Pybiographs Documentation

Release 0.0.1

David Kas, Guillem Duran

Apr 16, 2020

Getting started:

CHAPTER 1

Graphs

1.1 InteractionGraph

class pybiographs.graphs.**InteractionGraph**(*directed: bool = False*)

Represents a graph of protein-protein interactions.

It can load a directed or undirected graph, and wraps the corresponding networkx graph. If it represents a directed graph it behaves as a `networkx.Digraph`. Otherwise it behaves as a `networkx.Graph`.

Node attributes:

- label : uniprot_id from uniprot (https://www.uniprot.org/)
- **string node_type** [metabolome_graph (with pathway and metabolites associated)] or other_protein (not referenced as metabolome proteins: no metabolites and no pathway on smpd : https://smpdb.ca/)
- **string info** [Text explaining the products of the mRNA that codes] the protein from STRING database : https://string-db.org/).
- **list cellular_components** [list of Go Id cellular components the protein] is belonging to in QuickGO (see gene ontology : https://www.ebi.ac.uk/QuickGO/). `Mappings``.go_to_name‘‘ maps GoId to names.
- list molecular_functions : list of Go Id as above but for molecular functions.
- list biological_processes : list of Go Id as above but for biological processes.
- **list expression_data** [Vector of float of size 308 corresponding to expression] ranks of intial RNAm coding the protein renormalized from 0 to 1 in 308 tissues (see https://bgee.org/). index_tissue is a dict mapping index in vector to string tissue name.
- **list metabolites** [list of HMDB ID metabolites associated to protein if] it is a metabolome_protein (see https://hmdb.ca/). `Mappings``.metabolite_id_to_name‘‘ contains the

  mapping from id to metabolite name.

- **list pathways** [list of pathway of names the belonging to the metabolome_protein.] for more information on a pathway, search it on smpd (might not be referenced).
- string sequence : Amino acid sequence for the protein.

Directed edge attributes:

Undirected edge attributes:

__init__(*directed: bool = False*)
Initialize a *InteractionGraph*.

Parameters **directed** – If `True` the instance will represent a directed graph of interactions. If `False` the instance will represent an undirected graph of interactions.

classify_tissue_by_node_expression(*nodes*, *limit=30*)
Takes a list of nodes, then print tissues where the set of nodes is the most expressed.

Parameters

- **nodes** – nodes to be searched for.
- **limit** – limit to print.

Returns None.

get_nodes_by_sequence_regex(*sequence_regex*)
Search a regex in amino acid sequences stored in nodes and returns matching node results. :param sequence_regex: regex to search in sequences

Returns a list of nodes (uniprot ids).

info_sequence_regex(*res*, *reg*, *attribute*)
Depending on node attribute "info" or "sequence", search regex in all node attributes and return an union between query node results and nodes that have a match.

Parameters

- **res** – entry node list corresponding to query result so far.
- **reg** – regex to be search as a string.
- **attribute** – "info" or "sequence".

Returns union list of matching nodes and res.

is_directed
Return `True` if it represents a directed graph. Otherwise return `False`.

metabolites_regex(*res*, *reg*)
Search a regex in metabolites names in graph node attributes and return an union of matching results with query results so far.

Parameters

- **res** – entry node list corresponding to query results so far.
- **reg** – regex to be search as a string.

Returns union list of matching nodes and res.

most_present_biological_processes(*graph*, *tissue*, *bp_size_thresh=0*, *limit=10*)
After sub_graph_from_node_propagation, this function can be used to print most affected biological processes.

Parameters

- **graph** – sub graph to print most affected components.
- **tissue** – string, the tissue where to analyze the biological processes.
- **bp_size_thresh** – a threshold on size on number proteins in biological processes
- **limit** – limit to print.

Returns None.

most_present_cellular_components(*graph*, *tissue*, *cc_size_thresh=0*, *limit=10*)
Similar to most_affected_biological_processes; but for cellular components.

Parameters

- **graph** – sub graph to print most affected cellular components.
- **tissue** – string, the tissue where to analyze the biological processes.
- **cc_size_thresh** – a threshold on size on number proteins in molecular function.
- **limit** – limit to print.

Returns None.

ontology_regex(*res*, *reg*)
Search a regex in all ontological attributes of nodes in graph ("cellular_components", "biological_processes", "cellular_components") and return an union of matching results with query results so far.

Parameters

- **res** – entry node list corresponding to query results so far.
- **reg** – regex to be search as a string.

Returns union list of matching nodes and res.

pathway_regex(*res*, *reg*)
Search a regex in "pathways" attribute of nodes in graph and return union of arg res with matching nodes.

Parameters

- **res** – entry node list corresponding to query results so far.
- **reg** – regex to be search as a string.

Returns union list of matching nodes and res.

print_sub_graph_nodes(*graph*, *print_spec='i_o_p_m'*, *limit=30*)
Print nodes in graph up to a limit with specs similar to sub_graph_by_node_regex_search.

Parameters

- **graph** – graph where to print nodes
- **print_spec** – a string to specify what to print, a combination of "i" (for info), "p" (*for pathways), "m" (for metabolites), "o" (for ontologies*) –
- **by underscore "_". As a split is applied, the order is not important.** (*separated*) –
- **limit** – limit to the number of prints.

Returns None.

propagate_node(*node*, *diameter*)
Recursive part of sub_graph_from_node_propagation :param node: node to propagate :param diameter: diameter that still need to be propagated

Returns node results.

recurrent_ontology_query(*sub_search*, *nodes*)
test recursively queries in request for ontology and return result as list

Parameters

- **sub_search** – the list request.
- **nodes** – nodes to be searched.

Returns a list containing proteins satisfying results.

remove_edges_by_threshold(*graph*, *score_threshold=0.0*)
Remove from graph all edges that have a score inferior to threshold. Considering removing edges can do new orphan nodes (with no edges), those node are removed also from graph.

Parameters

- **graph** – graph to clean edges.
- **score_threshold** – attribute score threshold, should be between 0 and 1.
- **edge scores are.**(*as*) –

Returns cleaned graph.

restrict_by_tissue_threshold(*nodes*, *tissue*, *threshold*)
Remove all nodes from entry nodes that does'nt have an expression superior to a threshold

Parameters

- **nodes** – list of nodes.
- **tissue** – a string key for tissue.
- **threshold** – float between 0 and 1.

Returns new list containing nodes that satisfy threshold properties.

sub_graph_by_node_ontology_search(*ontology_query=None*, *tissue=None*, *score_threshold=0.0*, *expression_threshold=0.0*)
This is the public method that need to be used to query for a subgraph by searching for a set expression query in ontology. As for method above, returns a sub graph cleaned by threshold. ontology query language: simple query: "goid" -> returns subgraph with nodes in goid basic query : ["and", "goid1", "goid2", ...] -> returns subgraph with nodes in goid1 and goid2 and ... basic query : ["not", "goid1", ...] -> returns subgraph with nodes not in goid1, ... basic query : ["or", "goid1", "goid2"] -> returns subgraph with nodes in goid1 or goid2 or ... complex query : ["and", ["or", "g1", "g2", ["and", "g3", "g4"]], ["not", "g5"], "g6"] : -> return subgraph with nodes satisfying (g1 or g2 or (g3 and g4)) and (not g5) and g6 :param ontology_query: the query list :param tissue: restrict the search by tissue (exemple "lung"). Default None and ignored. :param score_threshold: threshold to apply to edges in subgraph, between 0 and 1 as the scores :param expression_threshold: threshold to apply to expression score in tissue. :param Ignored if tissue is None.:

Returns New sub graph.

sub_graph_by_node_regex_search(*regex*, *spec='i_p_m_o'*, *tissue=None*, *score_threshold=0.0*, *expression_threshold=0.0*)
This is the public method that need to be used to query for a subgraph of the graph by searching a regex in the node attribute. Step 1 : search nodes with matching regex in attributes. Step 2 : removes nodes that are

inferior to expression threshold. Step 3 : create subragph from parent graph and removes edges inferior to score threshold.

Parameters

- **regex** – regex to be searched in node attributes.
- **spec** – a string to specify where to search for, a combination of "i" (for info), "p" (*for pathways*), "m" (*for metabolites*), "o" (*for ontologies*) –
- "_". **As a split is applied, the order is not important.** (*underscore*) –
- **tissue** – restrict the search by tissue (exemple "lung"). Default None and ignored.
- **score_threshold** – threshold to apply to edges in subgraph, between 0 and 1 as
- **scores.** (*the*) –
- **expression_threshold** – threshold to apply to expression score in tissue.
- **if tissue is None.** (*Ignored*) –

Returns New sub graph.

sub_graph_from_node_propagation(*nodes*, *diameter=1*, *tissue=None*, *score_threshold=0.0*, *expression_threshold=0.0*)

Takes nodes and returns sub graph generated by neighbor propagation up to a diameter. Will start recursively to take all neighbors of entry nodes, then neighbors of neighbors, etc... The method will return subgraph thresholded eventually by tissue and scores on edges.

Parameters

- **nodes** – nodes to propagate
- **diameter** – diameter of the resulting sub graph around the node.
- **tissue** – restrict the search by tissue (exemple "lung"). Default None and ignored.
- **score_threshold** – threshold to apply to edges in subgraph, between 0 and 1 as the scores
- **expression_threshold** – threshold to apply to expression score in tissue.
- **if tissue is None.** (*Ignored*) –

Returns New graph with results.

1.2 OntologyGraph

class pybiographs.graphs.**OntologyGraph**(*name: str*)

CHAPTER 2

Covid Data

2.1 CovidData

class pybiographs.covid_data.**CovidData**

__init__()
Initialize a *CovidData*.

CHAPTER 3

Mappings

3.1 Mappings

class pybiographs.mappings.**Mappings**

CHAPTER 4

Deep Learning

4.1 PPInteractionDataset

pybiographs.dl_models.torch_datasets.**PPInteractionDataset**

4.2 PPGCN

pybiographs.dl_models.graph_dl_model.**PPGCN**

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Index

Symbols

C

G

I

M

O

P

R

S