Documentación Grafos Temporales

Grafos temporales de Kostakos (aproximación) - Implementación en Python

Getting started

La mejor forma de explorar el módulo es dándole a los notebooks disponibles en la carpeta 'notebooks'

Source code

class **temporal_graph.TemporalGraph**(*tiempos*)

Grafo temporal

_times (list):

Los _times nos sirven para manejar las columnas (visual) de tiempos. Es una lista de datetime.datetime. Corresponde a todos los tiempos que entran en juego en el grafo

_last_node_appearance (dict):

Estructura para mantener el ultimo (mayor) datetime.datetime en el que un nodo de su correspondiente fila es utilizado. Por ejemplo:

{

'a': datetime.datetime(2018, 12, 19, 14, 34, 14, 736048)

'b': datetime.datetime(2018, 12, 19, 14, 40, 34, 736048)

...

}

_graph (networkx.classes.digraph.DiGraph):

Grafo dirigido que le asigna peso 0.0 (instantáneo) a los links entre nodos de distintas filas y peso con diferencia en segundos entre nodos desagregados de una misma fila.

_step (int):

Paso que nos sirve para guardar una imagen (img_<step>) cada vez que se agrega un enlace/link.

average_temporal_proximity(*node_from*, *node_to*, *verbose=False*)

ATP

Args:

node_from (str): Label del nodo (base) desde el cual se calcula la proximidad temporal promedio.
Por ej: 'A', 'B', etc.

node_to (str): Label del nodo (base) hasta el cual se calcula la proximidad temporal promedio.
Por ej: 'A', 'B', etc.

verbose (bool): Indica si se muestra la salida de los pasos realizados.

Returns:

float: En promedio, cuánto tiempo toma ir desde X hasta Y.

`average_temporal_proximity_from_node`(*node*)

Retorna las proximidades temporales promedio del nodo hacia el resto de los nodos

Args:

node (str): Nodo desde el cual calcular las
proximidades temporales promedio. Por ejemplo: 'A'

Returns:

dict: Diccionario con las proximidades temporales promedio
desde el nodo recibido. Por ejemplo, para 'A' se puede devolver:

{
'A': 0.0,
'B': 561600.0,
'C': 43200.0,
'D': 144000.0,
'E': 43200.0
}

`average_temporal_proximity_to_node`(*node*)

Retorna las proximidades temporales promedio del resto de los nodos del grafo hacia el nodo recibido.

Args:

node (str): Nodo hacia el cual calcular las proximidades temporales promedio.
Por ejemplo: 'D'

Returns:

dict: Diccionario con las proximidades temporales promedio desde el nodo recibido.

Por ejemplo, para 'D' se puede devolver:

{
'A': 144000.0,
'B': 374400.0,
'C': None,
'D': 0.0,
'E': 86400.0
}

average_temporal_reach(*node*)

On average, how quickly does X reach the rest of the network.

Lease: `P out`

Args:

node (str): Nodo. Por ejemplo: 'A'.

Returns:

float, o None si desde el nodo no se alcanza ningun otro nodo.

average_temporal_reachability(*node*)

On average, how quickly is X reached by the rest of the network.

Lease: `P in`

Args:

node (str): Nodo. Por ejemplo: 'A'.

Returns:

float, o None si el nodo no es alcanzado por ningun otro nodo.

build_links_from_data(*data*, *col_sender='sender'*, *col_destination='recipient'*, *col_time='time'*, *save_images=False*, *verbose=True*)

Crea links en base al dataframe con la data correspondiente.

Args:

data (pandas.Dataframe):

```
| sender | recipient | time
|  (str) |  (str)    | (datetime.datetime)
-------------------------------------------------
| A      |   B       | 2018-12-19 14:34:14.736048
| A      |   C, E    | 2018-12-19 14:34:15.736424
-------------------------------------------------
```

column_sender (str): Columna que corresponde al emisor.

column_destination (str): Columna que corresponde al receptor.

column_time (str): Columna que corresponde al tiempo en el que se produce la interacción.

save_images (bool): Indica si se tiene que guardar el grafo cada vez que se agrega un nuevo enlace.

create_link(*sender*, *receiver*, *time*)

Crea un link entre los nodos recibidos y ademas crea un link con linea punteada a la aparición anterior de la fila del nodo correspondiente.

Args:

sender (str): Nodo (estatico) desde el cual se comienza la interacción.

receiver (str): Nodo (estatico) desde el cual se recibe la interacción.

Precondicion: sender != receiver

time (datetime.datetime): Tiempo en el que se produce la interacción.

Returns:

tuple: tupla con los elementos:

- sender (str),
- receiver (str),
- time (datetime),
- instancia creada del nodo origen (str)

Raises:

Exception: El `time` debe ser un datetime.datetime

de python.

get_graph()

Retorna el grafo de networkx

plot(*only_save=False*, *output_folder='output'*, *paleta={'label_color': '#23512f', 'links_color': '#c20d00', 'nodes_color': '#ffce00', 'temp_links_color': '#23512f'}*)

Dibuja el grafo temporal

Args:

only_save (bool): Indica si se deben guardar un png del grafo en lugar de mostrarlo una vez terminado por pantalla (True).

TODO: Esto está pensado para generar el gif de forma manual. Será posible generarlo de forma automática?

output_folder (str): Carpeta en la cual se van a guardar las imagenes generadas.

Por defecto, intenta guardarlas en una carpeta 'output'.

paleta (dict): Paleta de colores para el grafo.

Debe contener las claves:

- 'nodes_color',
- 'links_color',
- 'temp_links_color'

para indicar los colores de los nodos, de los links entre nodos distintos y los links entre nodos del mismo nodo base (a1, a2, a3, etc –> a) respectivamente.

temporal_proximity(*node_from*, *node_to*, *time_from=None*, *time_to=None*, *verbose=False*)

Devuelve la proximidad temporal entre los nodos

Args:

node_from (str): Label del nodo (base) desde el cual se calcula la proximidad temporal. Por ej: 'A', 'B', etc.

node_to (str): Label del nodo (base) hasta el cual se calcula la proximidad temporal. Por ej: 'A', 'B', etc.

time_from (int): precondicion temporal (tiempo desde)

time_to (int): poscondicion temporal (tiempo hasta)

Returns:

list: Lista de los nodos que representan el camino mas corto en cuanto a lo temporal, desde node_from hasta node_to.

weight(*path*) ¶

Retorna el peso del camino recibido.

Args:

path (list): lista de nodos del camino.

Se espera que la lista contenga instancias de nodos.

Returns:

float: Peso del camino.