22 April 2020

ATTENDEES

Seán O'Callaghan, Viktor Zamaraev

SYNOPSIS

Initial meeting to discuss project posting on COMP702 project system.

NOTES

In a dynamic network network, both vertices and edges can change over time. For Temporal networks, the vertices are fixed, only the edges change over time (over the duration of the networks timespan). Temporal graphs offer more accurate information of the network/system when compared to a static graph representation. An open source library would provide a readily available toolset/package for the modelling of temporal networks, while also encourage collaboration through contributions to the package on github. Networkx is an example of an open source project for static graphing of networks.

The project can be split into stages. The first stage would involve research into the area and investigating any tools/packages that are already available. Any packages found can be reviewed and summarized in terms of what functionality is offered (is the framework specialized/general).

The second stage is deciding on an architecture for the tool. What classes would form the core framework, what definitions would they reflect. Temporal graphs can be implemented in many ways. A core library would be a good startpoint. After that, one could implement basic algorithms. The final library should be something that can be posted (useful for others) on an open source platform (github).

Since the final project is intended to be open source, there is a need to understand how to deploy and manage such a project. Community collaboration will also need to be organized.

Assigned project and setup 10am weekly meetings.

ACTION ITEMS

- Start learning some of the theory surrounding temporal networks.
- Learn what packages are available for temporal networks (see where we are).

01 May 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev

SYNOPSIS

Weekly meeting

NOTES

Basic package functionality should include creating the network from input data. Adding/removing time stamped edges. Taking measurements/implementing basic algorithms (like demonstrating the optimal/shortest path). The initial network can work with a discrete timeline/timestamps (natural numbers). The timeline can be assumed to be an input parameter or sequence of edges. Later, a more general set of timestamps could be defined with floats, to represent a continuous timeline.

Some other implementations include teneto, pathpy. Both packages work on a single class implementation (network), with time-stamped edges & vertices loaded into a dictionary. Proposed to create a more generalized/granular framework that uses more classes for components like the edges & vertices themselves (good idea).

ACTION ITEMS

- Continue research into temporal networks (build base knowledge).
- Start logbook.

University of Liverpool | MSc Computer Science
COMP702 | Open Source Temporal Network Library

15 May 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev

SYNOPSIS

Weekly meeting

NOTES

Reading/developing background knowledge to form a good impression of the area before deciding on an initial architecture. Thoughts on classes; a network input/generator class which can validate inputs and create networks. Edges and nodes which are subcomponents of a network/graph, of which there can be various types (with base class?). An observer class which can view/request a representation of the network at time t, or over a time interval. Can also take measurements of the network and can be extended to run particular algorithms. A class for network visualization. All classes can have base class and be extended upon to allow for flexible design. Review open source library setup on github. Current temporal graph libraries include teneto and pathpy. A good algorithm to implement is to find the shortest path to a particular node (traverse network, edge duration). Check other libraries to see how they implement algorithms (networkx, for example). One aspect of the project could be implementing different network generators/models (when you don’t have input data, you can build a network using a model).

ACTION ITEMS

Create a name for the library.

Create a repository for the network (include a roadmap to provide project plan/direction).

05 June 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev

SYNOPSIS

Foremost-path algorithm discussion

NOTES

What to try and achieve first, for better understanding of architecture and what’s needed in the core library. Implementing a foremost-path algorithm would be a great way to start. Input; parse csv file to create graph (both nodes and edges).

ACTION ITEMS

Implement initial foremost-path algorithm.

12 June 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev

SYNOPSIS

Foremost-time implementation discussion.

NOTES

Edge-stream; multiple edges of the same source-sink pair. Can’t model standalone nodes if input is an edge list. Multiple constructors for graph creation? Need validation to ensure nodes & edges actually form a proper graph (by construction). The algorithm should produce a foremost-tree, which has the foremost-path from the root node to all other nodes in the graph.

ACTION ITEMS

Continue implementing the foremost-path algorithm.

19 June 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev

SYNOPSIS

General library discussion.

NOTES

Find/create sample data to test/demonstrate library (good idea). Look at modelling undirected graphs. When adding edges, add them in bunches, more efficient than adding one-by-one. Class TemporalTree. Basic abstractions of models for temporal networks and use algorithms. Develop a comprehensive library (compare NetworkX). Not for large scale networks. Visualization is very useful, circle plot is a good idea.

ACTION ITEMS

Continue the foremost-path implementation.

03 July 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev

SYNOPSIS

Design & specification report feedback.

NOTES

Update shortest-path to foremost-path in the algorithm section. Introduce foremost-path/tree concepts. Update pseudo-code. Make sure to understand everything mentioned in the report. TemporalTree class could be a subclass of TemporalGraph (investigate). Note in the report that standardized testing documentation will be added to the library as a guideline for contributors.

ACTION ITEMS

Presentation next Thursday 09/07/19.

31 July 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev

SYNOPSIS

Weekly meeting.

NOTES

The method ‘get_graph_by_time’ should return a static graph object. It will be renamed to ‘get_snapshot’. Similarly, ‘get_graph_by_interval’ will be renamed to ‘get_temporal_subgraph’. Adding nodes/edges should be done through a graph method, which calls the respective underlying nodes/edges methods. Investigate adding lines with arrows with pyplot. Build a nodelink plotting feature. Transportation showcase is a good idea, and can demonstrate the foremost tree algorithm. Another library showcase example can revolve around an approximation algorithm for the removal of edges in a graph to reduce the graph’s overall reachability (research area, spread of disease). Reachability can be calculated by using the foremost tree for each node.

ACTION ITEMS

Investigate paper ‘Deleting edges to restrict the size of an epidemic in temporal networks’. Build a graph generator using the Networkx RandomGNP static generator.

07 August 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev

SYNOPSIS

Weekly meeting.

NOTES

Try to connect points on the slice plot for better visibility of edges. Edges should not overlap, they are disjoint. For the library showcase, use the Transport for London public API to generate a temporal data-set. This API allows for ‘easy’ access to real-world data from a temporal network. Develop a client to generate nodes & edges from lines, stations and journeys.

ACTION ITEMS

Build a client to interact with the TFL Restful API.

14 August 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev

SYNOPSIS

Weekly meeting.

NOTES

Include a colormap/bubble plot for viewing node metrics such as reachability. Investigate the use of tooltips on pyplot (for viewing edge times). Develop a good way to distinguish between start and end points of an edge in the slice plot. Experiment with TFL data to find a good set of data for the showcase (i.e. every 5mins for 1 hour, multiple hours of a day, multiple days, etc.).

ACTION ITEMS

Prepare for the design showcase presentation, continue creating the project report (due next week).

27 August 2020

ATTENDEES

Seán O’Callaghan, Viktor Zamaraev, Othon Michael

SYNOPSIS

Project showcase presentation.

NOTES

Presented the library functionality through the Transport for London showcase example. Began with a very short introduction and then ran through a live demonstration of the library. Showcased functionality such as network creation, component definitions, snapshots and subgraphs. Demonstrated plotting functionality, such as plotting the raw transportation network, plotting the network with geographical data and also plotting the journeys on the network using the slice plot. Analyzed temporal subgraphs of the network through the foremost tree and reachability algorithms, and explained some of the resulting observations.

ACTION ITEMS

- Add Othon to the library GitHub once released.
- Add a readme to describe the addition of a new algorithm to the library.
- Add another data-set example to the library.

28 August 2020

ATTENDEES

Seán O'Callaghan, Viktor Zamaraev

SYNOPSIS

Weekly meeting.

NOTES

Need to add remove edge/node functionality. Make the default duration of an edge 0 if not duration is supplied in the input data. Develop GitHub page, add roadmap, strategy, philosophy and make it open source ready. For the foremost tree algorithm, check if the input graph is directed/undirected and cater for both. For the undirected case, for each edge, check which node has the foremost time 'inf' and which has already been reached, and travel from reached to unreached. If both nodes are unreached, skip. If the data structure of the edges & nodes were to be improved for efficiency, most likely it would be to cater for specific algorithms (for example, current structure for edges is an edge-stream which caters to the foremost tree algorithm). Therefore, additional data structures could be implemented in parallel with options. It would be useful to implement a method to extract the underlying (static) graph from a temporal graph. The 'get_temporal_graph' method should include parameters for specifying time interval(s) and also sets of vertices.

ACTION ITEMS

- Add remove node & remove edge.
- Outline library roadmap.
- Update the 'get_temporal_subgraph' method.
- Add 'extract_underlying_graph' method.

04 September 2020

ATTENDEES

Seán O'Callaghan, Viktor Zamaraev

SYNOPSIS

Weekly meeting.

NOTES

Build an examples directory with documented, commented examples detailing how to use the library in various ways. Note this in the readme (you can find more examples…). The method 'get_temporal_subgraph' will include a parameter of pairs of time intervals (include edges completely within the interval only). Keep the original timeline intact. Overlapping intervals should automatically merge as edges within graphs are unique.

ACTION ITEMS

- Begin preparation of thesis.
- Develop a new 'get_temporal_subgraph' method.