represent the server response an object-oriented and easy way

class `pyjulius.models.Sentence (words, score=0)`
A recognized sentence

Parameters

- **words** (list of `Word`) – words in the sentence
- **score** (*integer*) – score of the sentence

words
Words that constitute the sentence

score
Score of the sentence

classmethod from_shypo (xml, encoding='utf-8')
Constructor from xml element *SHYPO*

Parameters

- **xml** (*xml.etree.ElementTree*) – the xml *SHYPO* element
- **encoding** (*string*) – encoding of the xml

class `pyjulius.models.Word (word, confidence=0.0)`
A word within a *Sentence*

Parameters

- **word** (*string*) – the word

- **confidence** (*float*) – confidence of the recognized word

word

Recognized word

confidence

Confidence of the recognized word

classmethod **from_whyfo** (*xml*, *encoding='utf-8'*)

Constructor from xml element *WHYFO*

Parameters

- **xml** (*xml.etree.ElementTree*) – the xml *WHYFO* element
- **encoding** (*string*) – encoding of the xml

2.4 Exceptions

exception `pyjulius.exceptions.Error`

Base class for pyjulius exceptions

exception `pyjulius.exceptions.ConnectionError`

Raised when the initial connection to the server could not be established

exception `pyjulius.exceptions.SendTimeoutError`

Raised when could not send the command (timeout)

p

`pyjulius.exceptions`, 7
`pyjulius.models`, 6

C

Client (class in pyjulius.core), 5
confidence (pyjulius.models.Word attribute), 7
connect() (pyjulius.core.Client method), 6
CONNECTED (in module pyjulius.core), 5
ConnectionError, 7

D

disconnect() (pyjulius.core.Client method), 6
DISCONNECTED (in module pyjulius.core), 5

E

encoding (Client attribute), 5
Error, 7

F

from_shypo() (pyjulius.models.Sentence class method), 6
from_whypo() (pyjulius.models.Word class method), 7

H

host (Client attribute), 5

M

modelize (Client attribute), 5

P

port (Client attribute), 5
pyjulius.exceptions (module), 7
pyjulius.models (module), 6

R

results (Client attribute), 5
run() (pyjulius.core.Client method), 6

S

score (pyjulius.models.Sentence attribute), 6
send() (pyjulius.core.Client method), 6
SendTimeoutError, 7
Sentence (class in pyjulius.models), 6

sock (Client attribute), 5
state (Client attribute), 6
stop() (pyjulius.core.Client method), 6

W

Word (class in pyjulius.models), 6
word (pyjulius.models.Word attribute), 7
words (pyjulius.models.Sentence attribute), 6