IPython Cypher Documentation

Release 1.0.0

Javier de la Rosa

December 11, 2016

Contents

1 Releases **3**

2 Requirements **5**

3 Dependencies **7**

4 Installation **9**

5 Getting Started **11**

6 Configuration **13**

7 Contents **15**

7.1 Getting Started 15
7.2 Connections 15
7.3 Integration with Python 16
7.4 Pandas & NetworkX 16
7.5 Plotting 16
7.6 Dump 17
7.7 Options 17
7.8 Usage out of IPython 17
7.9 API 18

8 Indices and tables **19**

`ipython-cypher` is an IPython extension that provides `%cypher` and `%%cypher` magic for cells and lines, respectively. When executed through `ipython-cypher`, Cypher queries can be returned as a Pandas `DataFrame`, a NetworkX `MultiDiGraph`, or plotted using matplotlib.

This work is inspired by Catherine Devlin's ipython-sql.

CHAPTER 1

Releases

The latest release of `ipython-cypher` is **0.2.2**.

CHAPTER 2

Requirements

- Python 2.7, 3.3, 3.4
- Neo4j 1.9, 2.0, 2.1, 2.2

CHAPTER 3

Dependencies

- neo4jrestclient 2.0

Depending on your needs, you might want to Pandas, NetworkX and/or matplotlib in order for `ipython-cypher` to produce adapted ouputs from Cypher queries. The minimum versions supported are detailed below.

- Pandas 0.15
- NetworkX 1.9
- matplotlib 1.4

CHAPTER 4

Installation

To install, run the following:

```
$ pip install ipython-cypher
```

CHAPTER 5

Getting Started

Inside IPython, load the extension:

```
%load_ext cypher
```

And then you are reay to go by using the `%cypher` line magic:

```
%cypher MATCH (a)-[]-(b) RETURN a, b
```

Some Cypher queries can be very long, in those cases the the cell magic, `%%cypher` comes in handy:

```
%%cypher
create
    // Nodes
    (Neo:Crew {name:'Neo'}),
    (Morpheus:Crew {name: 'Morpheus'}),
    (Trinity:Crew {name: 'Trinity'}),
    // Relationships
    (Neo)-[:KNOWS]->(Morpheus),
    (Neo)-[:LOVES]->(Trinity),
```

Queries results can be stored in a variable and then converted to a Pandas `DataFrame`:

```
results = %cypher MATCH (a)-[]-(b) RETURN a, b
results.get_dataframe()
```

Or to a NetworkX `MultiDiGraph`:

```
results.get_graph()
```

See real examples in an IPython Notebook.

CHAPTER 6

Configuration

To change the behaviour of the cypher magic function, you can configure it:

```
%config CypherMagic

... list of options

%config CypherMagic.some_option = new_value
```

CHAPTER 7

Contents

7.1 Getting Started

Inside IPython, load the extension:

```
%load_ext cypher
```

And then you are reay to go by using the `%cypher` line magic:

```
%cypher MATCH (a)-[]-(b) RETURN a, b
```

Some Cypher queries can be very long, in those cases the cell magic, `%%cypher` comes in handy:

```
%%cypher
create
    // Nodes
    (Neo:Crew {name:'Neo'}),
    (Morpheus:Crew {name: 'Morpheus'}),
    (Trinity:Crew {name: 'Trinity'}),
    // Relationships
    (Neo)-[:KNOWS]->(Morpheus),
    (Neo)-[:LOVES]->(Trinity);
```

7.2 Connections

By default `ipython-cypher` will connect to <http://localhost:7474/db/data>, but the connection string can be passed at the beginning, and referenced later on:

```
%%cypher https://me:mypw@myhost:7474/db/data
match (n) return n limit 1
```

After that, the same connection can be reused by adding the pair username and host, as in `username@hostname`, as the connection string:

```
%%cypher me@myhost
match (n) return n limit 1
```

Assigning alias to connections is also available, and some times, can be even better:

```
%%cypher https://long.host.ec2.machin.com:7474/db/data as test1
match (n) return n limit 1
```

Once is set, can be used as usual:

```
%%cypher test1
match (n) return n limit 1
```

In order to change the default connection string, the environment variables `NEO4J_URI` or `NEO4J_URL` can be used. In the future, to change the default connection there will be an IPython option to set it, and even a config file to define all your Neo4j servers. Soon!

7.3 Integration with Python

Queries results can be stored in a variable and then converted to other formats:

```
results = %cypher MATCH (a)-[]-(b) RETURN a, b
```

When necessary, parameters are retrieved from the current namespace:

```
name = "Trinity"
%cypher MATCH (a)-[]-(b) WHERE a.name={name} RETURN a, b
```

Furthermore, the `%cypher` line magic can be used in-line with Python code:

```
for i in range(1, 5):
    %cypher match (n) return n, n.name limit {i}
```

7.4 Pandas & NetworkX

Results can be converted to a Pandas `DataFrame` by calling the funcion `get_dataframe()`:

```
results.get_dataframe()
```

The same can be achieved by using the lazy loading property `.dataframe`, but in this case default values for the creation of the `DataFrame` will be used:

```
results.dataframe
```

And the same applies for NetworkX `MultiDiGraph`. By default it will create a `MultiDiGraph`, but some options, such as if the graph should be directed or not, can be passed:

```
results.get_graph()
results.graph
```

These options are only functional when `pandas` and `networkx` packages are installed.

7.5 Plotting

However, we don't always need the full power of Pandas when we just want to take a quick look at the data. For those use cases, and if `matplotlib` is installed, `ipython-cypher` includes several handy functions:

- `.bar()`, will plot a bar chart trying its bets guesses.
- `.pie()`, the same for pie charts.
- `.plot()`, with the deafault `matplotlib` line bar, but supporint the passing of any keyword argument to the `matplotlib.plot` function.

- `.draw()`, will try to draw a simple NetworkX graph if the package is installed.

7.6 Dump

Other times, just generating a simple CSV is required, and `ipython-cypher` includes a function to export the results of a specific query:

```
results.csv(filename="filename.csv")
```

7.7 Options

The next parameters can set by using IPython Notebook config system, `CypherMagic`, or by passing arguments to the `run()` function when using `ipython-cypher` outside of IPython.

- **`auto_html (<bool>).`** Return a D3 representation of the graph instead of regular result sets (default: `False`).
- **`auto_limit (<int>).`** Automatically limit the size of the returned result sets (default: `0`).
- **`auto_networkx (<bool>).`** Return NetworkX MultiDiGraph instead of regular result sets (default: `False`).
- **`auto_pandas (<bool>).`** Return Pandas DataFrame instead of regular result sets (default: `False`).
- **`data_contents (<bool>).`** Bring extra data to render the results as a graph (default: `True`).
- **`display_limit (<int>).`** Automatically limit the number of rows displayed (full result set is still stored, default: `0`).
- **`feedback (<bool>).`** Print number of rows affected (default: `True`).
- **`rest (<bool>).`** Return full REST representations of objects inside the result sets (default: `False`).
- **`short_errors (<bool>).`** Don't display the full traceback on Neo4j errors (default: `True`).
- **`style (<unicode>).`** Set the table printing style to any of prettytable's defined styles (currently `DEFAULT`, `MSWORD_FRIENDLY`, `PLAIN_COLUMNS`, `RANDOM`, default: `u'DEFAULT'`).

7.8 Usage out of IPython

`ipython-cypher` can also be easily used outside IPython. The main function that makes this possible is `cypher.run()`, that takes a Cypher query string, and optional parameters for the query in a dictionary. By default, `http://localhost:7474/db/data` will be used, but a URL connection string to a Neo4j instance, or a `cypher.run.Connection` object can be passed as the last parameter:

```
import cypher

cypher.run("MATCH (a)-[]-(b) RETURN a, b")
```

7.9 API

7.9.1 `cypher` Package

7.9.2 `run` Module

7.9.3 `magic` Module

7.9.4 `column_guesser` Module

7.9.5 `parse` Module

7.9.6 `connection` Module

CHAPTER 8

Indices and tables

- genindex
- modindex
- search