pyWebGraph Documentation

Release 0.1a

Massimo Santini

June 01, 2012

CONTENTS

This package is intended as a simple bridge easing the usage of the WebGraph library in Python.

This software is hosted at http://code.google.com/p/py-web-graph/.

Given that the library is written in Java, some modules of this package must be run with Jython, or a similar Python interpreter able to execute Java code. To reduce the impact of such dependency, a basic XML-RPC server is provided to allow any other Python interpreter to access WebGraph data.

Finally, the package contains a simple wrapper for Ubigraph (a free tool for visualizing dynamic graphs) is provided, allowing some visual exploration of WebGraph data.

This documentation includes:

CHAPTER

ONE

QUICKSTART DOCUMENTS

The following documents give a very short introduction to the main aspects of the package and, alebit far from being complete, should be enough to kickstart you.

1.1 The pyWebGraph console

The pyWebGraph console is meant as a tool to explore WebGrph data. A sample session could be:

```
$ jython -m pywebgraph.console
Welcome to pyWebGraph console!
>> graph example
>> pwn
#0
>> ls
0: #1
1: #342
...
>> namemaps example
>> pwn
#0 http://acireale.src.cnr.it/
>> ls #0
0: #1 http://acireale.src.cnr.it/acireale2001/
1: #342 http://acireale.src.cnr.it/htm/diretta.htm
...
>> cn "http://acireale.src.cnr.it/acireale2001/"
```

First of all you load a graph with the `graph` command, then you can explore it (much like you would do with your filesystem): using `pwn` and `cn` (to print and change, respectively, the working node) and using `ls` to list outlinks. Additionaly, with `namemaps`, you can load a map from node numbers to their names (and *vice versa* if possible) that will be used when printing (and identifying) nodes.

1.1.1 Node specifications

A node can be specified to commands of the console in three ways:

- as an absolute node number (a non negative integer prefixed by `#`),
- as a node name (any string enclosed in double quotes),
- as a path (relative to the current working node).

The *current working node* is a given node of the graph. A *path* is a sequence of integers separated by `/` (leading, trailing, and repeated separators are ignored); numbers in the sequence refer to outlinks (or inlinks, if prefixed by a `-`, in any case, numbered starting from 0). For example, path `2/4` referes to the fifth outlink of the third outlink of the current node; similarly, path `3/-1/5` refers to the sixth outlink af the second inlink of the fourth outlink of the current working node.

Note: Since web graphs need not to be connected, it is not true in general that every node can be specified with a path relative to some specific (so to say "root") node, hence the notion of an *absolute* path makes little sense here. This navigational methaphor hence differs from the filesystem case, corresponding to a (of course connected) tree.

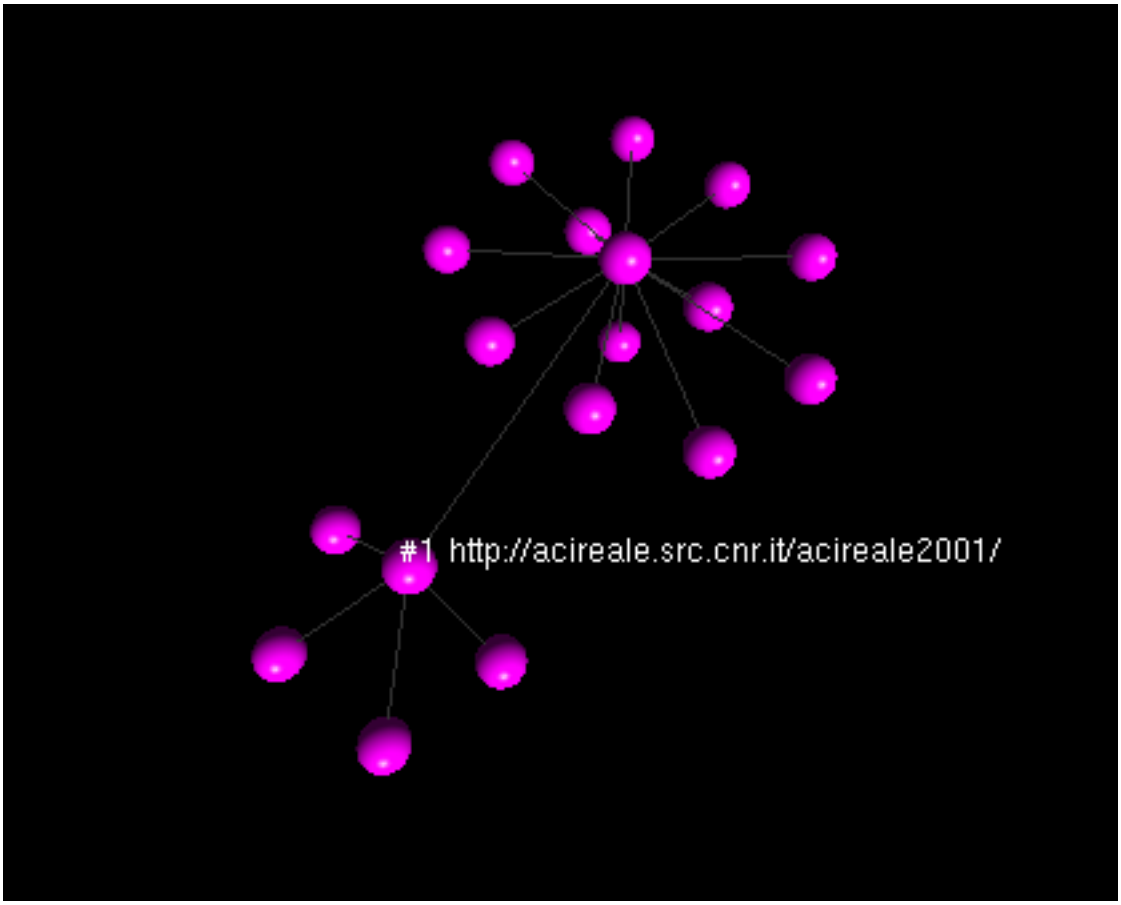
1.1.2 Visualization

If the console is given the `-r/--renderer` command line option pointing to a Ubigraph server, then the console can be used to visualize WebGraph data using the commands `bfs`, `isg`, `hl`, `label`, and `clear`; please use the `help` console command for details.

A sample session could be:

```
$ jython -m pywebgraph.console -r-
Welcome to pyWebGraph console!
>> graph example
>> namemaps example
>> bfs 0
>> bfs 1
>> label 0 on
```

This could give an output like



1.1.3 XML-RPC Server

It is likely that one wants to use this library with a standard Python interpreter (say one that supports the Python 2.6 version). This is made possible by limiting the need of a Java capable interpreter (like Jython that supports only the Python 2.5 vesrion) to the code accessing WebGraph, allowing any other interpreter to access the data via XML-RPC.

Two low level components are needed: a server (implemented in module `pywebgraph.webgraph.server`) exposing the data and a client (implemented in `pywebgraph.webgraph.client`) able to access it.

Such mechanism is made available the console via the `-g/--graph-server` command line option (and thanks to the fact that `pywebgraph.webgraph.server` is also executable).

A sample session could be:

```
$ jython -m pywebgraph.webgraph.server &
Listening on port 8000
$ python2.6 -m pywebgraph.console -g-
Welcome to pyWebGraph console!
>> graph example
>> ls
0: #1
1: #342
...
```

1.2 Integration with Ubigraph

pyWebGraph can use Ubigraph to visualize WebGraph data. To this aim, three possibilities are provided:

- the pyWebGraph *console* (that is, the `pywebgraph.console` executable module),
- the `pywebgraph.examples.explore` executable module,
- the `pywebgraph.ubigraph` module.

1.3 The pyWebGraph XML-RPC server

CHAPTER

TWO

PACKAGE DOCUMENTATION

2.1 pywebgraph.console — A simple CLI (and GUI) explorer

This module is executable. It supports the following options

-r

-rendered
The address of the Ubigraph server (if equals to -, it will be expanded to the default value).

-g

-graph-server
The address of the XML-RPC pyWebGraph graph server (if equals to -, it will be expanded to the default value).

See Also:

The pyWebGraph console In the *Quickstart documents* secetion.

2.2 pywebgraph.ubigraph — Access to Ubigraph visualization server

class pywebgraph.ubigraph.**Renderer**([*address*])

ADDRESS = 'http://127.0.0.1:20738/RPC2'
The default Ubigraph XML-RPC server address: *http://127.0.0.1:20738/RPC2*

addcallback(*callback_url*)

addedge(*from_node*, *to_node*, *imply=True*)

addnode(*node*)

clear()

highlight(*node*)

label(*node*, *label*)

unlabel(*node*)

2.3 pywebgraph.webgraph — Access to WebGraph data from Python

pywebgraph.webgraph.**new_local_graph**()

pywebgraph.webgraph.**new_remote_graph**([*address*])

2.4 pywebgraph.webgraph.local — A Graph class to access WebGraph data

class pywebgraph.webgraph.local.**Graph**

current_node

get_current_node()

get_num_nodes()

inlinks(*node*)

load_graph(**args*, ***kwargs*)

load_name_map(**args*, ***kwargs*)

name_to_node(*name*)

node_to_name(*node*)

node_tos(*node*)

num_nodes

outlinks(*node*)

resolve(*node_spec*)

set_current_node(*node*)

2.5 pywebgraph.webgraph.server — An XML-RPC server exposing WebGraph data

class pywebgraph.webgraph.server.**Graph**([*port*])

PORT = 8000

The default pyWebGraph XML-RPC server port: *8000*

2.6 pywebgraph.webgraph.client — An XML-RPC client consuming WebGraph data

class pywebgraph.webgraph.client.**Graph**([*address*])

ADDRESS = 'http://127.0.0.1:8000/'
The default pyWebGraph XML-RPC server address: *http://127.0.0.1:8000/*

current_node

get_current_node()

get_num_nodes()

node_tos(*node*)

num_nodes

set_current_node(*node*)

2.7 pywebgraph.examples.explore — A simple GUI interactive explorer

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

p