IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/44AI/2024/4443124/
Vol. 2024 (22), Article ID 4443124

Review of State-of-the-Art Tools for Network Analysis with NetworkX*

Edyta FRĄSZCZAK

Military University of Technology, Warsaw, Poland

Correspondence should be addressed to: Edyta FRĄSZCZAK, edyta.fraszczak97@gmail.com

* Presented at the 44th IBIMA International Conference, 27-28 November 2024 Granada, Spain

Abstract

This article offers a comprehensive review of the latest tools for network analysis, focusing on methods and applications that utilize the NetworkX package. It examines recent advancements in community detection, network propagation, source identification within social networks, and strategies for identifying critical nodes. Special emphasis is placed on advanced algorithms that enhance NetworkX's functionality in dynamic environments. This review serves as a valuable resource for researchers and practitioners looking for practical and effective tools for modern network analysis. The paper provides sample usages of the described libraries, encouraging readers to utilize them effectively.

Keywords: networks; network analysis; network data exploration; network analysis software

Introduction and research motivation

In today's world, networks are gaining attention across various research fields because they are often used to model and analyze complex systems. Networks can be found in many areas, helping to capture phenomena such as road systems, social connections, transport routes, and contact patterns. This widespread application has contributed to the popularity of network analysis in higher education and academic research [1], [2], [3]. With the rapid growth of network science, a variety of tools have been developed to make this field more accessible to a broader audience. Computer scientists, physicists, and mathematicians can utilize libraries, while graphical tools cater to social scientists, biologists, and educators [1], [3], [4], [5].

A key tool for network analysis in Python [6] is the NetworkX [4] package, which provides functions for creating, visualizing, and analyzing networks. It enables researchers to calculate centrality, detect communities, and explore relationships between nodes. However, NetworkX does not cover all available methods globally. As a result, specialized open-source projects have emerged that focus on specific areas such as community detection, centrality, propagation simulation, and identifying source nodes. While many algorithms have been developed, only a few well-known methods are included in general libraries like NetworkX or iGraph [7]. Utilizing other methods often requires finding the original paper that introduces the algorithm, implementing it, or locating a reliable implementation, learning how to use it, converting network formats, and adapting the results to meet specific needs. This process can make it challenging and time-consuming to use, study, and compare these methods.

This article showcases the latest tools researchers can use to reach specific goals, offering an overview of current solutions to help them understand available options and see how their methods may align. It also promotes these tools within the scientific community by demonstrating their applications across diverse scenarios.

Cite this Article as: Edyta FRĄSZCZAK, Vol. 2024 (22) "Review of State-of-the-Art Tools for Network Analysis with NetworkX" Communications of International Proceedings, Vol. 2024 (22), Article ID 4443124, https://doi.org/10.5171/2024.4443124

Nodes Evaluation

Node evaluation is a technique used to assess the importance of individual nodes within a network. This assessment helps identify the most influential nodes, providing insights into the network's structure and behavior. In practice, node evaluation involves measuring various forms of centrality, which indicate how "central" or "influential" a node is [8], [9], [10]. Centrality metrics capture different aspects of a node's position and influence, including its connectivity, control over network flow, and proximity to other nodes. These evaluations enable researchers to detect key nodes, analyze their contributions to information spread, and understand their roles within larger systems [11], [12], [13], [14]. Mathematically, the centrality of a node v can be represented as a function $C(v) = f(G, v)$.

`NetCenLib` (Network Centrality Library) [15] is a dedicated Python library that provides a convenient way to use an extensive set of network centrality measures. It allows us to quickly and easily run them and analyze the obtained results. The summary of the available techniques is presented in Table 1. The library is organized according to best Python practices, provides available documentation [16], and is easy to use. `NetCenLib` is available in the official Python packages repository and can be installed using the command: `pip install nsdlib.`

Table 1 List of implemented centrality measures in NetCenLib

Node evaluation metric name			
Algebraic	Current Flow Closeness	Group Closeness	Mnc
Average Distance	Decay	Group Degree	NetSleuth
Barycenter	Degree	Harmonic	Pagerank
Betweenness	Diffusion degree degree	Heatmap	Pdi
BottleNeck	Dispersion	Hubbell	Radiality
Centroid	Dynamic age	JordanCenter	Rumor
Closeness	Eccentricity	Katz	Second Order
ClusterRank ClusterRank	Eigenvector	Laplacian	Semi Local
Communicability Betweenness	Entropy	Leverage	Subgraph
Coreness	Geodestic k path path	Lin	Topological
Current Flow Betweenness	Group Betweenness	Load	Trophic Levels

Listing **1** shows a sample usage of the `NetCenLib` package to compute the four most popular centrality measures on the Karate Club graph using the `NetworkX` library. It evaluates nodes based on Degree Centrality, Betweenness Centrality, Closeness Centrality, and Eigenvector Centrality. All implemented techniques are accessible under the main module `netcenlib`, making the package easy to use.

```
import networkx as nx
import netcenlib as ncl

# Create a graph
G = nx.karate_club_graph()

# Compute degree centrality
degree_centrality = ncl.degree_centrality(G)

# Compute betweenness centrality
betweenness_centrality = ncl.betweenness_centrality(G)

# Compute closeness centrality
closeness_centrality = ncl.closeness_centrality(G)

# Compute eigenvector centrality
eigenvector_centrality = ncl.eigenvector_centrality(G)
```

```
# Print obtained score
ncl.print_dict(degree_centrality)
```

Listing 1 A sample usage of NetCenLib to compute different centrality measures

Community detection

Community detection refers to the process of identifying groups, or "communities," within a network where nodes are more closely connected to one another than to those outside the group. This technique is essential in network analysis because communities often represent significant clusters, such as social groups, functional modules, or interaction patterns [8], [19]. By detecting these communities, researchers can gain valuable insights into the structure and function of complex systems, whether they pertain to social networks, biological systems, or models of information spread. Mathematically, community detection methods aim to divide graph G into q disjoint sub-graphs $C_i = (V_i, E_i)$, in which $\forall i \neq j : C_i \cap C_j = \varnothing$ and $\bigcup_{i=1}^{k} C_i = V$. The `CDlib` (Community Discovery Library) [20] has been developed as a Python library to assist researchers in applying and comparing different community detection techniques. `CDlib` provides standardized access to a wide range of clustering algorithms, simplifying the analysis and evaluation of community structures across various networks. With easy access to diverse community detection methods and built-in tools for evaluation and visualization, `CDlib` supports researchers, developers, and practitioners in performing high-quality network analysis. This library streamlines the process of testing, refining, and selecting the most suitable community detection methods for specific applications, enhancing the depth and accuracy of network studies. The list of available community detection methods in `CDLib` package is presented in Table 2.

Table 2 Community detection methods available in CDLib package

Community detection method name			
AGDL	Entropy-based clustering	LAIS2	Rb_pots
Angel	Newman's leading eigenvector	Leiden	Rber pots
ASLPAw	EM	Lemon	Ricci
FluidC	EnDNTM	LFM	SBM
Bayan	Eva	Louvain	SBM nestes
Belief	FRC-FGSN	LPAM	SCAN
BiMLPA	Genetic	LPANNI	Siblinarity
BRIM	Gdmp2	LSWL	Significance communities
CoAch	Girvan–Newman	LSWL plus	SLPA
Combo	Graph Entropy	MCL	Spectral
CONGA	CNM	MCODE	Spinglass
CONGO	Head/Tail	MOD M	Surprise
Core expansion	HLC	MOD R	Thresholdclustering
CPM	I-Louvain	MULTICOM	TILES
CPM Bipartite	Infomap	Node perception	UMSTMO
Divide and Conquer Strategy	Infomap bipartite	OSSE	Weighted community
Demon	IPCA	Paris	Walkscan
DER	k-clique	PercoMVC	Walktrap
DPClus	Kcut	Principled custering	
Ego-networks	Label propagation	Regularised spectral	

Similarly to NetCenLib, access to a wide range of community detection techniques in CDLib is straightforward as it follows the best Python practices, and all implemented methods are exported into the main module. Listing 2 presents sample usage of how to obtain communities by the Louvain method over Karate club network. This package is also available in the PyPI repository and can be installed via the `pip install cdlib` command.

```
from cdlib import algorithms
import networkx as nx

# Create a graph
G = nx.karate_club_graph()

# Compute communities
coms = algorithms.louvain(G, weight='weight', resolution=1., randomize=False)

# Print communities
print(coms)
```

Listing 2 A sample usage of CDLib to compute communities via the Louvain algorithm

Propagation simulation

Propagation simulation, also known as diffusion simulation, involves mathematical models that describe how information, diseases, or influence spread through a network over time. These models are based on research and real-world observations, with certain simplifications to make them applicable across various domains. Propagation simulations are essential in fields like epidemiology, sociology, and marketing, as they allow researchers to explore and predict the dynamics of spread within complex systems. Using these models, researchers can simulate how quickly a disease might move through a population, identify key influencers within a social network, or estimate the potential reach of a marketing campaign [11], [21], [22], [23]. There are two main categories of those models: Epidemic and Opinion dynamics. `NDLib` (Network Diffusion library) [24] is a Python library that provides a unified platform to simulate a wide range of diffusion models, making it accessible for researchers, educators, and developers to analyze and visualize these complex processes. This package is also available in the PyPI repository and can be installed via the `pip install ndlib` command. Table 3 presents the list of available methods grouped by type.

Table 3 Propagation models available in NDLib

Type	Name	Type	Name
Epidemic	SI	Opinion dynamics	Voter
	SIS		Q-Voter
	SIR		Majority Rule
	SEIR (DT)		Sznajd
	SEIR (CT)		Cognitive Opinion Dynamics
	SEIS (DT)		Algorithmic Bias
	SEIS (CT)		Algorithmic Bias Media Model
	SWIR		Example
	Threshold		Weighted Hegselmann-Krause
	Generalized Threshold		Hegselmann-Krause
	Kertesz Threshold		
	Independent Cascades		
	Profile		
	Profile Threshold		

	UTLDR		
	Independent Cascades with Community Embeddedness and Permeability		
	Independent Cascades with Community Permeability		
	Independent Cascades with Community Embeddedness		

Listing 3 presents a sample code for configuring a simulated propagation process over the Karate Club network using the SI epidemic model. This process is slightly more complex than previous examples, as it not only runs the algorithm but also allows for configuring specific simulation conditions. NDLib is well-documented, actively maintained, and offers comprehensive online resources, making it user-friendly and reliable for conducting network simulations.

```
import ndlib.models.epidemics as ep
import ndlib.models.ModelConfig as mc
import networkx as nx

# Create a graph
g = nx.karate_club_graph()

# Model selection
model = ep.SIModel(g)

# Model configuration
cfg = mc.Configuration()
cfg.add_model_parameter('beta', 0.01)
cfg.add_model_parameter("fraction_infected", 0.05)
model.set_initial_status(cfg)

# Simulation
iterations = model.iteration_bunch(200)
```

Listing 3 A sample configuration and simulation of propagation by SI model over Karate club network with NDLib package

Source detection

Source detection in propagation refers to identifying the origin or initial source of a spreading process within a network. This is a critical task in fields such as epidemiology, cybersecurity [25], and social network analysis [3], [26], where tracing back to the source can help contain disease outbreaks, track down malware origins, or identify the first source of rumors or misinformation [9]. Technically, source detection algorithms aim to identify the origin(s) of propagation in a network by applying graph theory. In most cases the network is represented as an undirected graph $G(V,E)$, where V is a set of nodes (users/computer, etc.) and E is a set of edges (relationships between them). Propagation starts from a source node or a set of nodes, and it spreads along the edges based on specific diffusion models, such as SI, SIS, or SIR. These models simulate the propagation, creating a "propagation graph" that reflects how the spread unfolds over time under controlled conditions. At a given time t, the propagation graph $G_I(V_I,E_I)$ is the subgraph of G induced by the set of propagated (infected) nodes $V_I \subseteq V$ and the edges $E_I \subseteq E$ connecting them. The goal of source detection algorithms is to infer the initial source nodes S^* from the observed propagation graph G_I. Mathematically, this task involves estimating a probability distribution $P(s \mid G_I)$, where $s \in V$ (or $s \subseteq V$ for multiple sources) represents the most likely source or sources of propagation [9], [10], [27], [28]. A range of tools has emerged, providing implementations of various source detection algorithms. Among these, `NSDLib` (Network Source Detection Library)[29] offers a comprehensive library of detection algorithms, allowing for efficient integration and testing within different projects. Additionally, `RPaSDT` (Rumor Propagation and Source Detection Toolkit) [30] provides a graphical interface that enables users to conduct simulations, perform network

analysis, and evaluate detection techniques visually. This combination of libraries and graphical interfaces allows researchers and practitioners to explore and validate source detection methods more effectively, enhancing both usability and depth of analysis. Moreover, GUI based approach can be used for educational purposes by doing analyses on smaller networks.

NSDLib is a tool that integrates well-known network-related libraries into one framework dedicated to detecting sources of propagation, providing three core strategies to identify multiple origins within a network. The first strategy, Propagation Outbreak Detection, divides the network into distinct propagation regions, each potentially originating from a different source. This segmentation allows researchers to isolate propagation clusters, making it possible to trace each one back to its respective starting point. The second technique, Threshold-Based Selection, involves evaluating all nodes based on specific criteria, where nodes exceeding a designated threshold score are identified as potential sources. This approach offers flexibility, as the threshold can be adapted based on network characteristics and detection requirements. The third strategy, Two-Step Evaluation, is based on the method presented in [32] and applies a two-phase scoring system: nodes undergo an initial evaluation to select those with the highest scores, and a second evaluation then identifies top-ranked nodes among this subset, designating them as primary sources. It is worth mentioning that it provides also access to the implementation of propagation reconstruction techniques like SbRP or SHNI. `NSDLib` is available in the official Python packages repository and can be installed using the command: `pip install nsdlib.` It can be used in any Python-supported environment, like GUI applications, and scripts, or exposed as API and used by frontend code for JavaScript-based solutions [31].

Listing 4 provides sample code for performing source detection using the NSDLib library on a propagation graph derived from the Karate network, after removing nodes 10, 15, 20, and 33. First, the SbRP method is applied to reconstruct the propagation graph. Then, outbreaks are identified using the Leiden algorithm. The next step involves node evaluation within the identified outbreaks. Finally, source detection selection and evaluation are conducted to determine the likely origins of propagation.

```
import networkx as nx
import nsdlib as nsd

G = nx.karate_club_graph()
# propagation graph
IG = G.copy()
IG.remove_nodes_from([10,15,20,33])

real_sources = [0,8]
# 1. Graph reconstruction
EIG = nsd.reconstruction_sbrp(G, IG)
# 2. Propagation outbreaks detection
outbreaks = nsd.outbreaks_leiden(EIG)
# 3. Outbreaks evaluation and source detection
# detected_sources = []
for outbreak in outbreaks.communities:
    outbreak_G = G.subgraph(outbreak)
    nodes_evaluation = nsd.evaluation_jordan_center(outbreak_G)
    outbreak_detected_source = max(nodes_evaluation,
key=nodes_evaluation.get)
    detected_sources.append(outbreak_detected_source)
# 4. Propagation source detection evaluation
evaluation = nsd.compute_source_detection_evaluation(
    G=EIG,
    real_sources=real_sources,
    detected_sources=detected_sources,
)
print(evaluation)
```

Listing 4 A sample code executing propagation source detection with NSDLib

RPaSDT provides a graphical interface to access the techniques discussed, offering an intuitive platform for users to work with network analysis tools. Currently, it integrates solutions such as NetCenLib, CDLib, and NDLib, with planned expansions to incorporate a dedicated source detection solution, NSDLib. This integration will further enhance RPaSDT's functionality in identifying propagation sources. An example of using the RPaSDT's solution is shown in Figure 1. This product is not available via PyPI, ready to use binaries on Linux and Windows are available on the project GitHub repository.

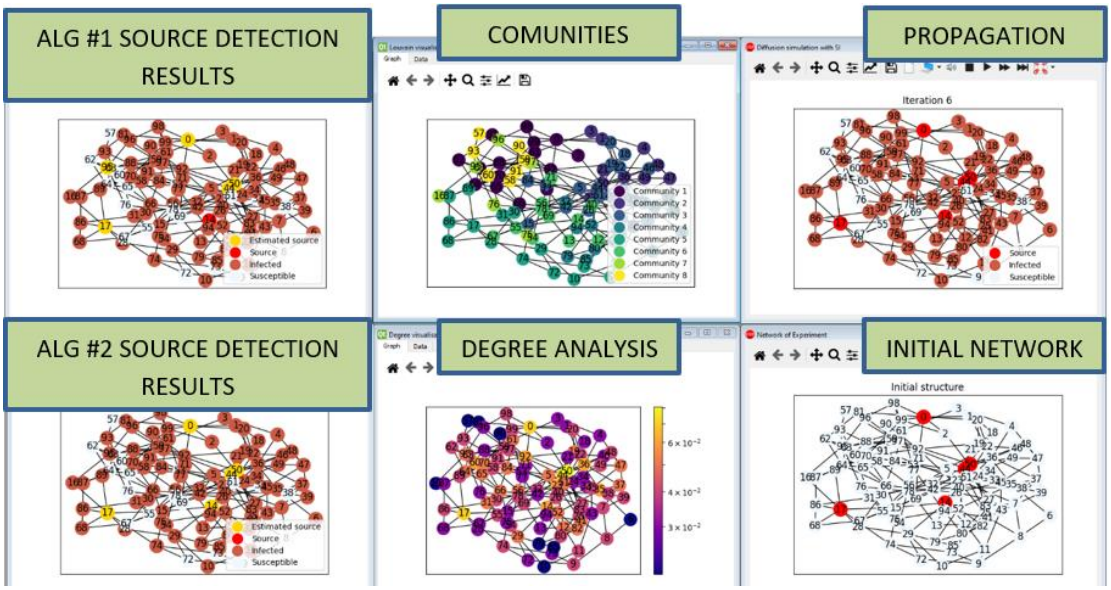

Figure 1 Sample usage of RPaSDT to perform source detection simulation experiment

Conclusions

This article provides a detailed review of recent tools for network analysis, emphasizing methods and applications that leverage the NetworkX package. It explores advancements in community detection, network propagation, source identification within social networks, and techniques for identifying influential nodes. The review highlights advanced algorithms that extend NetworkX's capabilities in dynamic contexts, offering researchers and practitioners practical insights into modern network analysis tools. Sample applications of the discussed libraries are included, encouraging readers to implement these tools effectively in their work.

References

- S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang, 'Complex networks: Structure and dynamics', Phys. Rep., vol. 424, no. 4–5, pp. 175–308, Feb. 2006, doi: 10.1016/j.physrep.2005.10.009.
- D. Hevey, 'Network analysis: a brief overview and tutorial', Health Psychol. Behav. Med., vol. 6, no. 1, pp. 301–328, Jan. 2018, doi: 10.1080/21642850.2018.1521283.
- C. C. Aggarwal and C. C. Aggarwal, Social network data analytics. New York: Springer, 2011.
- E. L. Platt, Network Science with Python and NetworkX Quick Start Guide: Explore and Visualize Network Data Effectively. Birmingham: Packt Publishing, Limited, 2019.
- R. Albert and A.-L. Barabási, 'Statistical mechanics of complex networks', Rev. Mod. Phys., vol. 74, no. 1, pp. 47–97, Jan. 2002, doi: 10.1103/RevModPhys.74.47.
- 'Top programming languages for data scientists in 2023'. Accessed: Jan. 15, 2024. [Online]. Available: https://www.datacamp.com/blog/top-programming-languages-for-data-scientists-in-2022
- 'igraph – Network analysis software'. Accessed: Jan. 15, 2024. [Online]. Available: https://igraph.org/
- D. Frąszczak, 'Detecting rumor outbreaks in online social networks', Soc. Netw. Anal. Min., vol. 13, no. 1, p. 91,

Jun. 2023, doi: 10.1007/s13278-023-01092-x.
- D. Frąszczak, ‘Fake News Source Detection – The State of the Art Survey for Current Problems and Research’, in Proceedings of the 37th International Business Information Management Association (IBIMA), Cordoba, Spain: International Business Information Management, 2021, pp. 11381–11389. doi: http://dx.doi.org/10.6084/m9.figshare.16545675.
- S. Shelke and V. Attar, ‘Source detection of rumor in social network – A review’, Online Soc. Netw. Media, vol. 9, pp. 30–42, Jan. 2019, doi: 10.1016/j.osnem.2018.12.001.
- P. Dey, S. Bhattacharya, and S. Roy, ‘A Survey on the Role of Centrality as Seed Nodes for Information Propagation in Large Scale Network’, ACMIMS Trans. Data Sci., vol. 2, no. 3, pp. 1–25, Aug. 2021, doi: 10.1145/3465374.
- S. P. Borgatti and M. G. Everett, ‘A Graph-theoretic perspective on centrality’, Soc. Netw., vol. 28, no. 4, pp. 466–484, Oct. 2006, doi: 10.1016/j.socnet.2005.11.005.
- P. Chebotarev and D. Gubanov, ‘How to choose the most appropriate centrality measure?’, ArXiv200301052 Phys., Mar. 2020, Accessed: May 22, 2021. [Online]. Available: http://arxiv.org/abs/2003.01052
- L. C. Freeman, ‘Centrality in social networks conceptual clarification’, Soc. Netw., vol. 1, no. 3, pp. 215–239, Jan. 1978, doi: 10.1016/0378-8733(78)90021-7.
- D. Frąszczak and E. Frąszczak, ‘NetCenLib: A comprehensive python library for network centrality analysis and evaluation’, SoftwareX, vol. 26, p. 101699, May 2024, doi: 10.1016/j.softx.2024.101699.
- ‘Welcome to NetCenLib’s documentation! — NetCenLib 0.2.1 documentation’. Accessed: Feb. 26, 2024. [Online]. Available: https://netcenlib.readthedocs.io/en/latest/index.html
- M. Viswanath, ‘ONTOLOGY-BASED AUTOMATIC TEXT SUMMARIZATION’.
- U. Brandes, ‘A faster algorithm for betweenness centrality*’, J. Math. Sociol., vol. 25, no. 2, pp. 163–177, Jun. 2001, doi: 10.1080/0022250X.2001.9990249.
- S. Fortunato, ‘Community detection in graphs’, Phys. Rep., vol. 486, no. 3–5, pp. 75–174, Feb. 2010, doi: 10.1016/j.physrep.2009.11.002.
- G. Rossetti, L. Milli, and R. Cazabet, ‘CDLIB: a python library to extract, compare and evaluate communities from complex networks’, Appl. Netw. Sci., vol. 4, no. 1, p. 52, Dec. 2019, doi: 10.1007/s41109-019-0165-9.
- D. Frąszczak, ‘Information Propagation In Online Social Networks - A Simulation Case Study’, in Proceedings of the 38th International Business Information Management Association (IBIMA), Cordoba, Spain: International Business Information Management, 2021. doi: 10.6084/m9.figshare.18974987.v1.
- J. Vallet, H. Kirchner, B. Pinaud, and G. Melançon, ‘A Visual Analytics Approach to Compare Propagation Models in Social Networks’, Electron. Proc. Theor. Comput. Sci., vol. 181, pp. 65–79, Apr. 2015, doi: 10.4204/EPTCS.181.5.
- I. Z. Kiss, J. C. Miller, and P. L. Simon, Mathematics of Epidemics on Networks: From Exact to Approximate Models, vol. 46. in Interdisciplinary Applied Mathematics, vol. 46. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-50806-1.
- G. Rossetti, L. Milli, S. Rinzivillo, A. Sîrbu, D. Pedreschi, and F. Giannotti, ‘NDlib: a python library to model and analyze diffusion processes over complex networks’, Int. J. Data Sci. Anal., vol. 5, no. 1, pp. 61–79, Feb. 2018, doi: 10.1007/s41060-017-0086-6.
- E. Frąszczak and D. Fraszczak, ‘A review of a website phishing detection taxonomy’, in Procedings of the 43th International Business Information Management Association Conference (IBIMA), Madrid, Spain, 2024. doi: 10.6084/m9.figshare.26345473.
- S. P. Borgatti and M. G. Everett, ‘A Graph-theoretic perspective on centrality’, Soc. Netw., vol. 28, no. 4, pp. 466–484, Oct. 2006, doi: 10.1016/j.socnet.2005.11.005.
- K. Zhu, Z. Chen, and L. Ying, ‘Catch’Em All: Locating Multiple Diffusion Sources in Networks with Partial Observations’, p. 7.
- Z. Wang, C. Wang, J. Pei, and X. Ye, ‘Multiple Source Detection without Knowing the Underlying Propagation Model’, Proc. AAAI Conf. Artif. Intell., vol. 31, no. 1, Feb. 2017, doi: 10.1609/aaai.v31i1.10477.
- D. Frąszczak and E. Frąszczak, ‘NSDLib: A comprehensive python library for network source detection and

evaluation', SoftwareX, vol. 28, p. 101950, Dec. 2024, doi: 10.1016/j.softx.2024.101950.
- D. Frąszczak, 'RPaSDT—Rumor Propagation and Source Detection Toolkit', SoftwareX, vol. 17, p. 100988, Jan. 2022, doi: 10.1016/j.softx.2022.100988.
- D. Frąszczak, 'NEFBDAA — .NET Environment for Building Dynamic Angular Applications', SoftwareX, vol. 19, p. 101163, Jul. 2022, doi: 10.1016/j.softx.2022.101163.