An approach for analysing the impact of data integration on complex network diffusion models

Nevin, J.; Groth, P.; Lees, M.

DOI
10.1093/comnet/cnad025

Publication date
2023

Document Version
Final published version

Published in
Journal of complex networks

Journal of Complex Networks (2023) **4**, Advance Access Publication on 21 July 2023
https://doi.org/10.1093/comnet/cnad025

An approach for analysing the impact of data integration on complex network diffusion models

James Nevin†, Paul Groth and Michael Lees
Informatics Institute, University of Amsterdam, Amsterdam 1098 XH, Netherlands
†Corresponding author. Email: j.g.nevin@uva.nl

[Received on 12 December 2022; editorial decision on 13 March 2023; accepted on 9 June 2023]

Complex networks are a powerful way to reason about systems with non-trivial patterns of interaction. The increased attention in this research area is accelerated by the increasing availability of complex network data sets, with data often being reused as secondary data sources. Typically, multiple data sources are combined to create a larger, fuller picture of these complex networks and in doing so scientists have to make sometimes subjective decisions about how these sources should be integrated. These seemingly trivial decisions can sometimes have significant impact on both the resultant integrated networks and any downstream network models executed on them. We highlight the importance of this impact in online social networks and dark networks, two use-cases where data are regularly combined from multiple sources due to challenges in measurement or overlap of networks. We present a method for systematically testing how different, realistic data integration approaches can alter both the networks themselves and network models run on them, as well as an associated Python package (*NIDMod*) that implements this method. A number of experiments show the effectiveness of our method in identifying the impact of different data integration setups on network diffusion models.

Keywords: dark networks; online social networks; data integration; diffusion models.

1. Introduction

With the continuous advances in data storage and computational power, the opportunities for complex network research are rapidly expanding [1]. There is a growing wealth of relational data available in online repositories for researchers to access and study [2]. Often the value of these data go beyond the original study and in some cases may be reused multiple times and in some cases decisions at collection may be made with different goals than those (re)using the data. Additionally, it is common practice to combine multiple sources to create more comprehensive data sets or to address completely new research questions. In all cases, the robustness, reliability and value of the complex network research will depend on the quality and manner in which the data are collected, merged and reused [3]. Clearly, there is a growing need to address and understand the ramifications of data collection and use in complex network research.

Alongside the study of networks themselves, researchers are often interested in the application of models that can run on these complex networks. Some models can be used to 'complete' the network by estimating unobserved information, such as through a collective classification model [4]. Other network models can be used to understand spreading processes on a network, such as the spread of a disease through a community or the transmission of information through a friend group. In some cases, both infectious diseases and information can be modelled; for example, Zhang *et al.* [5] modelled the coupled diffusion of information and the pandemic of COVID-19. The accuracy and reliability of these different

types of network models are totally dependent on the underlying network structure, and thus the data used to create the complex network.

As a result, data handling concerns, such as sourcing, cleaning and integration, can have an impact on the network itself, its analysis and any network model that is ultimately applied. Network models can play important roles in policy and decision making [6, 7]. Hence, there is a clear need for having network data that are applicable and accurate. However, fully addressing all concerns in complex network data to create a perfect snapshot of reality is both infeasible and, in many cases, unnecessary. This depends on the purposes for which the network is being analysed or the network model being applied.

There is currently very little work in the literature addressing the impact of data handling on network outcomes and even less testing the extent to which data decisions can impact downstream network models, despite the sensitivity that these models can have [8]. Some authors have considered how data mismeasurement can impact on network properties [9–11]. However, as mentioned before, many of the modern day data concerns are more than just measurement errors. Instead, they are decisions about how to clean and combine various different data sources. There is a lack of both a refined method and tools for analysing how these data handling concerns impact on networks and any associated models.

In this article, we propose a novel method for systematically testing the sensitivity of complex network data sets and network models to data handling decisions. The particular data handling problem we focus on is data integration, which is the process of creating a single, unified view of multiple data sources [12]. This is a common data handling concern in complex network research, particularly in fields like online social networks [13] and dark networks [14]. For a given diffusion model and associated complex network data sets that need to be combined, different real-world data integration setups are applied to the complex networks. These result in a set of integrated networks (each corresponding to an integration setup), on which the diffusion model can be run and outputs compared. Furthermore, we offer an implementation of our method in the form of a Python software package, *NIDMod (Network Integration and Diffusion Modeller)*.[1]

We apply our method to three different types of diffusion process: *information transfer*, *information spread* and *label propagation*. These types cover a wide range of diffusion cases. For each type of process, we perform an experiment wherein we choose a model and associated data sets from a canonical domain for such a diffusion process, and apply our method. We highlight which data integration decisions each diffusion model is sensitive to. We compare and contrast the effects of the data integration decisions on the different diffusion processes with each other.

The contributions of this article are as follows:

1. A method for testing network model sensitivity to data integration setups.
2. An implementation of this method as a Python software package.
3. Results of experiments for three types of diffusion process, which highlight the method's ability to identify sensitivities.

We will generally focus on network diffusion models (spreading processes on networks), but this method could just as well be applied to other forms of network model.

The remainder of this article is structured as follows: Section 2 offers a literature review that highlights the problem of data integration in online social networks and dark networks; Section 3 introduces our

[1] https://github.com/jim-g-n/nidmod

method; Section 4 shows the application of our method to various data sets and diffusion models and finally, Section 5 summarizes our findings.

2. Literature review

In this section, we highlight two research fields where the problem of data integration affects complex network models: online social networks and dark networks. The difficulty in solving this problem in these domains emphasizes the need for methods to measure the effect of different integration solutions.

2.1 *Online social networks*

Online social networks provide an avenue for researching real-world complex networks with a large number of nodes and generally easy access to the data of the network. These social networks exhibit many of the theoretical properties of complex networks, e.g. small world and scale-free [15].

One of the key challenges in social network analysis (SNA) is the problem of linking the same user across multiple online social networks and as such has been an area of significant study. Shu *et al.* [13] provide a review of the problem. They note that there is a significant overlap in the userbase of various online social networks, such as Instagram and Facebook. Linking user identities across networks offers a variety of opportunities. Of particular interest is the ability to model information diffusion more accurately, which could be used, for example, for more effective advertising or dissemination of important health information.

Reference [13] describes a general framework for user identity linkage that corresponds closely to traditional data integration frameworks. Online profile features such as usernames, locations and education, function similarly to attributes in traditional databases, with different measurements possible for measuring similarity. They also suggest the use of network features to measure similarity, such as the proportion of neighbours that overlap. Matches are identified using similar supervised, semi-supervised and unsupervised algorithms as might be used in traditional data integration.

Finally, Ref. [13] discusses the limited data sets available for this topic. While there are many individual social networks available for research, there are no agreed benchmark data sets for matching users across different online social networks. The authors suggest the use of full synthetic and partial synthetic (where one real-world network is split into two subnetworks using node sampling or edge sampling) networks.

Noor *et al.* [16] introduce a methodology for identifying real-world underlying social networks using multiple online social networks. Different online social networks data sets are combined using traditional cleaning and integration techniques to create a single, fused network. Strengths of ties within this network are based on the number of individual networks in which the same tie exists. The authors test their methodology on data collected from Facebook, emails and LinkedIn. They suggest that these (and other) online social networks can be used to identify real-world, dark networks such as criminal or terrorist networks.

These various challenges and methodologies for conducting research into online social networks highlight the opportunity for applying a systematic approach to the problem of entity resolution in these networks.

2.2 *Dark networks*

Dark networks are complex social networks representing networks that operate outside of the law [17]. These are networks related to areas such as terrorist and crime networks, as well as illegal trades such as

drugs and arms. As the goals of these networks represent a threat to society, much of the research focus is on the disruption of such networks.

Since the terrorist attacks of 9–11, there has been an increase in the plans and efforts to counter terrorism. However, there are still major challenges that need to be addressed, particularly in data collection and analysis [14]. These challenges include issues like scalability (dealing with the massive amounts of information that are being collected), data collection (data can vary by source and type; dark networks data are difficult to collect and are dynamic) and data analysis (data are being collected more quickly than they can be analysed; sorting through the data to answer the questions of interest).

Many sources attest to these challenges in data collection for dark networks and offer potential solutions. Shaikh and Jiaxin [18] explain how these networks are covert, with a willingness to trade efficiency for secrecy. Gera *et al.* [19] identify issues like incompleteness, fuzzy boundaries and dynamic behaviours in constructing dark networks. They suggest creating multilayered terrorist networks to alleviate some of these issues. Another challenge is the mismatch between those analysing the data and those collecting the data (like police) [20]. This impacts how collection boundaries are defined, and compromises often need to be made. Morris and Deckro [21] highlight risks like unknown amounts of missing or corrupt data, or intentionally false data. A point of interest here is disambiguation of actor aliases—many actors in these dark networks use various aliases for security reasons, and these duplicates need to be identified when constructing the network.

Roberts [14] notes that data integration is central to tackling these challenges of data analysis in dark networks. The data are from various information sources, with differing levels of abstraction. They need to be integrated in order to create a consolidated picture of the threat environment, which can then be analysed. Reference [14] identifies three approaches to data integration in the literature: *discipline-centric*, *place-centric* and *virtual*. Discipline-centric integration 'evolved from different methodological approaches to data analysis'. SNA falls in this category, where relationship types from different sources are combined into one aggregate network. There are numerous examples of studies into dark networks that highlight this approach. We detail five such examples below.

Xu *et al.* [22] evaluate the Global Salafi Jihad terrorist network and how it evolves over time and its robustness. They explain how this network is built from a variety of different sources:

- Transcripts of court proceedings.
- Reports of court proceedings.
- Corroborated information from people with direct access.
- Uncorroborated statements from people with access.
- Statements from people who heard the information second hand.

These sources offer different levels of reliability, and needed to be carefully examined and decisions made on how to integrate them. They collected various attributes of the criminals, such as geography, occupation and mental illnesses.

Cunningham *et al.* [23] provide a detailed analysis on the *Fuerzas Revolucionarias de Colombia* (FARC) criminal network in Columbia. They go in-depth about the organization, its history and organizational structure. The network was built using a variety of different data sources: official documentation from the Colombian government, academic works on the group and open-source media reports on FARC activity. They collected both relational and attribute data of FARC members and used different sources to fill different gaps in the data. They explain how the data are a mix of one-

and two-mode network data with different relations and how these data were aggregated. Following the construction of this network, they analysed its structure to determine key players and potential leaders.

Magouirk *et al.* [24] highlight their findings on their Global Transnational Terrorism (GTT) project. They identify a major problem for studying terrorism today: a lack of strong, quantitative data that are freely available for research. They created a GTT Database to record such information. They collected detailed information on biographical and socioeconomic data such as names and nationalities, and network connection data. Ties in the database are collected from various sources, including: letters, photos, government documents and court testimony, among others. Different sources offer different levels of reliability and the authors needed to take this into account.

Everton [25] explains that actors in terrorist networks are usually involved in more than one type of relation (e.g. financial and kinship), and aggregating these various ties is important for accurately testing disruption strategies. They also explain the creation of the so-called *Noordin Top Terrorist Network*, a terrorist network based in Indonesia. This network was constructed using an International Crisis Group publication, 'Terrorism in Indonesia: Noordin's Networks'. Relational and attribute data for individuals identified in the report were coded to create a terrorist network, on which the author tested various SNA approaches.

Duijn *et al.* [26] perform experiments on a criminal network built from two data sources and their respective networks. The first source is police arrest data. In this data set, they have access to information on criminals who were arrested and build a network based on which criminals were arrested together. In this case, the information is reliable, but has a bias towards certain individuals in the network, as it is more likely for less senior/important members of the network to be arrested. The second data source is criminal informants and anonymous reports. This data set is less reliable than the previous, as there is room for informants to be dishonest or hide information. However, it is also more likely to find information on senior criminals than with the arrest data. The two networks built from each of the data sources represent the same underlying network, but both are constructed in a biased way. Reference [26] integrated these two networks into one to create an overall less biased network.

The examples described above show the challenges with constructing dark networks. However, the construction of these networks is often not the final goal. Research into dark networks is done primarily with the intention of disrupting them, as they represent dangers to society. Reference [26] offers an overview of the disruption of criminal networks. They detail various approaches to disrupting these networks, with two main focus areas: social capital disruption and human capital disruption. Social capital disruption targets individuals in the network based on their placement within the network, e.g. targeting individuals with a high centrality. Human capital disruption targets individuals based on the role they play, attempting to target those who have a critical job in the criminal chain and are not easily replaceable. The effectiveness of a network disruption is measured by how easily the network is able to adapt and how much information flow within the network is disrupted, both of which are analysed using network models.

Due to the complexity of data collection for dark networks, data integration is thus of critical importance. Furthermore, models run on these dark networks also play an important role, as they can test the ability of the networks to spread information and adapt. Thus, understanding the impact of data integration decisions on network model output is vital.

In summary, online social networks and dark networks are two examples of research areas where data integration plays a crucial role. In both fields, there is a lack of a well-defined method for systematically testing the effect the integration can have on the networks and any diffusion model run on them.

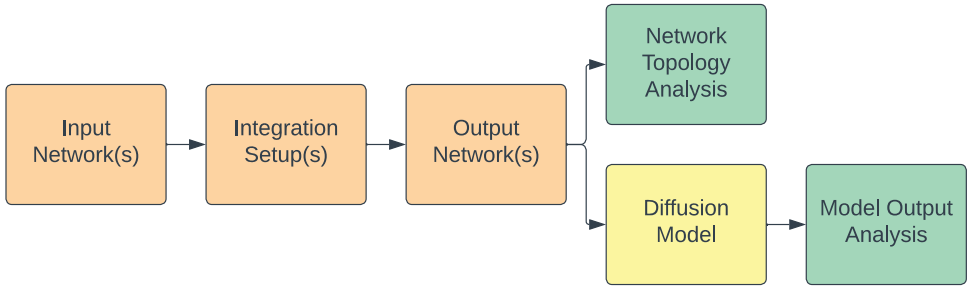

FIG. 1. Overview of method. Different compartments of the method are highlighted in different colours: integration steps are in orange, the diffusion step is in yellow and analysis steps are in green.

3. Method

The issues of data integration for complex networks described in the previous section highlight the need for a method for systematically testing the sensitivity of a given diffusion model and data sets to different data integration setups. Here, we present a method for executing all steps of a data-driven complex network diffusion procedure with multiple different real-world data integration setups. In the method, the results of the diffusion model on the different integrated networks are compared. This identifies which data integration decisions the specified model is sensitive to. The method consists of three compartments: *integration*, *diffusion* and *analysis*. The steps are outlined below and in Fig. 1.

Integration

1. One or many networks with associated node attributes are used as input.
2. One or many integration setups are defined.
3. These different integration setups are each applied to the network(s) to generate a set of output networks.

Diffusion

4. The diffusion model is run on each of these output networks.

Analysis

5. The properties of the output networks are evaluated and compared.
6. The final results of this diffusion model on the different networks are compared.

Our Python package, *NIDMod*[2], implements the method as described above. The package is also compartmentalized into the three different parts (integration, diffusion and analysis), each of which can be executed independently or jointly. The remainder of this section explains these compartments further and describes the Python integration of the method.

[2] https://github.com/jim-g-n/nidmod

3.1 *Integration*

Our method uses canonical data integration techniques like indexing, comparing, classifying and clustering. These terms are fully defined in Appendix A.

In our method, an initial set of unintegrated networks with associated node attributes are provided as input. Multiple different data integration setups (indexing, comparison, etc.) are defined on the attributes. These integration setups are applied to the original networks, which produce a set of integrated networks, each corresponding to one of the integration setups.

3.2 *Diffusion*

In the subsequent step of our method, the different integrated networks are used as input to a network diffusion model. Network diffusion models are used to model spreading processes on a complex network. This could include things like the spread of a disease, or information or failures in a system. The structure of the network influences how the spread plays out, with spreading often happening over the edges in the network.

The network diffusion model of interest is run on each of the integrated networks and the results recorded.

3.3 *Analysis*

In the analysis compartment, two sets of comparisons are made. Firstly, the properties of the integrated networks are compared. This can include different network metrics. Secondly, the results of the diffusion model on the different integrated networks are compared, as the changes in topology can heavily influence model output. These analysis results will indicate which data integration decisions most heavily affect the network itself and diffusion model output.

3.4 *Implementation*

The package implementation of our method builds on prior work where there are a number of existing toolkits for performing data integration [27–29].[3] *NIDMod* specifically uses the popular Python Record Linkage Toolkit [30]. It is Python-based and built to function directly using pandas—a common data manipulation package used in complex networks. It implements all standard data integration techniques and is extensible. This toolkit allows users to identify potential matched entities across tables or duplicate entities within a single table. The Python Record Linkage Toolkit (and, by extension, *NIDMod*) directly applies the indexing, comparing and classifying steps described above.

All standard indexing approaches like blocking are included directly in *NIDMod*. Furthermore, users are able to define their own indexing algorithms using the Python Record Linkage Toolkit ontology. Details of how to do this, as well as information on the aforementioned algorithms, are available here https://recordlinkage.readthedocs.io/en/latest/ref-index.html.

For comparison, users select attributes to be compared (one from each table in the case of combining tables) and decide how to determine whether attributes match. For example, one might compare first name attributes using a Levenshtein string similarity. Each pair of attributes chosen results in a feature that can be used in classification. Numerous different algorithms are available

[3] https://github.com/J535D165/data-matching-software

(https://recordlinkage.readthedocs.io/en/latest/ref-compare.html), as well as the ability to create user-defined algorithms.

Both supervised and unsupervised classification algorithms are supported, depending on whether training data are available. Once again, there is the possibility of creating user-defined algorithms (https://recordlinkage.readthedocs.io/en/latest/ref-classifiers.html).

An extension offered by *NIDMod* over the Python Record Linkage Toolkit is the option for users to define clustering algorithms that take as input a graph/set of graphs and a set of matched nodes and return an integrated graph. These completes the final step of the data integration process specifically for complex networks.

In *NIDMod*, users define a set of data integration setups, each involving indexing, comparing, classification and clustering steps. These multiple setups can then be applied to the input networks to create a set of integrated networks, which can be independently analysed or fed into the network diffusion model part of the package.

We employ an existing network diffusion package to execute diffusion models on the integrated networks. As with data integration, there are a number of available packages for executing them [31–33]. We use NDlib [34, 35], as it is quite versatile and popular package. The network diffusion part of *NIDMod* allows users to define a custom diffusion model and run it on a set of networks.

Users can define their own custom diffusion model exactly as is done in NDlib, with some classical models like the Susceptible-Infected-Removed [36] and Threshold [37] already built-in. We extend upon NDlib by allowing users to define their models independently of networks, which allows for the same defined model to be repeatedly applied to a number of different networks, such as the set of integrated networks that can be created. The model can be run on each of these networks and the results analysed and compared.

Finally, *NIDMod* provides functionality to extract some key statistics for the various networks' topologies and average measures for the diffusion model run on them, in order to assess the impact that the different integration setups have had.

4. Experiments

In this section, we perform a series of experiments that highlight the effectiveness of the method in measuring the impact that data integration setups can have on downstream network diffusion models.

In each experiment, we consider a different diffusion modelling problem and an associated specific domain in which such diffusion processes often arise. We select a diffusion model and data sets that model the given diffusion process and domain, and test the model's sensitivity to different data integration setups. In the first experiment, we consider *information transfer*, which often arises in dark networks. In the second experiment, we consider *information spread*, which commonly occurs in online social networks. Finally, we look at *label propagation*, which can be seen in coauthor networks. For each experiment, we describe the goal of the experiment, the choice of domain, the diffusion model and data sets used, the experiment setup and the results. All experiments are performed using *NIDMod*, with examples of the code included in Appendix B.

In all of the experiments, we have access to the ground truth data. When addressing these data integration problems in reality, one would have at most some training data for fitting a supervised algorithm. As a result, someone applying this method to a given diffusion model would need to interpret the results in terms of sensitivity of the model to the different data integration setups tested, and perhaps decide to delve more deeply into the data if they observe very high sensitivities.

4.1 *Information transfer*

4.1.1 *Goal* In information transfer models on complex networks, packets of information are sent from specific source nodes to specific target nodes. How well information can be transmitted through the network can be heavily dependant on the structure of the network itself. As data integration can have an effect on network structure, we wish to test to what extent information transfer can be influenced by different integration setups.

4.1.2 *Domain* As mentioned in Section 2.2, dark networks are often concerned with balancing the ability to spread information and retain robustness to network attacks. The ability of a network to transfer information is thus of crucial importance for illicit activities.

4.1.3 *Model* Czaplicka *et al.* [38] investigate the effect that noise can have on information transfer in hierarchical networks. Their algorithm tests how well packets of information can be transferred from one source node to a specific target node (Algorithm 1). This is exactly what might be of concern to members of a dark network: can messages reach target individuals through the network? We use this algorithm to test how effectively information is transferred in dark networks built using different integration setups.

Algorithm 1 Packet transfer dynamics [38].

1. A sender node s and a receiver node $r \neq s$ are randomly selected.
2. In consecutive time steps $t = 0, ..., T$, the packet travels from node to node. Therefore, packet a arrives at node i in time step t.

   (a) If $t < T$ then the packet algorithm jumps to step (b). Otherwise the algorithm loop is completed and the packet is considered undelivered.

   (b) If in the nearest neighbourhood of node i there is a receiver, r, for packet a then with a probability $1 - q$ packet a is delivered to its receiver. If not, then the algorithm goes to step (c).

   (c) If in the nearest neighbourhood of node i there is a node belonging to the same community as the receiver, r, then with a probability $1 - q$, packet a is moved to this node. Otherwise, the algorithm moves to step (d).

   (d) Packet a is moved to a randomly chosen node from its nearest neighbourhood.

   (e) If packet a finds its destination, the loop is completed. If not, then the navigation algorithm returns to the starting point (a).

4.1.4 *Data* Due to the challenges of sourcing unintegrated dark network data, we apply this information transfer dynamics model to partially synthetic data. This data set is accessible through the Python Record Linkage Toolkit, originally offered by Christen [39]. These data contain information on persons, including names, addresses and dates of birth. Table 1 shows the first few rows of the data set. In total, there are 1000 entries, with 500 original entries and 500 duplicates, for which we have the true matches.

TABLE 1 *First entries in FEBRL data set*

	Given_name	Surname	Street_number	Address_1	Address_2	Suburb	Postcode	State	Date_of_birth	Soc_sec_id
rec-223-org	NaN	Waller	6.0	Tullaroop street	Willaroo	St James	4011	WA	19081209.0	6988048
rec-122-org	Lachlan	Berry	69.0	Giblin street	Killarney	Bittern	4814	QLD	19990219.0	7364009
rec-373-org	Deakin	Sondergeld	48.0	Goldfinch circuit	Kooltuo	Canterbury	2776	VIC	19600210.0	2635962
rec-10-dup-0	Kayla	Harrington	NaN	Maltby circuit	Coaling	Coolaroo	3465	NSW	19150612.0	9004242
rec-227-org	Luke	Purdon	23.0	Ramsay place	Mirani	Garbutt	2260	VIC	19831024.0	8099933

The entries have been labelled based on whether they are duplicates or originals, but we do not use this information in defining our integration setups.

In order to create a complex network associated with this data set, we generate a synthetic Barabási–Albert graph [40]. This graph is created with 500 nodes and 2 edges attached from new nodes to existing nodes. This represents the 'true' graph, where each node is randomly chosen to correspond to one of the original entries in the data set. We then add 500 nodes corresponding to the duplicate entries. For each of the original nodes, its edges are randomly reassigned to a corresponding duplicate node. Thus, the number of edges in the network remains the same after adding the duplicate nodes and each of them is connected randomly to some of the same nodes as their original. The goal of the data integration setups is thus to identify these duplicate nodes in the network and recombine them and their edges with the correct original nodes.

4.1.5 *Experiment setup* To test the sensitivity of the information transfer model to data integration, we define a variety of different (and realistic) setups. As per the method described in Section 3.1, integration setups are defined using indexing setups, comparison setups, classifying setups and clustering setups. We apply two different blocking setups, one blocking on given name and one blocking on surname, thus only performing pairwise comparisons between nodes that have the exact same given name in the first case and the exact same last name in the second case. We use two different comparison setups, both of which use a combination of exact and string matches on given names, surnames, states, etc. A Naive Bayes Classifier is used for classification and clustering is done with a walktrap approach [41]. This walktrap approach identifies communities in the match graph using a walktrap algorithm, and combines nodes that fall within the same community into one node. No edges are added or removed with this algorithm, only rewired from old nodes to the combined nodes they are added to. We use a random subsample of the true matches for a training set, taking 20% of the full set. In total, this results in $2^2 = 4$ different integration setups, one for each different combination of the indexing and comparison setups.

Following this, we apply the information transfer algorithm to test the effect the integration setups have had on the ability of the networks to transfer information. For each of the different integrated networks, we generate 10 000 information packets and set the maximum delivery time T equal to the size of the network, as done in Ref. [38]. We use a q-value of 0.2. The community of the target node is calculated using the walktrap community algorithm.

4.1.6 *Results* Table 2 shows the results of these simulations. We report the important statistics for the algorithm as noted by Ref. [38]: the proportion of packets that reach their target node and the average of inverse waiting times (a value of 1 means all packets were delivered in one step, while a value of 0 means no packets were delivered in time). The different integration setups result in networks ranging in size

TABLE 2 *Information transfer with different integration setups*

	T	Proportion packets delivered	Average inverse delivery time (standard deviation)
Setup 1	714	0.8516	0.0424 (0.0910)
Setup 2	771	0.7740	0.0345 (0.0801)
Setup 3	743	0.8050	0.0397 (0.0906)
Setup 4	744	0.8139	0.0392 (0.0925)

from 714 to 771 nodes, an approximate increase of 8%. In general, the smaller the network, the greater the proportion of packets that reach their target nodes within the maximum time. In the network with 714 nodes, approximately 85% of packets reach their target node. In the largest 771 node network, 77% of packets reach their target node. This is a decrease of 9.41%, even though the packets are given more time to reach their target node in the larger networks. The delivery times are also shorter in the smaller network. The change seen here is more significant, with a decrease of 18.63% in inverse delivery times when changing from the smallest network to the largest network.

These results would be crucial to those working to combat dark networks. One of the aims of law enforcement is to disrupt the ability of the network to spread information from member to member. When testing different network disruption approaches, researchers need to be conscious of the sensitivity of both the network itself and the information transfer to data integration. Removing 5% of nodes from a dark network through law enforcement might be a challenging task, but we see these sorts of changes in the size of the network through different data integration setups alone. Furthermore, a network disruption strategy that reduces the effectiveness of a dark network by 10–20% may be considered fairly effective. However, these results again show that simply changing one's data integration setup can result in changes of similar magnitude, with the different approaches tested above all being realistic and reasonable. As a result, researchers in dark networks should take account of this, and consider and test how changes to their data integration can influence their model output.

4.2 *Information spread*

4.2.1 *Goal* Similarly to information transfer diffusion models, information spread models are concerned with information moving through a complex network. As opposed to information transfer models, information spread models do not have specific source or target nodes, and instead consider general spread of information to nodes. As this spread happens over edges in the network, the topology of the network dictates which nodes information reaches or 'infects'. Hence, we wish to test how sensitive an information spread diffusion model is to different data integration setups.

4.2.2 *Domain* One of the domains in which information spread is studied is online social networks. As many people use online social networks as a means of communication, these networks can act as an avenue for the spread of information or ideas. How widely accepted certain information is can have an impact on society, and thus understanding how well information spreads through an online social network is important.

4.2.3 *Model* Kumar and Sinha [42] explain the use of diffusion models for modelling the spread of information through social networks. They suggest the effectiveness of using epidemic models to model information spread. This approach can answer questions such as 'how long does it take for the population

to become infected (for everyone to adopt the information)?' and 'what proportion of the population becomes infected (adopts the information)?'. In order to use such epidemic models in information spread modelling, the states require a different interpretation than in disease modelling. We apply a Susceptible-Infected-Recovered model [36]. In this model, nodes can only change from susceptible to infected or from infected to removed. In modelling information spread, we use the following state interpretations:

- Susceptible: Node has not yet adopted the information.
- Infected: Node has adopted the information, and is actively trying to spread it to its neighbours.
- Recovered: Node has adopted the information, but is no longer trying to spread it to its neighbours.

This is similar to Ref. [42], who use a Healthy-Susceptible-Infected model. We choose to use an SIR model as it is more widely studied and understood.

4.2.4 *Data* The data set was introduced by Zhang *et al.* [43]. They provide data for a number of different online social networks (Twitter, LiveJournal, Flickr, Last.fm and MySpace). These data include the usernames for the respective networks and the connections between users within each network. Additionally, they supply the matches of usernames across networks, i.e. the usernames from different networks that belong to the same real-world person.

We choose to focus on the Last.fm and MySpace networks. These networks are relatively smaller than the others and have a disproportionate number of people that use both. The MySpace network has a largest component with approximately 850 000 nodes and 6.5 million edges. The Last.fm network has a largest component with approximately 135 000 nodes and 1.7 million edges. There are a total of 956 nodes in the largest components of the two networks that represent the same entity.

We choose not to work with the full networks due to their still fairly large size, and instead subsample. We employ similar methodology to Peled *et al.* [44]: we chose pairs of users from the two networks that correspond to the same real-world entity. We then perform snowball sampling independently on the networks to create our two subsampled networks.

We randomly choose 200 seed nodes from each network, which represent the same real-world entity. For each seed node, we randomly choose five of its neighbours. For each of these neighbours, we randomly choose five of their neighbours. We then select the subgraph from the full network with all of these nodes. This gives a network with a magnitude around $200 \times 5^2 = 5000$. Doing this for both the MySpace and Last.fm networks results in two networks with at least 200 nodes in common and of similar sizes. We treat these as the two networks we wish to test integration setups and information diffusion on.

4.2.5 *Experiment setup* For the data integration setups, we use the usernames as comparison attributes between the two networks. We apply two different string similarity measurements (Levenshtein and Jarowinkler) and a number of different string similarity thresholds for each. We match two nodes if their string similarity exceeds the chosen threshold and perform walktrap clustering as described in the previous section. We do not use a sophisticated classification algorithm in this example, as we only have one feature for comparison.

In the SIR model, we set an infection rate of 0.02 and a recovery rate of 0.01. For each integrated network, we run 25 realizations of the diffusion model with 600 time steps each (long enough to reach stability). We repeat the entire subsampling, integration and diffusion process 25 times, which provides smooth results while balancing computation time.

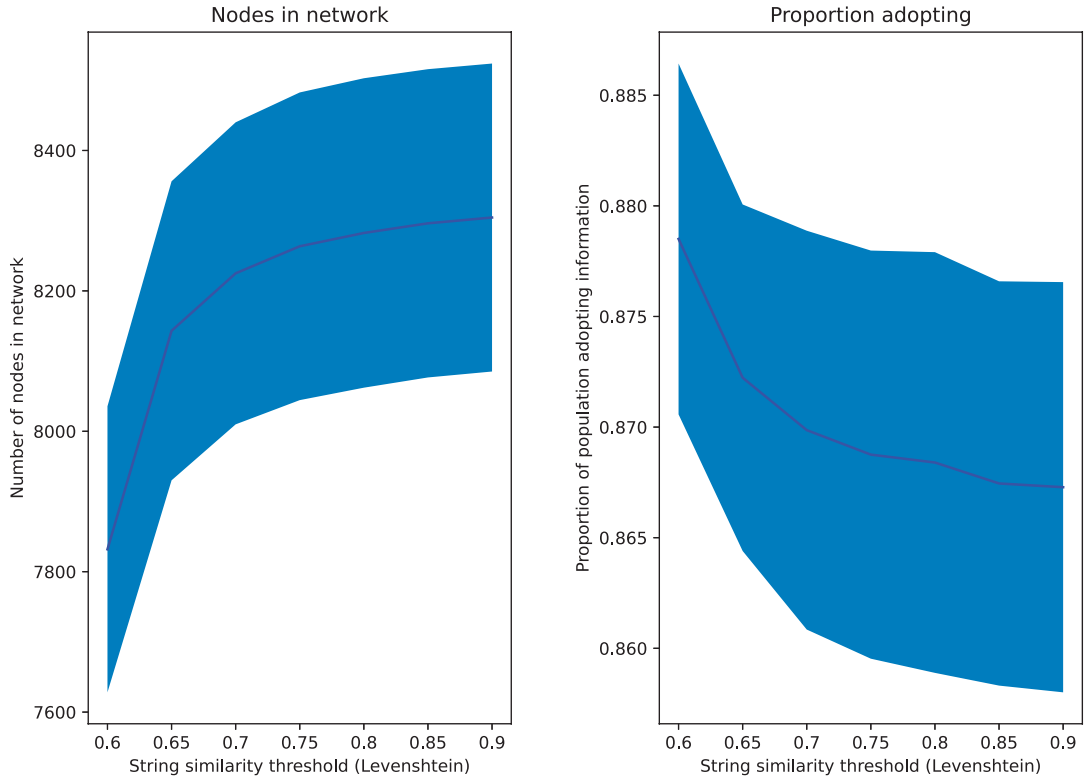

FIG. 2. Network size and information adoption in integrated network for varying string similarity threshold (Levenshtein). (Left) Number of nodes in the network after integrating nodes based on username Levenshtein string similarity, with 90% confidence intervals based on multiple snowball samples shown by filled region. (Right) Proportion of the population becoming infected in SIR simulations run on integrated networks, with 90% confidence intervals based on multiple snowball samples and model runs shown by filled region.

4.2.6 *Results* Figure 2 shows the results of the information diffusion when using a Levenshtein string similarity measure, with 90% confidence intervals based on the sample standard deviation. As the threshold is decreased, more nodes are predicted to represent the same entity and are thus combined. This decrease is non-linear, showing more rapid changes as the threshold is continually lowered. The average number of nodes in the network varies approximately between 8300 and 7800. There are also some changes in the proportion of the population adopting the information, but these are not large, with only a 2.5% difference at the extreme.

Figure 3 shows the results when using a Jarowinkler string similarity measure instead. Once again, reducing the threshold value reduces the number of nodes in the integrated network. However, this change is now roughly linear, and much greater. The total number of nodes in the integrated network can vary between roughly 8100 and 100, a significantly larger difference than observed in the Levenshtein example. Additionally, the proportion adopting information behaviour is now non-linear, with a clear threshold at which the diffusion model results drastically change. Maximum changes are now in the range of 20% total information adoption. We also note that the variance between simulations is significantly lower using a Jarowinkler threshold than a Levenshtein threshold.

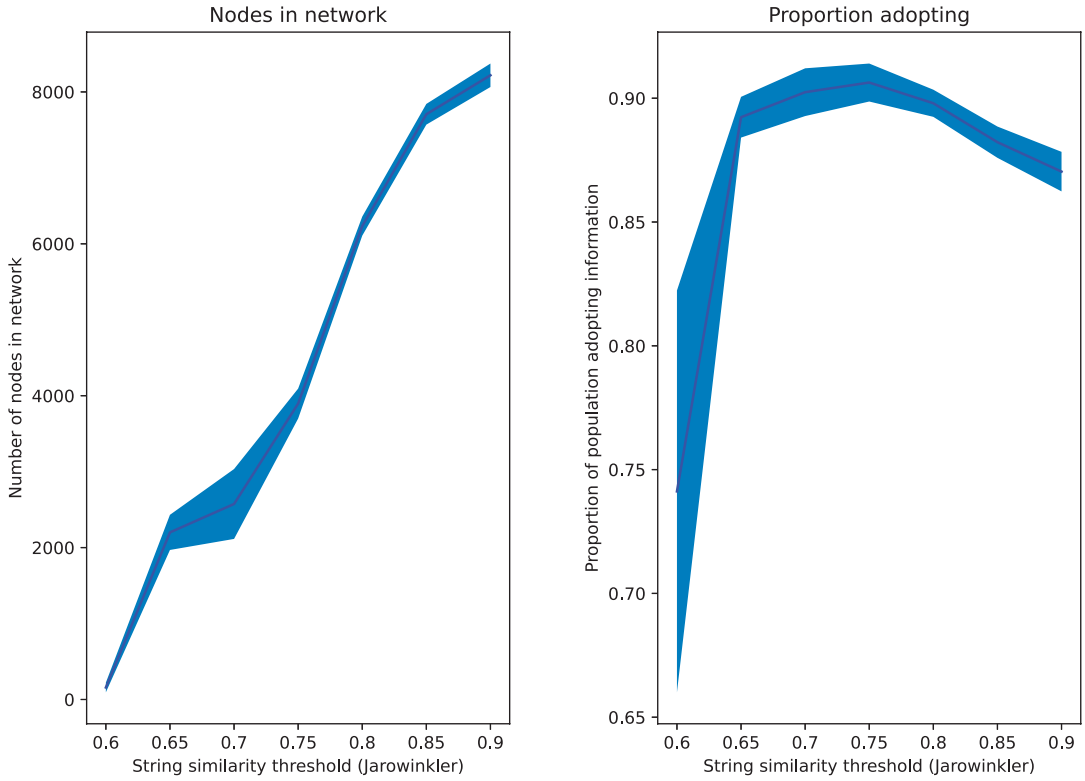

FIG. 3. Network size and information adoption in integrated network for varying string similarity threshold (Jarowinkler). (Left) Number of nodes in the network after integrating nodes based on username Jarowinkler string similarity, with 90% confidence intervals based on multiple snowball samples shown by filled region. (Right) Proportion of the population becoming infected in SIR simulations run on integrated networks, with 90% confidence intervals based on multiple snowball samples and model runs shown by filled region.

Both experiments highlight the effect that changing the string similarity threshold can have on the network. Reducing the threshold reduces the number of nodes in the integrated network, but the behaviour of this change is not always reflected in the diffusion model results. In the Levenshtein case, there is a non-linear change in the number of nodes in the integrated network, with an average reduction of roughly 6% of nodes as the threshold is decreased from 0.9 to 0.6. However, there is almost no appreciable change in the proportion of the population adopting the information. In the Jarowinkler case, changing the threshold changes the number of nodes linearly, with almost all of the nodes being integrated into one as the threshold is reduced. We see in this case much more significant changes in the proportion of the population adopting the information, where the diffusion model starts to show much lower spread at a critical string threshold where the network begins to collapse. The results of the Jarowinkler setup are also more consistent between simulations, as can be seen with the lower variance.

This diffusion model is thus not very sensitive to string threshold when calculating similarities using the Levenshtein measure, but is highly sensitive when using Jarowinkler similarity. Hence, the decision of which measure to use needs to be carefully considered. If a Jarowinkler measure is used, a prudent

approach would be to calculate model results for a number of different thresholds, and consider these results in aggregate, rather than for a single threshold value.

4.3 *Label propagation*

4.3.1 *Goal* Label propagation models attempt to accurately label all nodes in a network based on an initial set of prelabelled nodes. As in the previous two experiments, this labelling is done through the edges in the network. Through this experiment, we wish to highlight how different diffusion models can show different sensitivities to data integration. The data sets used are similar to the information spread data, but the two models show different sensitivities.

4.3.2 *Domain* Coauthorship networks capture how different scientific authors work together on papers. As authors can be experts in different domains, or work within different research communities, label propagation models can be used to describe this community structure.

4.3.3 *Model* The diffusion model we wish to test is community label propagation. We provide the community labels for nodes in one network only, and then propagate these labels through the integrated network after combining two networks. We use a harmonic function to propagate the labels, which sets a node's label based on the most common label of its neighbours.

4.3.4 *Data* The data used are from Ji and Jin [45]. In their paper, the authors constructed and analysed both a coauthorship and citation network for statisticians. One of the key findings they communicated was different community structures they were able to identify in the coauthorship network, where researchers from certain geographical regions or research areas tended to be more densely connected. We perform experiments on a subset of the coauthorship network. This network was built using paper research information, where authors are nodes and have an edge between them when they have coauthored at least two papers. We use only the giant component of this network, which is made up of 236 nodes and can all be grouped as 'High Dimensional Data Analysis' researchers. The data consist of the author names and to which community they belong within the giant component. Reference [45] identifies two communities: the 'North Carolina' community and the 'Carroll-Hall' community.

We create a partially synthetic data integration problem by dividing this network into two and then recombining them after adding realistic noise to the author names. We do so using the *Valentine Dataset Generator* [46]. This library allows users to add noise to string data based on keyboard proximity, emulating realistic data entry errors. We use an error rate of 10% as an input to the data generator. We create a network with duplicate nodes from the original network and randomly move some edges from the original network to the new network. Following this, we randomly relabel some nodes on the new network with the noisy author names.

4.3.5 *Experiment setup* We perform a total of 100 simulations. In each simulation, each edge is randomly assigned to either the original or duplicate network with equal probability. Each node in the duplicate network is renamed to the noisy author name with 50% probability. The two networks are then integrated using Levenshtein string similarity on the author names with various thresholds between 0.5 and 0.9. Following this, only the original nodes are given their correct community labels, while the new nodes are labelled using the harmonic function propagation.

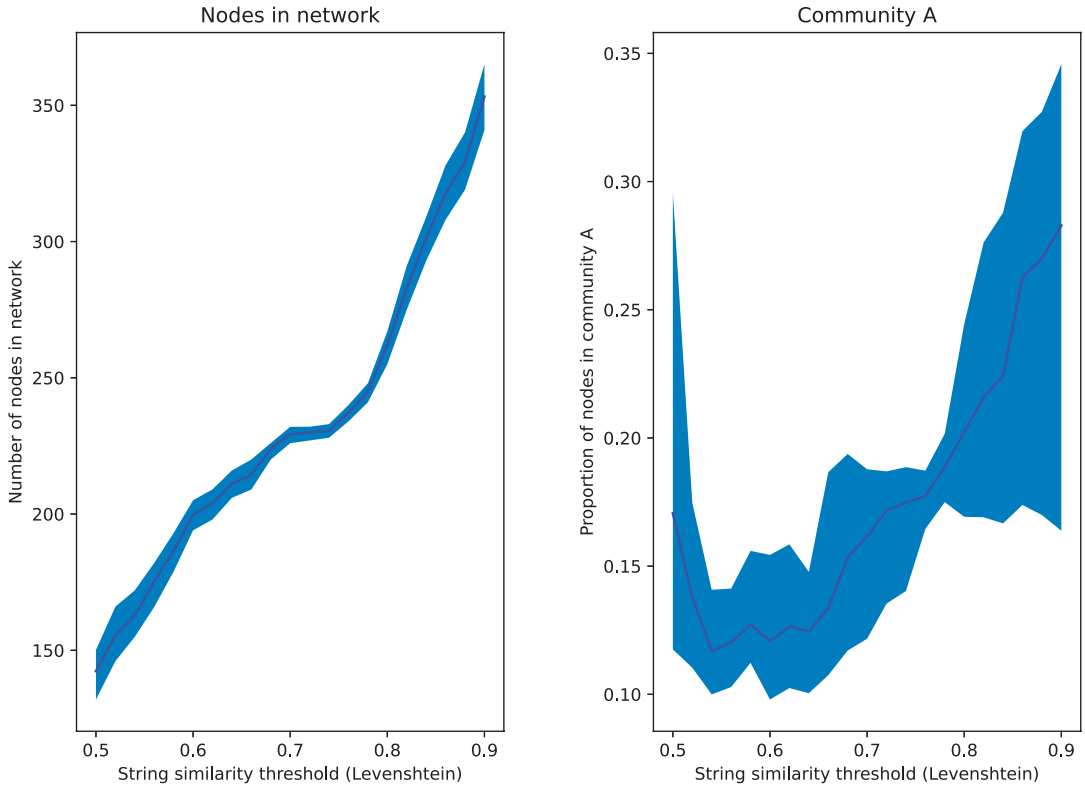

FIG. 4. Network and community size in integrated network for varying string similarity threshold (Levenshtein). (Left) Number of nodes in the network after integrating nodes based on name Levenshtein string similarity, with 90% confidence intervals based on multiple edge and noisy name assignment simulations shown by filled region. (Right) Proportion of population labelled as community A in integrated networks, with 90% confidence intervals based on multiple edge and noisy name assignment simulations shown by filled region.

4.3.6 *Results* Figure 4 shows the results of these simulations. The left figure shows the average number of nodes in the integrated network for varying string similarity thresholds, as well as the 95th and 5th percentile number of nodes. As mentioned earlier, there are two communities in this data set. The right figure shows the proportion of the nodes being labelled in community A, with the percentiles shown as for the nodes.

The number of nodes in the integrated network decreases fairly linearly as the string threshold is lowered. On the other hand, the label propagation is non-linear, with an initial decrease in the proportion of the population in community A as the threshold is decreased, followed by an increase when the threshold is lowered below approximately 0.55. These changes can be quite significant, varying between a mean high of 0.283 and a low of 0.117. This differs from what was observed in the Online Social Network experiment (Fig. 2), which had a similar entity resolution task using Levenshtein similarity on names. In that case, the change in the number of nodes was non-linear, while the changes in diffusion model results were not significant. This contrast highlights the importance of applying this method on a case-by-case basis, as different diffusion models and integration setups can behave differently with similar data sets and entity resolution tasks.

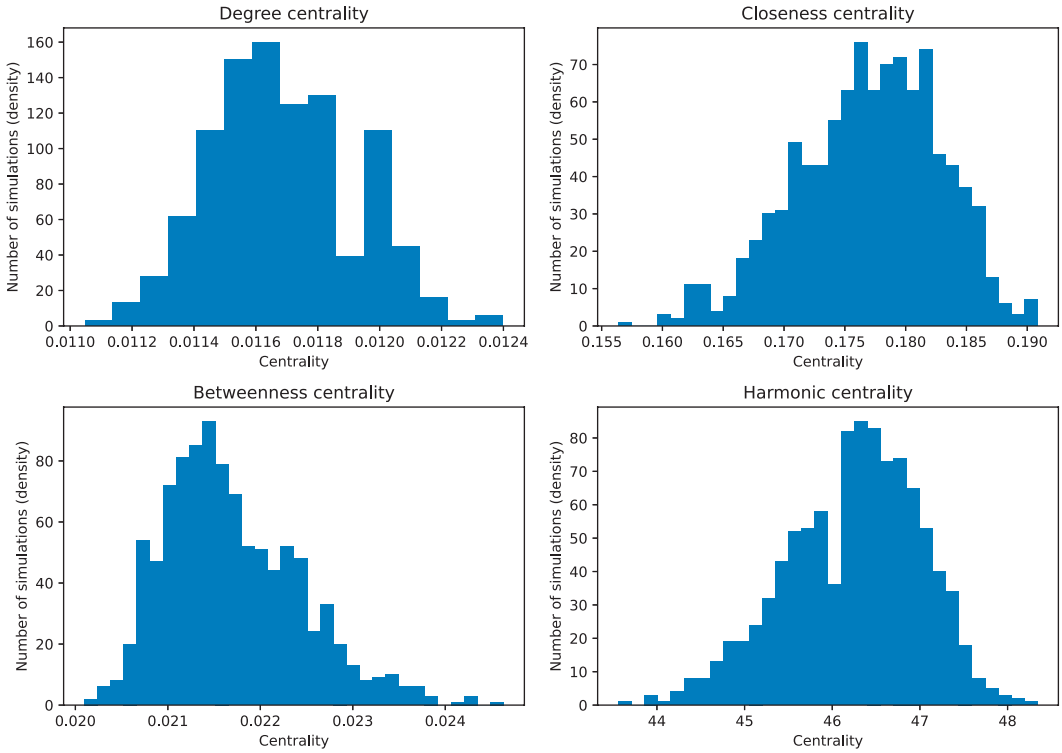

FIG. 5. Distributions of various network properties of integrated networks over different simulations of edge and noisy name assignment data. Nodes are integrated based on name similarity with a Levenshtein string threshold of 0.7. Clockwise from top left: degree centrality; closeness centrality; harmonic centrality; betweenness centrality.

Another noticeable point in Fig. 4 is the much higher variance between simulations in the label propagation results than in the network size results. This shows that while there may be low variance in the number of matches identified in the entity resolution of networks, the way that these networks are divided and integrated can have a significant effect on other network properties and the network model being run. How data are divided between different sources is thus also a concern, and not only what the data looks like in aggregate.

To help illustrate this concern, we perform an additional 1000 simulations of the data creation and integration process, this time testing only a Levenshtein string threshold of 0.7 and not running the diffusion model on the integrated networks. Figure 5 shows the distributions of average network properties over these simulations. The figure shows the fairly high variance that can be seen in average properties of the networks between simulations, which partially explains the variance observed in the label propagation results. This variance is due to the way that the edges are divided between the two networks, and the noise that is added to the second network. For these same simulations, the mean number of nodes in the integrated networks is 225.43 with a standard deviation of 2.31, which is relatively lower variance than seen in the network properties. Thus, while how the data are divided between different sources may not drastically affect network size, it can vary other network properties and downstream diffusion models.

4.4 *Comparative analysis*

In all three experiments, we see significantly different diffusion model results on the different integrated networks. In the information transfer experiment, we were able to identify sensitivity of the model and graph size to changes in the integration setup. These changes could be of a sizeable magnitude, comparable to what might be seen with different network disruption approaches applied in the dark networks domain. In the information spread experiment, we noted a linear sensitivity of the size of the integrated network to the string similarity threshold in the Jarowinkler case and a non-linear change in the Levenshtein case. The behaviour of the network was not always equivalent to what was observed in the diffusion model, with either no real changes or a non-linear, critical threshold change being observed. The results of a similar approach in the label propagation experiment were rather different, with differences in both linearity and variance. The contrast between the results of the information spread and label propagation models shows that different diffusion models can exhibit different sensitivities to data integration decisions even with similar data sets.

All of the experimental results illustrate the magnitude of effect that data integration decisions can have on diffusion models, and how these effects can be shown using our method. In cases where there is high sensitivity, one should be extra careful with the choice of data integration approach. As there is seldom a ground truth, capturing results from a number of different data integration setups can give a more full picture of the potential behaviour of the diffusion model in question.

5. Conclusions

As more and more data sources become available for building and analysing complex networks using network diffusion models, the need for a systematic way to test data handling decisions is growing.

Hence, we introduced a method for testing realistic data integration approaches to complex network data and their effects on network diffusion models, as well as an associated software package that implements this method. We find that:

- based on our literature review, data integration for complex networks is growing increasingly important as the number of possible data sources grows, particularly in domains such as online social networks and in dark networks where data are almost always incomplete from a single source;
- our method is effective in identifying sensitivities of diffusion models to data integration decisions, and highlighting decisions for which particular caution needs to be exercised. The analytic results show the change in diffusion model results as data integration changes.

These findings further emphasize the importance of this research area and the possibilities for further research. The method can be applied to even more complex and realistic data sets, where results for different data integration decisions can be contrasted with the results obtained with a single data integration approach.

While the domains and types of diffusion processes chosen cover a wide range of diffusion cases, these results are not necessarily generalizable to all diffusion processes. The differences observed in sensitivities between different models highlight the need for more extensive testing. Furthermore, all of the data sets used are semi-synthetic. It is challenging to source open data to test these sorts of methods, but this could prove important as the field develops.

There are many avenues for extending the method. When identifying matches using entity resolution, we currently only use attributes of the nodes. In some cases, properties of the network(s) might

also be useful. For example, the proportion of neighbours shared by two nodes. This can be indirectly implemented through the clustering algorithm, but there is the possibility of extending the comparing and classifying approaches to include this. Integration of complex networks also involves link prediction, as there may be edges that exist in reality that are not present in any of the source graphs. We again indirectly allow for this through custom clustering algorithms, but having some state-of-the-art algorithms included might prove useful. Finally, other aspects of data handling like data cleaning and more complex data fusion can be added.

References

1. Kitts, J. A. & Quintane, E. (2020) Rethinking social networks in the era of computational social science. *The Oxford Handbook of Social Networks*. (R. Light & J. Moody eds) Oxford, UK: Oxford University Press, pp. 71–97.
2. Robins, G., Bright, D., Weissinger, L. & Stys, P. (2022) Data collection for social network research. *Social Networks*, **69**, 1–2. https://doi.org/10.1016/j.socnet.2021.08.010.
3. Rice, E., Holloway, I. W., Barman-Adhikari, A., Fuentes, D., Brown, C. H. & Palinkas, L. A. (2014) A mixed methods approach to network data collection. *Field Methods*, **26**, 252–268.
4. Espín-Noboa, L., Karimi, F., Ribeiro, B., Lerman, K. & Wagner, C. (2021) Explaining classification performance and bias via network structure and sampling technique. *Appl. Netw. Sci.*, **6**, 1–25.
5. Zhang, X., Zhang, Z.-K., Wang, W., Hou, D., Xu, J., Ye, X. & Li, S. (2021) Multiplex network reconstruction for the coupled spatial diffusion of infodemic and pandemic of COVID-19. *Int. J. Digit. Earth*, **14**, 401–423.
6. Sadovykh, V., Sundaram, D. & Piramuthu, S. (2015) Do online social networks support decision-making?. *Decis. Support Syst.*, **70**, 15–30.
7. Smith, K. P. & Christakis, N. A. (2008) Social networks and health. *Annu. Rev. Sociol.*, **34**, 405–429.
8. Nevin, J., Lees, M. & Groth, P. (2021) The non-linear impact of data handling on network diffusion models. *Patterns*, **2**, 100397.
9. Smith, J. A. & Moody, J. (2013) Structural effects of network sampling coverage I: Nodes missing at random. *Soc. Netw.*, **35**, 652–668.
10. Smith, J. A., Moody, J. & Morgan, J. H. (2017) Network sampling coverage II: The effect of non-random missing data on network measurement. *Soc. Netw.*, **48**, 78–99.
11. Wang, D. J., Shi, X., McFarland, D. A. & Leskovec, J. (2012) Measurement error in network data: A re-classification. *Soc. Netw.*, **34**, 396–409.
12. Dong, X. L. & Srivastava, D. (2015) Big data integration. *Synth. Lect. Data Manag.*, **7**, 1–198.
13. Shu, K., Wang, S., Tang, J., Zafarani, R. & Liu, H. (2017) User identity linkage across online social networks: A review. *ACM SIGKDD Explor. Newsl.*, **18**, 5–17.
14. Roberts, N. C. (2011) Tracking and disrupting dark networks: Challenges of data collection and analysis. *Inform. Syst. Front.*, **13**, 5–19.
15. Mislove, A., Marcon, M., Gummadi, K. P., Druschel, P. & Bhattacharjee, B. (2007) Measurement and analysis of online social networks. *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pp. 29–42.
16. Noor, F., Shah, A. & Khan, S. A. (2015) Relation mining using cross correlation of multi domain social networks. *2015 SAI Intelligent Systems Conference (IntelliSys)*. London, UK: IEEE, pp. 898–903.
17. Bakker, R. M., Raab, J. & Milward, H. B. (2012) A preliminary theory of dark network resilience. *J. Policy Anal. Manag.*, **31**, 33–62.
18. Shaikh, M. A. & Jiaxin, W. (2008) Network structure mining: Locating and isolating core members in covert terrorist networks. *WSEAS Trans. Inform. Sci. Appl.*, **5**, 1011–1020.
19. Gera, R., Miller, R., MirandaLopez, M., Saxena, A. & Warnke, S. (2017) Three is the answer: Combining relationships to analyze multilayered terrorist networks. *2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Sydney, Australia: IEEE. pp. 868–875.

20. Campana, P. & Varese, F. (2020) Studying organized crime networks: Data sources, boundaries and the limits of structural measures. *Soc. Netw.*, **69**, 149–159.

21. Morris, J. F. & Deckro, R. F. (2013) SNA data difficulties with dark networks. *Behav. Sci. Terror. Polit. Aggress.*, **5**, 70–93.

22. Xu, J., Hu, D. & Chen, H. (2009) The dynamics of terrorist networks: Understanding the survival mechanisms of Global Salafi Jihad. *J. Homel. Secur. Emerg. Manag.*, **6**. Article 27.

23. Cunningham, D., Everton, S., Wilson, G., Padilla, C. & Zimmerman, D. (2013) Brokers and key players in the internationalization of the FARC. *Stud. Confl. Terror.*, **36**, 477–502.

24. Magouirk, J., Atran, S. & Sageman, M. (2008) Connecting terrorist networks. *Stud. Confl. & Terror.*, **31**, 1–16.

25. Everton, S. F. (2012) *Disrupting Dark Networks*, vol. 34. New York, USA: Cambridge University Press.

26. Duijn, P. A., Kashirin, V. & Sloot, P. (2014) The relative ineffectiveness of criminal network disruption. *Sci. Rep.*, **4**, 1–15.

27. Gregg, F. & Eder, D. (2022) Dedupe. https://github.com/dedupeio/dedupe.

28. Konda, P. V. (2018) *Magellan: Toward Building Entity Matching Management Systems*. The University of Wisconsin–Madison.

29. Papadakis, G., Tsekouras, L., Thanos, E., Giannakopoulos, G., Palpanas, T. & Koubarakis, M. (2018) The return of Jedai: End-to-end entity resolution for structured and semi-structured data. *Proc. VLDB Endow.,* **11**, 1950–1953.

30. De Bruin, J. (2019) *Python Record Linkage Toolkit: A Toolkit for Record Linkage and Duplicate Detection in Python*. Ver: 0.14. Zenodo. DOI: 10.5281/zenodo.3559043. https://doi.org/10.5281/zenodo.3559043.

31. Dobson, S. (2020) *Epidemic Modelling: Some Notes, Maths, and Code*. Independent Publishing Network.

32. Miller, J. C. & Ting, T. (2020) Eon (epidemics on networks): A fast, flexible python package for simulation, analytic approximation, and analysis of epidemics on networks. *arXiv preprint arXiv:2001.02436*.

33. van den Ende, M., Mayer, M., Epskamp, S., Lees, M. & van der Maas, H. (2021) Dynamics in and dynamics of networks using DyNSimF. https://doi.org/10.31234/osf.io/6kcfz.

34. Rossetti, G., Milli, L., Rinzivillo, S., Sirbu, A., Pedreschi, D. & Giannotti, F. (2017) Ndlib: Studying network diffusion dynamics. *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. Tokyo, Japan: IEEE, pp. 155–164.

35. Rossetti, G., Milli, L., Rinzivillo, S., Sîrbu, A., Pedreschi, D. & Giannotti, F. (2018) NDlib: a python library to model and analyze diffusion processes over complex networks. *Int. J. Data Sci. Anal.*, **5**, 61–79.

36. Kermack, W. O. & McKendrick, A. G. (1927) A contribution to the mathematical theory of epidemics. *Proc. R. Soc. Ser. A*, **115**, 700–721.

37. Granovetter, M. (1978) Threshold models of collective behavior. *Amer. J. Sociol.*, **83**, 1420–1443.

38. Czaplicka, A., Holyst, J. A. & Sloot, P. (2013) Noise enhances information transfer in hierarchical networks. *Sci. Rep.*, **3**, 1–8.

39. Christen, P. (2008) Febrl—an open source data cleaning, deduplication and record linkage system with a graphical user interface. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1065–1068.

40. Barabási, A.-L. & Albert, R. (1999) Emergence of scaling in random networks. *Science*, **286**, 509–512.

41. Pons, P. & Latapy, M. (2006) Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, **20**, 284–293.

42. Kumar, P. & Sinha, A. (2021) Information diffusion modeling and analysis for socially interacting networks. *Soc. Netw. Anal. Min.*, **11**, 1–18.

43. Zhang, Y., Tang, J., Yang, Z., Pei, J. & Yu, P. S. (2015) Cosnet: Connecting heterogeneous social networks with local and global consistency. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1485–1494.

44. Peled, O., Fire, M., Rokach, L. & Elovici, Y. (2013) Entity matching in online social networks. *2013 International Conference on Social Computing*. Alexandria, USA: IEEE, pp. 339–344.

45. Ji, P. & Jin, J. (2016) Coauthorship and citation networks for statisticians. *Ann. Appl. Stat.*, **10**, 1779–1812.
46. Koutras, C., Siachamis, G., Ionescu, A., Psarakis, K., Brons, J., Fragkoulis, M., Lofi, C., Bonifati, A. & Katsifodimos, A. (2021) Valentine: Evaluating matching techniques for dataset discovery. *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, Chania, Greece: IEEE. pp. 468–479.

Appendix A. Data integration background

Different data sources may represent the same real-world entity (nodes in a network) in different ways. They may record different attributes for each entity. For example, one might record dates of birth and another ID numbers. Even when the same attributes are recorded by different data sources, they may do so in a different way, like recording a person's name using their initials and surname versus using their full name. Additionally, there may be differences in the actual values recorded, possibly due to measurement or entry error. Data integration attempts to resolve these issues and identify the same real-world entities across and within data sets and combine them into one.

There are three key steps in data integration [12]: *schema alignment*, *record linkage* and *data fusion*. For the remainder of this article, we will generally use the terms 'data integration' and 'deduplication' interchangeably. We primarily focus on the record linkage and data fusion steps of data integration, which involve identifying which records refer to the same entity and the process of combining them into one, respectively.

The first step in record linkage is choosing a technique for pairwise matching. That is, a technique that compares two records and decides whether they refer to the same entity. This technique could be rule-based (such as some ID attribute needing to be the same), classification-based (training a classification algorithm using examples) or distance-based. This process can be divided into two components.

Comparison involves deciding on which attributes/features pairs of records will be compared. Attributes can be compared using string similarity, numerical measurements and distance measurements.

Once record attribute comparison values have been calculated, matches and non-matches need to be identified based on these comparison features. This is known as *classification*; there are both supervised (requiring training data) and unsupervised (not requiring training data) algorithms and approaches to this.

In the second step of record linkage, the defined pairwise matching technique can then be applied to the data sources. Two issues need to be addressed: computational limitations and inconsistency of matching outcome. When dealing with large amounts of data, it becomes computationally infeasible to perform the pairwise matching between all possible pairs of records. This is especially the case when optimizing for inconsistencies, where these comparisons are repeated numerous times. *Indexing* is the process of determining which records within the data should be compared in testing potential duplicate entities. With a full index, every possible pair of entries is compared.

A less computationally demanding indexing approach is to 'block' records, where rules are used to partition records into different blocks. The pairwise matching is performed within blocks, and never between blocks. This means that records assigned to different blocks are never compared, and thus are assumed never to refer to the same entity. For example, if comparing users in different social networks, one might block according to location, since two people living in two different places could never be the same, no matter how similar other information on them is. There are further indexing algorithms like sorted neighbourhood indexing (essentially a less strict blocking, allowing for small spelling mistakes) and random indexing, where random pairs of records are compared. Which approach is best is dependent on the accuracy of the data used for the indexing rules, hence one may test multiple blocking/indexing rules, and choose the best overall result.

Finally, *clustering* is used to address inconsistent pairwise match/non-match conclusions and decide how to fuse duplicate nodes on the network. Here, issues like transitivity ($A = B, B = C, A \neq C$) and edge prediction need to be addressed. In order to deal with the inconsistencies, one can create a penalty function to be applied within and between 'clusters'—groups of records that have been identified as referring to the same entity. Records within the same cluster that do not refer to the same entity according to the pairwise matching are penalized and records in different clusters that do refer to the same entity according to the pairwise matching are penalized. One then searches for the clustering that minimizes the clustering penalty function.

Appendix B. Experiment code

Figure A1 shows the code for testing the different integration setups in Section 4.1. We see how there are two different blocking and comparison setups, and one classifier and clustering setup. We could also define more of each, if desired. These are combined in all possible combinations and executed to calculate a list of integrated networks.

Figure A2 shows the code for running an SIR model with parameters as in Section 4.2 on a set of different integrated networks using *NIDMod*. We define the statuses, compartments, transition rules and model parameters as for a custom diffusion model in NDlib,[4] followed by executing the runs with the set of simulation parameters described above according again to the NDlib utility.[5]

[4] https://ndlib.readthedocs.io/en/latest/custom/custom.html
[5] https://ndlib.readthedocs.io/en/latest/reference/utils/multiple_run.html

```
# different indexing setups
index_setup_0 = {'Block': [['given_name', 'given_name']]}
index_setup_1 = {'Block': [['surname', 'surname']]}
all_index_setups = [index_setup_0, index_setup_1]

# different comparison setups
compare_setup_0 = {'Exact': [['given_name', 'given_name'],
                      ['date_of_birth', 'date_of_birth'],
                      ['suburb', 'suburb'], ['state', 'state']],
           'String': [['surname', 'surname', 'jarowinkler', 0.85],
                   ['address_1', 'address_1', 'levenshtein',
                    0.85]]}
compare_setup_1 = {'Exact': [['surname', 'surname'],
                       ['suburb', 'suburb'],
                       ['state', 'state']],
      'String': [['given_name', 'given_name', 'jarowinkler', 0.85],
                 ['address_1', 'address_1', 'levenshtein', 0.85]]}

all_compare_setups = [compare_setup_0, compare_setup_1]

# different classifier setups
classifier_name_0 = 'NaiveBayesClassifier'
all_classifier_names = [classifier_name_0]

# clustering algorithms
clustering_alg_0 = 'walktrap_integration'
all_clustering_algs = [clustering_alg_0]

# building all combinations
from nidmod.parameter_sweeper import CombinationBuilder
combination_builder = CombinationBuilder(all_index_setups,
   all_compare_setups, all_classifier_names, all_clustering_algs)
integration_setups = combination_builder.get_all_combinations()

# integrating according to different integration setups
from nidmod.parameter_sweeper import ParameterSweeper

# the 'dark network', including node attributes
graphs = [febrl_graph_attr]
parameter_sweeper = ParameterSweeper(integration_setups, graphs,
   training_matches)
different_integrated_networks = parameter_sweeper.
   get_integrated_networks()
```

FIG. A1. Code for implementing a variety of different integration setups to create a list of integrated networks.

```
from nidmod.diffusionmodel.diffusionmodel import CustomDiffusionModel
from nidmod.parameter_sweeper import MultiNetworkDiffusion

statuses = ['Susceptible', 'Infected', 'Removed']
compartments = {'NodeStochastic': {'c1': [0.02, 'Infected'], 'c2':
    [0.01]}}
transition_rules = [["Susceptible", "Infected", "c1"],
                ["Infected", "Removed", "c2"]]
model_parameters = [['fraction_infected', 0.1]]

custom_diffusion_model = CustomDiffusionModel(statuses, compartments,
     transition_rules, model_parameters)

multi_network_diffusion = MultiNetworkDiffusion(
    different_integrated_networks, custom_diffusion_model)

simulation_parameters = [25, 600, None, 5]
graph_assc_results_analysers = multi_network_diffusion.
    run_diffusion_model(simulation_parameters)
```

FIG. A2. Code for running an SIR model on multiple integrated networks.