The Journal of Open Source Software

Raphtory: The temporal graph engine for Rust and Python

Ben Steer [1,4], **Naomi A. Arnold** [3¶], **Cheick Tidiane Ba** [6,4], **Renaud Lambiotte** [2,1,8], **Haaroon Yousaf** [1], **Lucas Jeub** [1], **Fabian Murariu**[5], **Shivam Kapoor**[1], **Pedro Rico** [1], **Rachel Chan**[1], **Louis Chan**[1], **James Alford**[1], **Richard G. Clegg** [4], **Felix Cuadrado** [7,4], **Matthew Russell Barnes**[4], **Peijie Zhong**[4], **John Pougué-Biyong** [2], **and Alhamza Alnaimi**[1]

1 Pometry, United Kingdom **2** Mathematical Institute, University of Oxford, United Kingdom **3** Networks Science Institute, Northeastern University London, United Kingdom **4** School of Electronic Engineering and Computer Science, Queen Mary University of London, United Kingdom **5** 32 Bytes Software, United Kingdom **6** University of Milan, Italy **7** Universidad Politécnica de Madrid, Spain **8** Alan Turing Institute, United Kingdom ¶ Corresponding author

DOI: 10.21105/joss.05940

Software

- Review
- Repository
- Archive

Editor: Luiz Irber

Reviewers:

- @abhishektiwari
- @arashbm

Submitted: 26 June 2023
Published: 26 March 2024

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Raphtory is a platform for building and analysing temporal networks. The library includes methods for creating networks from a variety of data sources; algorithms to explore their structure and evolution; and an extensible GraphQL server for deployment of applications built on top. Raphtory's core engine is built in Rust, for efficiency, with Python interfaces, for ease of use. Raphtory is developed by network scientists, with a background in Physics, Applied Mathematics, Engineering and Computer Science, for use across academia and industry.

Statement of need

Networks are at the core of data science solutions in a variety of domains, including computer science, computational social science, and the life sciences (Newman, 2018). Networks are a powerful language focusing on the connectivity of systems, and offer a rich toolbox to extract greater understanding from data. Several network analysis tools exist, including NetworkX (Hagberg et al., 2008), graph-tool (Peixoto, 2014) and igraph (Csardi et al., 2006), and are freely accessible to scientists, practitioners and data miners.

However, with abundant cheap storage and tools for logging every event which occurs in an ecosystem, datasets have become increasingly rich, combining different types of information that cannot be incorporated in a standard network model (Lambiotte et al., 2019). In particular, the temporal nature of many complex systems has led to the emergence of the field of temporal networks, with its own models and algorithms (Holme & Saramäki, 2012; Masuda & Lambiotte, 2016).

Unfortunately, despite active academic research in the last decade, no efficient, generalised and production-ready system has been developed to explore the temporal dimension of networks. To support practitioners who wish to exploit both the structure and dynamics of their data, we have developed Raphtory.

Related Software

Besides the aforementioned packages, few open access tools have been developed for the mining of temporal networks, with the existing solutions focusing on specific sub-problems within the space. Those which have attempted to generalise to all temporal network analysis are either actively under development, but too preliminary to use in production, or have been abandoned due to lack of funding or changing research goals.

As examples of these three categories: Pathpy is a Python package for the analysis of time series data on networks, but focuses on extracting and analysing time-respecting paths (Hackl et al., 2021). Similarly, DyNetX (Rossetti et al., 2023), a pure Python library relying on networkX, focuses on temporal slicing and the computation of time-respecting paths. The recently released Reticula offers a range of methods developed in C++ with a Python interface (Badie-Modiri & Kivelä, 2023). Phasik (Lucas, Townsend-Teague, et al., 2023; Lucas, Morris, et al., 2023), written in Python, focuses on inferring phases from temporal network data. EvolvingGraphs.jl (Zhang, 2015), RecallGraph (Mukhopadhyay, Accessed 19-06-2023) and Chronograph (Erb et al., 2017) all saw significant work before development was halted indefinitely.

Raphtory is a valuable addition to this ecosystem for the following reasons. Originally developed in Scala (Steer et al., 2020), its current core is entirely written in Rust. This is to ensure fast and memory-efficient computation that a pure Python implementation could not achieve, and to handle the sheer volume of temporal network data, which often dwarfs that of an equivalent static network.

The library provides an expressive Python interface for interoperability with other data science tools, as well as simpler and more maintainable code. In addition, the library is built with a focus on scalability, as it relies on efficient data structures that can be used to extract different views of large temporal graphs. This avoids the creation of multiple graph objects that is not feasible with large datasets. The use of these new features is supported by well-documented APIs and tutorials, guiding the user from data loading through to analysis.

Overview

The core Raphtory model consists of a base temporal graph which maintains a chronological log of all changes to its structure and property values over time. A graph can be created using simple functions for adding/removing vertices and edges at different time points, as well as updating their properties. Alternatively, a graph can be generated through in-built loaders for common data sources/formats (Example 1).

Once a graph has been created, a user may generate 'graph views' which set some structural or temporal constraints through which the underlying graph may be observed. Graph views can be generated programmatically over a desired time range (windows), over sets of nodes which pass some user-defined criteria (subgraphs), or over a subset of layers if the graph is multilayered. Additionally, the views can leverage event durations and support various semantics for deletions.

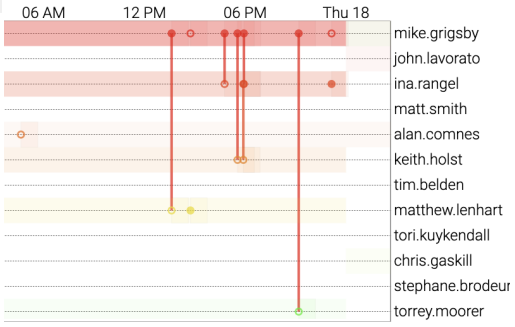
To reduce memory footprint, graph views are only materialised upon access. This allows a user to maintain thousands of different perspectives of their graph simultaneously, which can be explored and compared through the application of graph algorithms and metrics (Example 2). Furthermore, Raphtory provides extensions for automatic null model generation and exporting of views to other graph libraries such as NetworkX.

Raphtory includes fast and scalable implementations of algorithms for temporal network mining such as temporal motifs (Example 3) and temporal reachability. In addition, it exposes its internal API for implementing algorithms in Rust, and surfacing them in Python.

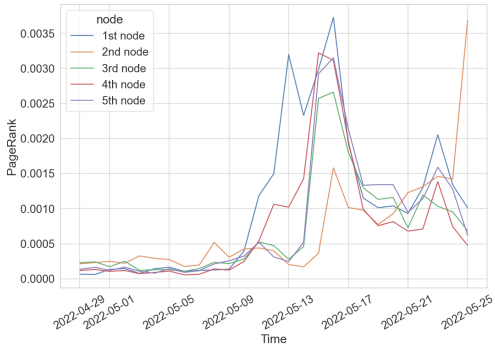
Finally, Raphtory is built with a focus on ease of use and can be installed using standard Python and Rust package managers. Once installed it can be integrated within an analysis

pipeline or run standalone as a GraphQL service.

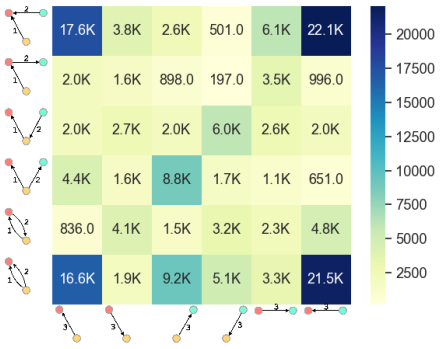
```python
import raphtory as rp
import pandas as pd

enron = pd.read_csv("enron.csv", names = ["time",
  "src",
    "dst",
      "message",
        "type"])

g = rp.Graph()

for id, record in enron.iterrows():
      g.add_edge(timestamp = record['time'],
        src = record['src'],
         dst = record['dst'],
           properties = {"message": record['message']},
             layer = record['type'])
```




```python
from raphtory.plottingutils import ordinal_number

# Load graph of transactions
g = rp.graph_loader.stable_coin_graph(path = dataset_path,
  subset=True)
layer_graph = g.layer("LUNC")

# Get top 5 nodes by PageRank
pg = rp.algorithms.pagerank(g = layer_graph)
top_5, scores = zip(*pg.top_k(5))

# Extract their daily PageRank value
metric_list = []
graph_views = layer_graph.rolling(window = '1 day',
  step = '1 day')

for view_g in graph_views:
    time = view_g.end_date_time()
    values = pagerank(g = view_g)
    for i, n in enumerate(top_5_nodes):
        metric_list.append( (time,
          values[n],
            f"{ordinal_number(i+1)} node") )

# Plot these values
plotting_function(metric_list, x = "Time",
  y = "PageRank", hue = "node")
```




```python
from raphtory.algorithms import global_temporal_three_node_motif
from raphtory.plottingutils import global_motif_heatplot

# Load the graph using the load-from-pandas functionality
df = pd.read_csv("sx-mathoverflow.txt", names=["src", dst",
  "time"])
g = rp.Graph()
g.load_edges_from_pandas(df,src="src",dst="dst",time="time")

# Count the 1-hour motifs for g
counts = global_temporal_three_node_motif(g, delta=3600)

# Out-of-the-box motif plot
ax = global_motif_heatplot(counts, cmap='YlGnBu')
```



Caption. First line (Example 1): In a temporal network, edges are dynamical entities connecting pairs of nodes. Second line (Example 2): Generation of a sequence of graph views at a given time resolution and on selected layers, to run standard network algorithms, here Pagerank. Third line (Example 3): Raphtory offers rapid implementations of algorithms specifically designed for temporal networks, here finding significant temporal motifs (Paranjape et al., 2017).

Projects using Raphtory

Raphtory has proved an invaluable resource in industrial and academic projects, for instance to characterise the time evolution of the fringe social network Gab (Arnold et al., 2021),

transactions of users of a dark web marketplace Alphabay using temporal motifs (Paranjape et al., 2017) or anomalous patterns of activity in NFT trades (Yousaf et al., 2023). The library has recently been significantly rewritten, and we expect that with its new functionalities, efficiency and ease of use, it will become an essential part of the network science community.

Arnold, N. A., Steer, B., Hafnaoui, I., Parada G, H. A., Mondragón, R. J., Cuadrado, F., & Clegg, R. G. (2021). Moving with the times: Investigating the alt-right network Gab with temporal interaction graphs. *Proceedings of the ACM on Human-Computer Interaction*, *CSCW*. https://doi.org/10.1145/3479591

Badie-Modiri, A., & Kivelä, M. (2023). Reticula: A temporal network and hypergraph analysis software package. *SoftwareX*, *21*, 101301. https://doi.org/10.1016/j.softx.2022.101301

Csardi, G., Nepusz, T., & others. (2006). The igraph software package for complex network research. *InterJournal, Complex Systems*, *1695*(5), 1–9. https://doi.org/10.5281/zenodo.7682609

Erb, B., Meißner, D., Pietron, J., & Kargl, F. (2017). Chronograph: A distributed processing platform for online and batch computations on event-sourced graphs. *Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems*. https://doi.org/10.1145/3093742.3093913

Hackl, J., Scholtes, I., Petrović, L. V., Perri, V., Verginer, L., & Gote, C. (2021). Analysis and visualisation of time series data on networks with pathpy. *Companion Proceedings of the Web Conference 2021*, 530–532. https://doi.org/10.1145/3442442.3452052

Hagberg, A., Swart, P., & S Chult, D. (2008). *Exploring network structure, dynamics, and function using NetworkX*. Los Alamos National Lab.(LANL), Los Alamos, NM (United States); https://www.osti.gov/biblio/960616.

Holme, P., & Saramäki, J. (2012). Temporal networks. *Physics Reports*, *519*(3), 97–125. https://doi.org/10.1007/978-3-642-36461-7

Lambiotte, R., Rosvall, M., & Scholtes, I. (2019). From networks to optimal higher-order models of complex systems. *Nature Physics*, *15*(4), 313–320. https://doi.org/10.1038/s41567-019-0459-y

Lucas, M., Morris, A., Townsend-Teague, A., Tichit, L., Habermann, B., & Barrat, A. (2023). Inferring cell cycle phases from a partially temporal network of protein interactions. *Cell Reports Methods*, *3*(2). https://doi.org/10.1101/2021.03.26.437187

Lucas, M., Townsend-Teague, A., Neri, M., Poetto, S., Morris, A., Habermann, B., & Tichit, L. (2023). Phasik: A python package to identify system states in partially temporal networks. *Journal of Open Source Software*, *8*(91), 5872. https://doi.org/10.21105/joss.05872

Masuda, N., & Lambiotte, R. (2016). *A guide to temporal networks*. World Scientific. https://doi.org/10.1142/q0033

Mukhopadhyay, A. (Accessed 19-06-2023). *RecallGraph*. https://github.com/RecallGraph/RecallGraph.

Newman, M. (2018). *Networks*. Oxford University Press. https://doi.org/10.1093/oso/9780198805090.001.0001

Paranjape, A., Benson, A. R., & Leskovec, J. (2017). Motifs in temporal networks. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 601–610. https://doi.org/10.1145/3018661.3018731

Peixoto, T. P. (2014). The graph-tool python library. *Figshare*. https://doi.org/10.6084/m9.figshare.1164194.v14

Rossetti, G., Hoeven, E. ter, Norman, U., Jorquera, D., Dormán, H., & Dorner, M. (2023). *GiulioRossetti/dynetx: v0.3.2* (Version v0.3.2b). Zenodo. https://doi.org/10.5281/zenodo.8009585

Steer, B., Cuadrado, F., & Clegg, R. (2020). Raphtory: Streaming analysis of distributed temporal graphs. *Future Generation Computer Systems*, *102*, 453–464. https://doi.org/10.1016/j.future.2019.08.022

Yousaf, H., Arnold, N. A., Lambiotte, R., LaRock, T., Clegg, R. G., Zhong, P., Alnaimi, A., & Steer, B. (2023). Non-Markovian paths and cycles in NFT trades. *arXiv Preprint arXiv:2303.11181*. https://doi.org/10.48550/arXiv.2303.11181

Zhang, W. (2015). *Dynamic network analysis in Julia*. https://eprints.maths.manchester.ac.uk/2376/