Fedivertex: a Graph Dataset based on Decentralized Social Media

Marc Damie
m.f.d.damie@utwente.nl
University of Twente
Enschede, The Netherlands
Inria
Villeneuve-d'Ascq, France

Edwige Cyffers
edwige.cyffers@cnrs.fr
Institute of Science and Technology Austria
Klosterneuburg, Austria

Abstract

Social network graphs are central to graph learning research, serving as standard benchmarks for algorithm evaluation. However, existing datasets focus mainly on mainstream social media platforms whose structures are shaped notably by algorithmic recommendations. This raises an important question: would alternative, decentralized social networks exhibit different properties?

We address this by studying the Fediverse; a collection of decentralized social networks (such as Mastodon and Lemmy). These platforms differ fundamentally from for-profit social media, notably in decentralization and absence of recommendation algorithms, which may yield distinct graph structures.

We introduce Fedivertex, a dataset of over 400 graphs from seven decentralized networks, collected weekly over six months. The dataset, released with a companion Python package to facilitate its use, supports research on temporal and structural aspects of decentralized social networks. In particular, we benchmark applications to decentralized machine learning and community detection.

CCS Concepts

• **Computing methodologies** → *Machine learning*; • **Mathematics of computing** → **Graph algorithms**; • **Security and privacy**; • **Networks** → **Social media networks**;

Keywords

Graph, Social Network, Machine Learning, Decentralized Systems

ACM Reference Format:
Marc Damie and Edwige Cyffers. 2026. Fedivertex: a Graph Dataset based on Decentralized Social Media. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3774904.3792868

1 Introduction

Social network graphs enable various graph learning tasks such as decentralized personalized learning, community detection, node classification, or edge prediction. While for-profit social media graphs are widely used and available (e.g. via SNAP [7]), datasets from non-profit social networks are missing. As algorithmic recommendations (which foster engagement on for-profit platforms) modify the graph structure by accelerating the triadic closure process and exacerbating inequality in popularity [9, 11], existing datasets do not cover the full spectrum of social graph structures.

In particular, we are missing graphs from the *Fediverse* — a contraction of *federation* and *universe* — which designates decentralized social networks interconnected via the ActivityPub protocol, an open W3C standard enabling interoperability among independently operated servers. In the Fediverse, anyone can host a server, called *instance*, with its own moderation policies, while interacting seamlessly with users from other instances. This setting is thus particularly interesting for trustworthy Machine Learning (ML), as many participants aim to address issues such as privacy, moderation, and algorithmic bias in opposition to mainstream social networks. The Fediverse offers thus more diversity, across platforms, instances and users, and is at a younger stage of development. Despite these unique features, existing work on the Fediverse only studied a single network or a subset of interactions between networks and have not release reusable datasets [1, 4, 11].

Our contributions: We introduce *Fedivertex*, the first dataset covering multiple software platforms within the Fediverse. Fedivertex covers seven Fediverse platforms, resulting in more than 400 graphs: 13 different graphs each with, *at least*, 24 weekly snapshots obtained through web crawls, and we currently continue data collection[1]. Compared to existing datasets, *Fedivertex* presents three distinctive features. *(i)* the decentralized nature of the Fediverse enables the construction of *instance-level* graphs that model interactions between Fediverse servers rather than *user-level* graphs (Figure 1). This granularity does not exist in centralized social media and is particularly suitable for simulating decentralized ML, where individuals are likely to have too little data and agency to personalize their participation whereas an instance moderator could do so. *(ii) Fedivertex* provides temporal evolution over at least 6 months, the longest temporal coverage we are aware of for social network graphs. *(iii) Fedivertex* graph topologies such as degree distributions differ substantially from those of mainstream social media, offering new opportunities to test algorithms under alternative dynamics.

We release a Python package, `fedivertex` to facilitate the dataset usage. We also publish the code [2] of our crawler and notebooks on Kaggle [3] demonstrating various tasks, including the decentralized ML and community detection benchmarks included in our paper.

2 Dataset presentation

Fediverse software is run by decentralized independent *instances* (i.e., servers), each one hosting a subset of *users*, and enforcing

This work is licensed under a Creative Commons Attribution 4.0 International License.
WWW '26, Dubai, United Arab Emirates
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792868

[1] We will continue adding new graphs to Fedivertex on a monthly basis as long as possible. The updates will only add new graphs and leave the existing graphs unchanged.
[2] https://github.com/MarcT0K/Franck
[3] https://kaggle.com/datasets/marcdamie/fediverse-graph-dataset/code

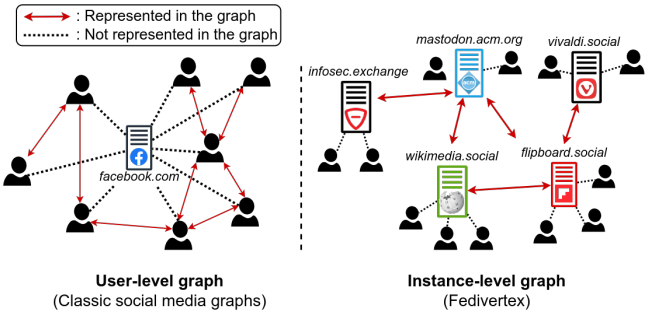

Figure 1: Comparison of user-level and instance-level graphs

its own internal rules, data storage and moderation. Communication across instances and platforms is seamlessly ensured by the ActivityPub protocol: content from a platform (e.g. a video from PeerTube) can be shared on another one (e.g. in a Mastodon toot), which is different from for-profit platforms which discourage interoperability. A user's data is stored on their respective instance but can be duplicated and cached on other instances to be accessible to other users. As a result, data persistence, confidentiality, and moderation depend on the instance.

We selected seven software platforms with large number of servers (**s**) and users (**u**) and diverse purposes: micro-blogging platforms with **Mastodon** (8.1M **u**, 7.8k **s**), **Misskey** (1.1M **u**, 957 **s**), **Pleroma** (66k **u**, 349 **s**), **Friendica** (8k **u**, 74 **s**); social news platforms with **Lemmy** (459k **u**, 317 **s**); book cataloging with **Bookwyrm** (24k **u**, 63 **s**); and video streaming with **Peertube** (329k **u**, 1k **s**); data extracted from fedidb on November 4.

2.1 Fedivertex graphs

Fedivertex focuses on interactions between *instances* within a given software platform rather than between users, as depicted in Figure 1, a granularity never studied before. This granularity is more realistic for training and deploying decentralized machine learning models, such as for spam detection or content moderation. Indeed, instance moderators are more likely than individual users to weigh the pros and cons of deployment and instances both aggregate more data and offer greater reliability than individuals. Instance-level graphs are naturally smaller graphs than user-level graphs for a given user-base, because each instance (i.e. node of the graph) hosts hundreds or thousands of users. For instance, the 8k nodes of Mastodon graph capture more than 8M users.

Fedivertex contains six types of graph collected for several platforms with different properties:

Federation graph (Mastodon, Misskey, Pleroma, Lemmy, and Bookwyrm): an instance u is connected to v if the two instances interact. This is an undirected unweighted dense graph.

Federation with blocks graph (Lemmy): same as the federation graphs but includes negative edges if u blocks v.

Follow graph (Peertube): u can follow v to let its users watch all the videos posted on v. This is a directed unweighted graph.

Active user graphs (Mastodon, Misskey, Pleroma): for each instance u, we collect its 10k most recently active users. For each of those users, for each user i they follow, we increase the edge weight

```python
from fedivertex import GraphLoader
loader = GraphLoader() # Will download and cache the dataset
G = loader.get_graph(software = "mastodon", graph_type = "active_user", index = 0)
# G contains the NetworkX graph of the active users graph at the first date of collection
```

Listing 1: Code example from the `Fedivertex` package

by 1 between u and the instance hosting i. Thus, the weight of the edge from u to v measures how much content seen on u originates from v. The graph thus contains self-loops as users follow others on the same instance, and it is a weighted directed graph.

Intra-instance graphs (Lemmy): the instance u is linked to v if a user of u has published a message on instance v. This graph is directed and very sparse.

Cross-instance graphs (Lemmy): two instances are connected if there exists a pair of users who published a message in the same "community", possibly on a third instance. This is an undirected graph, denser that the previous one.

The construction of this six graph types balances the will to unify as much as possible graph construction to allow comparison between platforms and the need to cope with the diversity of APIs and features across platforms (e.g. it is not possible to construct federation graph for Peertube). We have 6 graph types with 13 different graphs. Our GDPR declaration allowed for 6-month scraping maximum duration, so *Active users*, *cross-instance*, and *intra-instance* graphs, which require to scrap user data are available for 24 weeks, but the rest of the graphs are still collected weekly.

2.2 Fedivertex usability

Our dataset can be downloaded from Kaggle[4]. We also provide a Python package, `fedivertex`, which allows users to download and load the graphs in NetworkX format through an easy-to-use API (see Listing 1). Fedivertex package is documented and comes with several Kaggle notebooks demonstrating various graph analysis tasks. Finally, we follow the Gephi convention for graph encoding, allowing the graph CSV files to be opened in this software.

2.3 Graph construction via Web Crawling

We query the Fediverse Observer, whose open-source crawler runs daily and provides a curated list of Fediverse instances commonly accepted by the community. We then query each of these instances to compute the edges of the graph. Relying on the Fediverse Observer list helps minimize server load and allows us to benefit from existing curation. We release the code of our crawler, enabling extensions to other social media or other scraping parameters.

Scope: our crawlers cover all instances listed by the Fediverse Observer. The only instance-level filtering we apply consists in excluding instances that explicitly disallow crawling in their `robots.txt` file. Apart from this compliance requirement, we do not apply any additional filtering that could bias the resulting graphs.

Ethical concerns: The dataset collection raised two potential issues. *(i) Service disturbance:* we designed our crawler to minimize the server load; e.g., using a delay of 0.4 second between requests on a given instance. We use a clearly identifiable user agent. We respect the `robots.txt` policy of each instance. *(ii) Privacy:* Our

[4]https://doi.org/10.34740/kaggle/ds/6877842

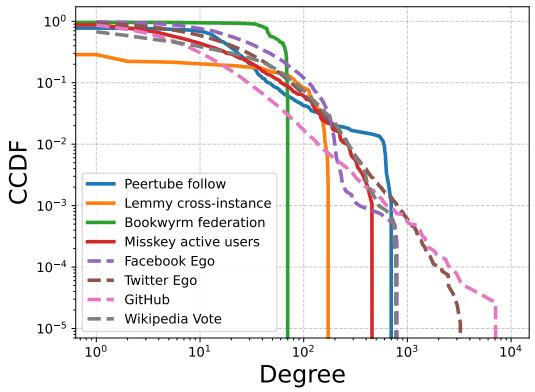

Figure 2: Complementary Cumulative Distribution Function (CCDF) of the degree for several Fedivertex graphs and other widely-used social network graphs.

dataset is instance-level and not user-level (see Figure 1), which is a better granularity for privacy. We only use public API endpoints, which do not require accounts on this platform. Finally, we also respect informal privacy practices (e.g., #NOBOT policy in user bios on Mastodon). The whole crawling process was supervised by the legal department to ensure GDPR compliance.

3 Dataset analysis

Fedivertex contains language labels for Peertube, Lemmy, Bookwyrm, and Misskey. For Peertube, the labels are directly extracted from the instance descriptions. For the others, we infer the label from the instance description. The labels can be easily used for label prediction tasks through our Python API. Other labels can also be extracted from our dataset such as video categories for Peertube.

All Fedivertex graphs except the active users graphs have a small-world σ between 1.14 and 4.45, active user graphs have $\sigma \simeq 21$. Such values ($\sigma > 1$) implies that a node is more likely to connect to their neighbors' neighbors and the average path length is small.

We compare these properties to properties of the most popular social network graphs from SNAP [7]: the Wikipedia Vote graph, the Twitter and Facebook Ego graphs, and the GitHub graph. For the small world properties, the SNAP dataset exhibit much higher σ: Facebook ego graph has $\sigma = 39.44$, Github graph $\sigma = 31.49$ and Wikipedia vote $\sigma = 519.64$. Differences are also significant for the average path length (from 1.18 to 3.95 in Fedivertex versus 3.25 and 3.7 for Facebook Ego and Wikipedia Vote) and of average degree (from 7 to 355 in Fedivertex versus 15 and 47).

Figure 2 shows that while GitHub and Twitter exhibit the classical power-law decay over much of the support of the distribution, consistently with preferential attachment networks, the degree distribution of Fedivertex is more diverse. We note some similarities between the Facebook Ego graph and the Peertube follow graph, with a smooth concave decay followed by a short power-law tail for the most popular users. Most of the graphs, however, exhibit only the a concave curve, which suggests that the attachment dynamics in Fedivertex differ from those in traditional social networks, allowing for a smoother distribution of node importance.

4 Applications

This section demonstrates the utility of Fedivertex on two applications: decentralized ML and community detection. Fedivertex can be applied to many graph learning tasks. We also introduce a new task called *defederation*; which aims to predict which edges or nodes will be removed from the graph at the next iteration.

4.1 Decentralized ML

Decentralized ML is a ML paradigm in which each node possesses a local dataset and some computational resources, and collaborates with other participants by exchanging messages through peer-to-peer communication during the training of a global, potentially personalized, ML model. The network topology has an impact on the learning dynamics [6], particularly in the presence of data heterogeneity [5] and in terms of privacy guarantees [8].

Decentralized ML has numerous real-world use cases, as nodes can represent entities as diverse as healthcare institutions or sensors distributed across installations, but also decentralized social networks [2]. In such cases, the graph often captures complex and diverse relationships and communities, which explains its popularity in ML. Social network dynamics also raise interesting questions related to polarization and time-evolving properties of the graph [10]. Fedivertex is then well adapted to experiments testing decentralized ML, as they provide a credible use-case scenario.

We illustrate this by reproducing experiments from [3]. This paper proposes training a global model with differential privacy via a random walk on the communication graph: at each step, a stochastic gradient is computed on the local dataset of the current node and sanitized by adding Gaussian noise. The paper connects the privacy guarantees and the communicability of the graph, showing that nodes close to each other have higher privacy budget than far apart ones. Figure 3 shows that real-world Fedivertex graphs can be more challenging than synthetic ones in terms of convergence, as Mastodon active-user graph exhibits slower convergence than synthetic graphs with the same number of nodes. Nodes with low centrality, typically connected by only a single edge, might make it harder for the random walk to visit all the nodes frequently enough within the chosen number of steps compared with more regular graphs tested in the original paper.

4.2 Community detection

The Fediverse social networks are used by many communities that might overlap. The same communities might also use several of the Fediverse software platforms. To illustrate the feasibility of community detection with our dataset, we use the official languages of each instance as ground truth labels, and run three algorithms: Louvain, Greedy Modularity, and Label Propagation on the Peertube follow graph and the Misskey active user graph. To avoid many unique labels, we keep only the 5 most represented languages in each graph and ignore others. Results are in Table 1.

We assess the quality of clustering using the Adjusted Rand Index (ARI) and the Adjusted Mutual Information (AMI). The Rand index corresponds to the proportion of node pairs belonging to the same cluster that are classified as such (i.e., the sum of true positives for all classes) over all node pairs, and the adjusted version corresponds to normalization with respect to a random clustering. Similarly, the

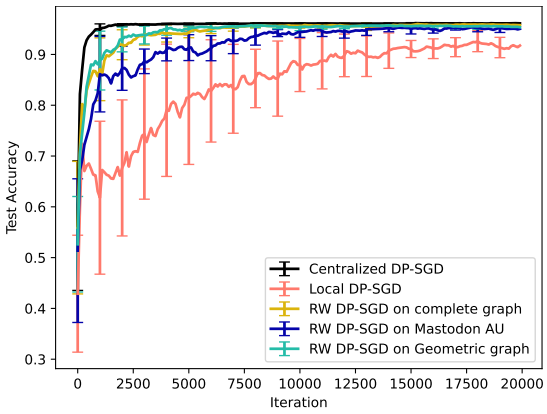

Figure 3: Numerical experiment reproducing the private logistic regression of [3] with Fedivertex graphs, using the Houses dataset for graph with 3852 nodes

Table 1: Community detection on Fedivertex

Graph	Algorithm	ARI	AMI	Modularity
Peertube follow	Louvain (100 runs)	0.055	**0.123**	**0.217**
	Greedy Modularity	**0.061**	0.110	0.209
	Label Propagation	0.008	0.029	0.003
Misskey A.U.	Louvain (100 runs)	0.097	0.250	**0.611**
	Greedy Modularity	0.014	0.165	0.513
	Label Propagation	**0.229**	**0.255**	0.027

AMI score corresponds to the mutual information between the ground truth and predicted labels, adjusted for chance. Finally, we report modularity for each clustering; a metric for unsupervised clustering that assesses the inherent quality of the partitioning. This suggests that other labels might be suitable as well for clustering the graphs. No method dominates in this benchmark, highlighting the diversity of Fedivertex graphs.

Other Fedivertex graphs or different labels can be used for community detection. For example, each Peertube instance has a list of video categories within a standardized list of 18 categories.

4.3 New links and defederation prediction

Fedivertex supports temporal graph analysis, as we release weekly snapshots for each graph. Understanding temporal changes in graphs is seen as a major challenge in graph learning, in particular for social networks.

In Fedivertex graphs, edges or nodes are sometimes deleted, often during quick variation, a phenomenon that sometimes dominates edge variation. This process is called *defederation* and can help to keep decentralization over time by avoiding a single star graph, but could also weaken the platform growth. So far this behavior has not been studied as it rarely appears in mainstream social networks. As new edge prediction is often studied, predicting jointly creation and deletion could bring a better understanding of graph dynamic.

Formally, let $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ be the graph collected at time t and compare it to the graph $\mathcal{G}_{t'} = (\mathcal{V}_{t'}, \mathcal{E}_{t'})$ collected at $t' > t$. Let $\mathcal{G}_c$ be the subgraph induced by $\mathcal{V}_c = \mathcal{V}_t \cap \mathcal{V}_{t'}$; the goal is to predict all edges either created ($e \in (\mathcal{V}_c \times \mathcal{V}_c) \cap (\mathcal{E}_{t'} \setminus \mathcal{E}_t)$) or deleted ($e \in (\mathcal{V}_c \times \mathcal{V}_c) \cap (\mathcal{E}_t \setminus \mathcal{E}_{t'})$) between t and t'. The possible new edges lie in $(\mathcal{V}_c \times \mathcal{V}_c) \setminus \mathcal{E}_t$, whereas the deleted ones are in $\mathcal{E}_t$, a set significantly smaller for sparse graphs. Similarly, one could predict nodes drops (i.e. predict $\mathcal{V}_t \setminus \mathcal{V}_{t+1}$ given $\mathcal{G}_t$). In particular, reliable prediction could help detect instances which stopped running because of technical problems despite being active in the graph, and possibly provide technical help to such instances. We report defederation in this notebook[5], showing that deleted edges or nodes tend to be persistent, and that using simple methods inspired by edge prediction methods (Adamic-Adar, Common Neighbors and Jaccard metrics) performs slightly better than random, leaving room for the development of better methods.

Acknowledgments

This work has supported by the Agence Nationale de la Recherche (ANR) in the context of the REDEEM project (ANR-23-PEIA-005), the Netherlands Organization for Scientific Research (NWO) in the context of the SHARE project [CS.011], and the Austrian Science Fund (FWF) [10.55776/COE12].

References

[1] Vibhor Agarwal, Aravindh Raman, Nishanth Sastry, Ahmed M Abdelmoniem, Gareth Tyson, and Ignacio Castro. 2024. Decentralised Moderation for Interoperable Social Networks: A Conversation-Based Approach for Pleroma and the Fediverse. *Proceedings of the International AAAI Conference on Web and Social Media* 18, 1 (May 2024), 2–14. doi:10.1609/icwsm.v18i1.31293

[2] Haris Bin Zia, Aravindh Raman, Ignacio Castro, Ishaku Hassan Anaobi, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. 2022. Toxicity in the Decentralized Web and the Potential for Model Sharing. *Proc. ACM Meas. Anal. Comput. Syst.* 6, 2 (June 2022), 35:1–35:25. doi:10.1145/3530901

[3] Edwige Cyffers, Aurélien Bellet, and Jalaj Upadhyay. 2024. Differentially Private Decentralized Learning with Random Walks. In *Proceedings of the 41st International Conference on Machine Learning*, Vol. 235. PMLR, 9762–9783.

[4] Shiori Hironaka, Mitsuo Yoshida, and Kazuyuki Shudo. 2024. Comparing User Activity on X and Mastodon. In *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 2967–2972. doi:10.1109/bigdata62323.2024.10825212

[5] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. 2020. The Non-IID Data Quagmire of Decentralized Machine Learning. In *Proceedings of the 37th International Conference on Machine Learning*, Vol. 119. PMLR, 4387–4398.

[6] Hanna Kavalionak, Emanuele Carlini, Patrizio Dazzi, Luca Ferrucci, Matteo Mordacchini, and Massimo Coppola. 2021. Impact of Network Topology on the Convergence of Decentralized Federated Learning Systems. In *2021 IEEE Symposium on Computers and Communications (ISCC)*. 1–6.

[7] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford large network dataset collection.

[8] Abdellah El Mrini, Edwige Cyffers, and Aurélien Bellet. 2024. Privacy Attacks in Decentralized Learning. In *Proceedings of the 41st International Conference on Machine Learning*, Vol. 235. PMLR, 36419–36433.

[9] Jessica Su, Aneesh Sharma, and Sharad Goel. 2016. The Effect of Recommendations on Network Structure. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1157–1167. doi:10.1145/2872427.2883040

[10] Riitta Toivonen, Lauri Kovanen, Mikko Kivelä, Jukka-Pekka Onnela, Jari Saramäki, and Kimmo Kaski. 2009. A comparative study of social network models: Network evolution models and nodal attribute models. *Social Networks* 31, 4 (Oct. 2009), 240–254. doi:10.1016/j.socnet.2009.06.004

[11] Matteo Zignani, Sabrina Gaito, and Gian Paolo Rossi. 2018. Follow the "Mastodon": Structure and Evolution of a Decentralized Online Social Network. *Proceedings of the International AAAI Conference on Web and Social Media* 12, 1 (June 2018), 541–550. doi:10.1609/icwsm.v12i1.14988

[5] https://www.kaggle.com/code/marcdamie/do-fediverse-instances-disappear-forever