CLARINET

version 1

Yasmine Ahmed

October 02, 2021

Contents

Welcome to CLARINET's documentation!

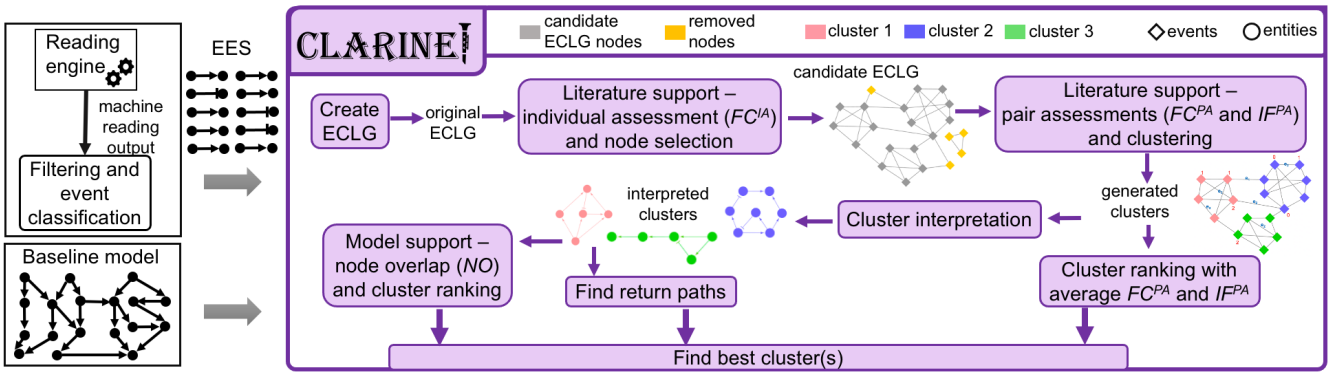
CLARINET (CLARIfying NETworks) ia a novel tool for rapid model assembly by automatically extending dynamic network models with the information published in literature. This facilitates information reuse and data reproducibility and replaces hundreds or thousands of manual experiments, thereby reducing the time needed for the advancement of knowledge.

CLARINET objectives

1. Utilizing the knowledge published in literature and suggests model extensions.
2. Studying events extracted from literature as a collaboration graph, including several metrics that rely on the event occurrence and co-occurrence frequency in literature.
3. Allowing users to explore different selection criteria when automatically finding best extensions for their models.

CLARINET architecture

(Left) CLARINET inputs: Extracted Event Set (EES) and Baseline model. (Right) Flow diagram of the CLARINET processing steps and outputs.



Dependencies

Python libraries: pandas, numpy, network, math, pickle, community, matplotlib.pyplot.

Applications

The primary application area of CLARINET is dynamic and causal network models.

Funding

CLARINET was partially supported by the AIMCancer DARPA award (W911NF-17-1-0135).

CLARINET functions (`CLARINET`)

This page provides a detailed documentation of the CLARINET functions.

Functions

`CLARINET.runClarinet.`**`create_eclg`**(`interaction_filename`, `model_dict`)

This function creates the ECLG where a node is an event (e.g., biochemical interaction) and there is an edge between two nodes (events) if they happen to occur in the same paper.

Parameters:

- **model_dict** (*dict*) – Dictionary that holds baseline model regulator and regulated elements
- **interaction_filename** (*str*) – The path of the reading output file returns**G** – Event CoLlaboration Graph rtypeGraph

`CLARINET.runClarinet.`**`node_weighting`**(`G`, `freqTh`, `path`)

This function assigns weights to nodes using frequency class.

Parameters:

- **G** (*undirected graph*) – ECLG
- **freqTh** (*int*) – Frequency class threshold value, events (nodes) having FC > this value will be removed
- **path** (*str*) – The output directory where the genereted files will be saved returns**G** – ECLG after the removal of the less frequent nodes rtypeundirected graph

`CLARINET.runClarinet.`**`edge_weighting`**(`G`, `path`, `weightMethod`)

This function assigns weights to nodes using frequency class (FC) or inverse frequency formula (IF).

Parameters:

- **G** (*undirected graph*) – ECLG
- **path** (*str*) – The output directory where the genereted files will be saved
- **weightMethod** (*str*) – FC or IF returns**G** – ECLG after assigning weights to edges rtypeundirected graph

`CLARINET.runClarinet.`**`clustering`**(`G`, `path`)

This function clusters the ECLG using the community detection algorithm by Blondel et al..

Parameters:

- **G** (*undirected graph*) – ECLG
- **path** (*str*) – The output directory where the genereted files will be saved

`CLARINET.runClarinet.`**`merge_clusters`**(`regulators`, `path`, `ReturnTh`)

This function prints indices of clusters to be merged based on the existence of one or more return paths. It generates the grouped_ext_Merged pickle file that contains the merged clusters.

Parameters:

- **regulators** (*dict*) – contains baseline model elements and corresponding regulator elements
- **path** (*str*) – The path of the directory that contains the grouped_ext file

`CLARINET.runClarinet.`**`make_diGraph`**(`mdldict`)

This function creates a directed graph for a model in the BioRECIPES format.

Parameters: **mdldict** (*dict*) – contains baseline model elements and corresponding regulator elements returns**G** – contains the baseline model in the form of directed graph where nodes are model elements and edges are interaction between model elements. rtypeDiGraph

Index