NOESIS-Python Documentation

The NOESIS Team

Sep 19, 2018

Contents

Official Python API for NOESIS, an open source framework for network data mining that provides a large collection of network analysis techniques, including the analysis of network structural properties, community detection methods, link scoring, and link prediction, as well as network visualization algorithms.

CHAPTER 1

Installation

From source:

```
git clone https://github.com/fvictor/noesis-python.git
cd noesis-python
python setup.py install
```

From PyPi:

```
pip install noesis
```

CHAPTER 2

Getting started

NOESIS for Python provides simple and unified interfaces for most of the implemented techniques. The following example loads a network from a GML file and detects its communities using the Kernighan–Lin algorithm:

```
from noesis import Noesis

ns = Noesis()

network_reader = ns.create_network_reader('GML')
network = network_reader.read('my_network.gml')

community_detector = ns.create_community_detector('KernighanLin')
communities = community_detector.compute(network)

for node in range(network.nodes()):
        print('Node {} belongs to community {}'.format(node, communities[node]))

ns.end()
```

The following example generates a network of 20 nodes and 100 links using the Erdös–Rényi model and computes the PageRank score of each node:

```
from noesis import Noesis

ns = Noesis()

network = ns.create_network_from_model('ErdosRenyi', 20, 100)

pagerank_scorer = ns.create_node_scorer('PageRank', 0.9)
scores = pagerank_scorer.compute(network)

for node in range(network.nodes()):
        print('Node {} has a PageRank score of {}'.format(node, scores[node]))

ns.end()
```

Always remember to call the *end* method of *Noesis* class to properly terminate the NOESIS session and finish the execution of your program.

CHAPTER 3

Reference

3.1 NOESIS

class noesis.**Noesis**

This class implements the entry point for the functionality provided by this package and handles the states of the Java Virtual Machine required to run NOESIS in Python.

create_community_detector(*detector*, **args*)

Create a community detector.

Parameters

- **detector** (*string*) – Technique used for community detection. See *communities.CommunityDetector* for more details.
- **args** (*parameters*) – Parameters for the community detector. See *communities.CommunityDetector* for more details.

Returns **detector** – A community detector with the specified parameters.

Return type *CommunityDetector*

create_layout(*layout*, **args*)

Create a layout.

Parameters

- **layout** (*string*) – Technique used to compute the layout. See *layout.Layout* for more details.
- **args** (*parameters*) – Parameters for the layout algorithm. See *layout.Layout* for more details.

Returns **layout** – A layout with the specified parameters.

Return type *Layout*

create_link_predictor(*predictor*, **args*)

Create a link predictor.

Parameters

- **scorer** (*string*) – Technique used to compute node pair scores. See *links.LinkPredictor* for more details.
- **args** (*parameters*) – Parameters for the link predictor. See *links.LinkPredictor* for more details.

Returns **scorer** – A link predictor with the specified parameters.

Return type *LinkPredictor*

create_link_scorer(*scorer*, **args*)
Create a link scorer.

Parameters

- **scorer** (*string*) – Technique used to compute link scores. See *links.LinkScorer* for more details.
- **args** (*parameters*) – Parameters for the link scorer. See *links.LinkScorer* for more details.

Returns **scorer** – A link scorer with the specified parameters.

Return type *LinkScorer*

create_network()
Create a network.

Returns **network** – A generated network.

Return type *Network*

create_network_from_model(*model*, **args*)
Create a network from a network formation model.

Parameters

- **model** (*string*) – Network formation model used to generate the network. See *models.NetworkModel* for more details.
- **args** (*parameters*) – Parameters for the network formation model. See *models.NetworkModel* for more details.

Returns **network** – A network generated using the specified network formation model.

Return type *Network*

create_network_reader(*reader*, **args*)
Create a network reader.

Parameters

- **reader** (*string*) – File format of the network reader. See *io.NetworkReader* for more details.
- **args** (*parameters*) – Parameters for the network reader. See *io.NetworkReader* for more details.

Returns **reader** – A network reader with the specified parameters.

Return type *NetworkReader*

create_node_scorer(*scorer*, **args*)
Create a node scorer.

Parameters

- **scorer** (*string*) – Technique used to compute node scores. See *nodes.NodeScorer* for more details.
- **args** (*parameters*) – Parameters for the node scorer. See *nodes.NodeScorer* for more details.

Returns **scorer** – A node scorer with the specified parameters.

Return type *NodeScorer*

end()
End the NOESIS session. This method must be called in order to properly end the program execution.

launch_analyzer(*show_splash=False*)
Launchs the NOESIS graphic user interface.

Parameters **show_splash** (*bool, default False*) – True to show the splash screen, False otherwise.

3.2 Network

Warning: Do not manually instantiate this class using the class constructor. Instantiate the class using the method *noesis.Noesis.create_network()*.

class noesis.network.**Network**(***kargs*)
This class implements the interface for networks, providing methods to query and manipulate network nodes and links.

add_link(*source*, *target*)
Add a directed link to the network.

Parameters

- **source** (*integer*) – Index of the source node.
- **target** (*integer*) – Index of the target node.

add_link2(*source*, *target*)
Add an undirected link to the network.

Parameters

- **source** (*integer*) – Index of the source node.
- **target** (*integer*) – Index of the target node.

add_node()
Add a node to the network.

Returns **index** – Index of the added node.

Return type integer

contains_link(*source*, *target*)
Check if the network contains a link.

Parameters

- **source** (*integer*) – Index of the source node.

- **target** (*integer*) – Index of the target node.

Returns **is_contained** – True if the network contains the link, False otherwise.

Return type boolean

contains_node(*node*)
Check if the network contains a node.

Parameters **node** (*integer*) – Index of the node to check.

Returns **is_contained** – True if the network contains the node, False otherwise.

Return type boolean

degree(*node*)
Get the degree of a given node in the network.

Parameters **node** (*integer*) – Index of the node from which to obtain the degree.

Returns **degree** – Degree of the node.

Return type integer

in_degree(*node*)
Get the in-degree of a given node in the network.

Parameters **node** (*integer*) – Index of the node from which to obtain the in-degree.

Returns **degree** – In-degree of the node.

Return type integer

in_links(*node*)
Get in-links for a given node in the network.

Parameters **node** (*integer*) – Index of the node from which to obtain in-links.

Returns **links** – List of indices of nodes with links to the specified node.

Return type list of integers

is_directed()
Check if a network is directed.

Returns **is_directed** – True if the network is directed, False otherwise.

Return type bool

links()
Get the number of links in the network.

Returns **link_count** – Number of links in the network.

Return type integer

nodes()
Get the number of nodes in the network.

Returns **node_count** – Number of nodes in the network.

Return type integer

out_degree(*node*)
Get the out-degree of a given node in the network.

Parameters **node** (*integer*) – Index of the node from which to obtain the out-degree.

Returns **degree** – Out-degree of the node.

Return type integer

out_links (*node*)
Get out-links for a given node in the network.

Parameters **node** (*integer*) – Index of the node from which to obtain out-links.

Returns **links** – List of indices of nodes with links from the specified node.

Return type list of integers

remove_link (*source*, *target*)
Remove a directed link from the network.

Parameters

- **source** (*integer*) – Index of the source node.
- **target** (*integer*) – Index of the target node.

remove_link2 (*source*, *target*)
Remove an undirected link from the network.

Parameters

- **source** (*integer*) – Index of the source node.
- **target** (*integer*) – Index of the target node.

remove_node (*node*)
Remove a given node from the network.

Parameters **node** (*integer*) – Index of the node to remove.

set_directed (*directed*)
Set network if network is directed or undirected.

Parameters **directed** (*boolean*) – True to set the network as directed, False otherwise.

3.3 Node scoring

Warning: Do not manually instantiate this class using the class constructor. Instantiate the class using the method *noesis.Noesis.create_node_scorer()*.

class noesis.nodes.**NodeScorer** (*scorer*, **args*)
This class implements the interface for node scorers. These algorithms compute a score for each node according to certain specific rules.

Parameters

- **scorer** (*string*) – Technique used to compute node scores. Currently supported techniques are 'AdjustedBetweenness', 'AdjustedCloseness', 'AveragePathLength', 'Betweenness', 'BetweennessScore', 'Closeness', 'ClusteringCoefficient', 'ConnectedComponents', 'Decay', 'Degree', 'DegreeAssortativity', 'DiffusionCentrality', 'Eccentricity', 'EigenvectorCentrality', 'FreemanBetweenness', 'HITS', 'InDegree', 'KatzCentrality', 'LinkBetweenness', 'LinkBetweennessScore', 'LinkEmbeddedness', 'LinkNeighborhoodOverlap', 'LinkNeighborhoodSize', 'LinkRays', 'NormalizedBetweenness', 'NormalizedDecay', 'NormalizedDegree', 'NormalizedInDegree', 'NormalizedOutDegree', 'OutDegree', 'PageRank', 'PathLength', and 'UnbiasedDegreeAssortativity'.

- **args** (*parameters*) – Parameters for the node scorer. These parameters are specific for each node scorer and more details are provided in NOESIS documentation.

compute(*network*)
Compute scores for each node in a given network.

Parameters **network** (Network) – Network for which the node scores will be computed.

Returns **scores** – Vector of scores for each node.

Return type ndarray, shape (num_nodes,)

3.4 Link scoring and prediction

Warning: Do not manually instantiate these classes using their class constructors. Instantiate these classes using the methods *create_link_scorer()* and *create_link_predictor()*.

class noesis.links.**LinkPredictor**(*predictor*, **args*)
This class implements the interface for link predictors. These algorithms compute a score for each pair of nodes according to certain specific rules.

Parameters

- **scorer** (*string*) – Technique used to compute node pair scores. Currently supported techniques are: - Local: 'CommonNeighbors', 'AdamicAdar', 'ResourceAllocation', 'PreferentialAttachment', 'HubDepressed', 'HubPromoted', 'Jaccard', 'LocalLeichtHolmeNewman', 'Salton', and 'Sorensen'. - Global: 'Katz', 'RandomWalk', 'RandomWalkWithRestart', 'FlowPropagation', 'PseudoinverseLaplacian', 'AverageCommuteTime', 'RandomForestKernel', and 'GlobalLeichtHolmeNewman'.
- **args** (*parameters*) – Parameters for the link predictor. These parameters are specific for each link predictor and more details are provided in NOESIS documentation.

compute(*network*)
Compute scores for each pair of nodes in a given network.

Parameters **network** (Network) – Network for which the node pair scores will be computed.

Returns **scores** – Matrix of node pair scores.

Return type ndarray, shape (num_nodes,num_nodes)

class noesis.links.**LinkScorer**(*scorer*, **args*)
This class implements the interface for link scorers. These algorithms compute a score for each link according to certain specific rules.

Parameters

- **scorer** (*string*) – Technique used to compute link scores. Currently supported techniques are: - Local: 'CommonNeighbors', 'AdamicAdar', 'ResourceAllocation', 'PreferentialAttachment', 'HubDepressed', 'HubPromoted', 'Jaccard', 'LocalLeichtHolmeNewman', 'Salton', and 'Sorensen'. - Global: 'Katz', 'RandomWalk', 'RandomWalkWithRestart', 'FlowPropagation', 'PseudoinverseLaplacian', 'AverageCommuteTime', 'RandomForestKernel', and 'GlobalLeichtHolmeNewman'.
- **args** (*parameters*) – Parameters for the link scorer. These parameters are specific for each link scorer and more details are provided in NOESIS documentation.

compute(*network*)
Compute scores for each link in a given network.

Parameters **network** (Network) – Network for which the link scores will be computed.

Returns **scores** – A list of tuples with the format (source_node, target_node, link_score).

Return type list of tuples

class noesis.links.**LinkTask**
Base abstract class for link-related scores.

3.5 Community detection

Warning: Do not manually instantiate this class using the class constructor. Instantiate the class using the method *noesis.Noesis.create_community_detector()*.

class noesis.communities.**CommunityDetector**(*detector*, **args*)
This class implements the interface for community detectors. These algorithms find groups of densely connected nodes.

Parameters

- **detector** (*string*) – Technique used for community detection. Currently supported techniques are: - Hierarchical agglomerative: 'SingleLink', 'AverageLink', and 'CompleteLink'. - Hierarchical divisive: 'NewmanGirvan' and 'Radicchi'. - Modularity-based: 'FastGreedy' and 'MultiStepGreedy'. - Partitioning-based: 'KernighanLin' and 'KMeans'. - Spectral: 'UKMeans', 'NJW', and 'EIG1'. - Overlapping: 'BigClam'.
- **args** (*parameters*) – Parameters for the community detector. These parameters are specific for each algorithm and more details are provided in NOESIS documentation.

compute(*network*)
Compute node communities for a given network.

Parameters **network** (Network) – Network for which the communities will be computed.

Returns **communities** – A list of integers indicating the community that each node belongs to.

Return type list of integers

3.6 Network input/output

Warning: Do not manually instantiate this class using the class constructor. Instantiate the class using the method *noesis.Noesis.create_network_reader()*.

Network input and output

class noesis.io.**NetworkReader**(*reader*, **args*)
This class implements the interface for networks readers. These readers can load a network from a file following the specified file format.

Parameters

- **reader** (*string*) – File format of the network reader. Currently supported formats are 'ASCII', 'GDF', 'GML', 'GraphML', 'Pajek', 'SNAP', and 'SNAPGZ'.
- **args** (*parameters*) – Parameters for the network reader. These parameters are specific for each file format and more details are provided in NOESIS documentation.

read (*filepath*)

Read a network from a specific file path.

Parameters **filepath** (*string*) – Path of the file to read.

Returns **network** – The network that has been read from the file.

Return type *Network*

3.7 Graph layouts

Warning: Do not manually instantiate this class using the class constructor. Instantiate the class using the method *create_layout()*.

class noesis.layout.**Layout** (*layout*, **args*)

This class implements the interface for network layouts. These algorithms compute visual coordinates for each node to obtain a more pleasant visualization of the network.

Parameters

- **layout** (*string*) – Technique used to compute the layout. Currently supported techniques are 'Random', 'KamadaKawai', 'FruchtermanReingold', 'Hierarchical', 'Linear', 'Mesh', 'Radial', 'BinaryTree', 'Circular', 'Hypercube', 'Star', and 'Toroidal'.
- **args** (*parameters*) – Parameters for the layout algorithm. These parameters are specific for each algorithm and more details are provided in NOESIS documentation.

compute (*network*)

Compute a layout for a given network.

Parameters **network** (Network) – Network for which the layout will be computed.

Returns

- **x** (*ndarray, shape (num_nodes,)*) – Vector of x coordinates for each node.
- **y** (*ndarray, shape (num_nodes,)*) – Vector of y coordinates for each node.

3.8 Network formation models

Warning: Do not manually instantiate this class using the class constructor. Instantiate the class using the method *noesis.Noesis.create_network_from_model()*.

class noesis.models.**NetworkModel** (*model*, **args*)

This class implements the interface for network models, allowing to instantiate a network given a network formation model.

Parameters

- **model** (*string*) – Network formation model used to generate the network. Currently supported models are 'ErdosRenyi', 'BarabasiAlbert', 'WattsStrogatz', 'Gilbert', 'PriceCitation', 'AnchoredRandom', 'ConnectedRandom', 'Complete', 'BinaryTree', 'Hypercube', 'Isolate', 'Mesh', 'Ring', 'Star', 'Tandem', and 'Toroidal'.
- **args** (*parameters*) – Parameters the for network formation model. These parameters are specific for each network formation model and more details are provided in NOESIS documentation.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index