



Hands-on scikit-network A Python Package for Graph Analysis

Thomas Bonald, Tiphaine Viard

Graph Embedding and Mining workshop @ ECML-PKDD
September 2021

Tutorial outline

09:50 - 10:00	Opening
10:00 - 11:30	Data structures, manipulation, simple tasks <i>Tiphaine Viard</i>
11:30 - 11:45	Coffee break
11:45 - 12:45	Analysis of the Wikipedia “vital articles” graph <i>Thomas Bonald</i>
12:45 - 13:00	Concluding Remarks

All material on
<https://gem-ecmlpkdd.github.io/program/>

Graph data

Graphs

Social networks

Web graphs

Knowledge graphs

Bipartite graphs

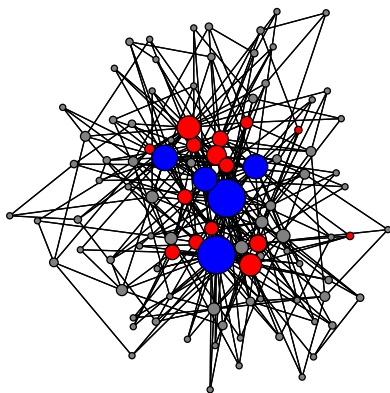
User \leftrightarrow Product

Actor \leftrightarrow Movie

Deputy \leftrightarrow Bill

Document \leftrightarrow Word

Patient \leftrightarrow Med



(Bi-)adjacency matrix

Graphs

Social networks

Web graphs

Knowledge graphs

$$A = \begin{bmatrix} & & 1 & & 1 \\ & 1 & & & \\ & & & 1 & \\ & & 1 & & \\ 1 & & & & \end{bmatrix}$$

Bipartite graphs

User \leftrightarrow Product

Actor \leftrightarrow Movie

Deputy \leftrightarrow Bill

Document \leftrightarrow Word

Patient \leftrightarrow Med

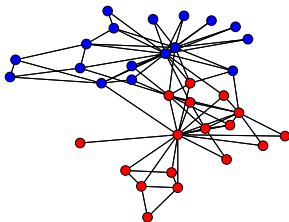
$$B = \begin{bmatrix} & 1 & & 1 & 1 & \\ 1 & & & & & \\ & & 1 & & & \\ 1 & & & 1 & & \\ & & & & 1 & \\ & & & 1 & & \end{bmatrix}$$

Sparse data

Graph	#nodes	#edges	Density
Openflights	2,939	30,500	$\approx 10^{-3}$
WordNet	146k	657k	$\approx 10^{-5}$
Wikipedia	12M	378M	$\approx 10^{-6}$
Twitter	42M	1.5G	$\approx 10^{-6}$
Friendster	68M	2.5G	$\approx 10^{-7}$

Bipartite graph	#nodes	#edges	Density
Message-Word	11k; 56k	1M	$\approx 10^{-3}$
Movie-Actor	88k; 45k	304k	$\approx 10^{-4}$
User-Product	21M; 10M	83M	$\approx 10^{-7}$

Graph analysis



Key tasks

Clustering

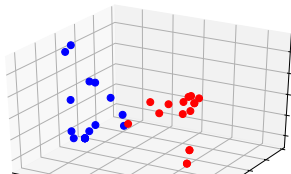
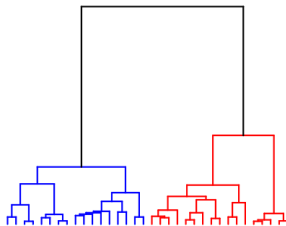
Hierarchy

Ranking

Classification

Embedding

Link prediction

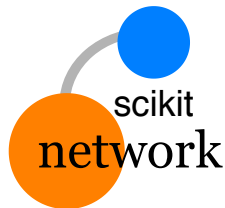


Scikit-network

A Python library for graph analysis

- ▶ **easy** to install `pip install scikit-network`
- ▶ **easy** to use `algorithm.fit(data)`
- ▶ **fast** and **memory-efficient**

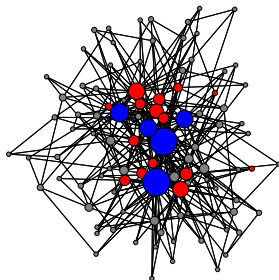
Relies on **NumPy** and **SciPy** only
BSD license



Data format

Graph = **adjacency** matrix or **biadjacency** matrix

Represented in the **CSR** (Compressed Sparse Row) format of SciPy



$$A = \begin{bmatrix} & 1 & & 1 \\ 1 & & & \\ & & 1 & \\ 1 & & 1 & \end{bmatrix}$$

Fast matrix-vector products

The COOrdinate format

A sparse matrix (in dense format):

$$\begin{bmatrix} 3 & 0 & 5 & 0 & 0 & 0 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 2 & 4 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 3 & 5 & 3 & 0 & 0 \\ 0 & 2 & 0 & 4 & 1 & 5 & 0 & 0 & 1 & 0 \\ 5 & 0 & 3 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The COOrdinate format

A sparse matrix (in dense format):

$$\begin{bmatrix} 3 & 0 & 5 & 0 & 0 & 0 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 2 & 4 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 3 & 5 & 3 & 0 & 0 \\ 0 & 2 & 0 & 4 & 1 & 5 & 0 & 0 & 1 & 0 \\ 5 & 0 & 3 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The same matrix in COO (COOrdinate) format:

$$(0,0,3), (0,2,5), (0,6,4), \dots$$

or equivalently,

$$\text{row} = 0, 0, 0, 0, 1, 1, \dots$$

$$\text{col} = 0, 2, 6, 7, 4, 6, 7, \dots$$

$$\text{data} = 3, 5, 4, 4, 4, 2, 4, \dots$$

The CSR (Compressed Sparse Row) format

A (not so) sparse matrix (in dense format):

$$\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

The CSR (Compressed Sparse Row) format

A (not so) sparse matrix (in dense format):

$$\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

The same matrix in coordinates:

$$(0,0,2), (1,1,3)$$

or equivalently, in CSR format:

$$\text{indices} = 0, 1$$

$$\text{indptr} = 0, 1, 2$$

$$\text{data} = 2, 3$$

Dense = 4 values, sparse = 7 values

The CSR (Compressed Sparse Row) format

A sparse matrix (in dense format):

$$\begin{bmatrix} 3 & 0 & 5 & 0 & 0 & 0 & 4 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 2 & 4 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 3 & 5 & 3 & 0 & 0 \\ 0 & 2 & 0 & 4 & 1 & 5 & 0 & 0 & 1 & 0 \\ 5 & 0 & 3 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Coordinates:

$$(0,0,3), (0,2,5), (0,6,4), \dots$$

The same matrix in CSR (Compressed Sparse Row) format:

$$\text{indices} = 0, 2, 6, 7, 4, 6, 7, 0, \dots$$

$$\text{indptr} = 0, 4, 7, 12, 17, 20$$

$$\text{data} = 3, 5, 4, 4, 4, 2, 4, \dots$$

Dense = 50 values, sparse = 46 values ($20 + 6 + 20$)

Properties of the CSR format

Pros

- ▶ Efficient storage
- ▶ Fast row slicing
- ▶ Fast matrix-vector product

Cons

- ▶ Slow column slicing
- ▶ Slow modification (e.g., add an entry)
- ▶ Slow transpose

Properties of the CSR format

Pros

- ▶ Efficient storage
- ▶ Fast row slicing
- ▶ Fast matrix-vector product

Cons

- ▶ Slow column slicing
- ▶ Slow modification (e.g., add an entry)
- ▶ Slow transpose

For efficient column slicing, use the CSC (Compressed Sparse Column) format (= CSR of the transpose matrix)

Other Python libraries for graphs

NetworkX
Python only

iGraph
Core in C/C++

graph-tool
Core in C/C++

	NetworkX	iGraph	graph-tool	scikit-network
Data	✓	✗	✓	✓
Topology	✓	✓	✓	✓
Clustering	✓	✓	✗	✓
Hierarchy	✗	✓	✓	✓
Ranking	✓	✓	✓	✓
Classification	✓	✗	✗	✓
Embedding	✓	✗	✓	✓
Visualization	✓	✓	✓	✓

✓ Available


✓ Partially available or not scalable

✗ Not available


Performance

Test on the Orkut graph (3M nodes, 117M edges)

RAM usage

NetworkX	iGraph	graph-tool	scikit-network
	18G	10G	1G

Running times

	iGraph	graph-tool	scikit-network
Louvain	33 min		2 min
PageRank	3 min 56 s	45 s	48 s
HITS	1 min 20 s	2 min 24 s	1 min 49 s

Perspectives

New tasks

- ▶ Graph generation (e.g., from nearest neighbors)
- ▶ Graph kernels
- ▶ Anomaly detection

New types of graphs

- ▶ Dynamic graphs
- ▶ Stream graphs
- ▶ Labeled graphs

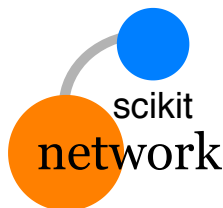
Credits

Development lead

- ▶ Thomas Bonald
- ▶ Marc Jeanmougin
- ▶ Nathan de Lara
- ▶ Quentin Lutz
- ▶ Tiphaine Viard

Contributors @LINCS

- ▶ Armand Bosch
- ▶ Bertrand Charpentier
- ▶ François Durand
- ▶ Alexandre Hollocou
- ▶ Fabien Mathieu



+ External contributors