DEXA 2022, LNCS 13426
This is a self-archived pre-print version of this article.
The final publication is available at Springer via
https://doi.org/10.1007/978-3-031-12423-5_10.

Anonymisation of Heterogeneous Graphs with Multiple Edge Types *

Guillermo Alamán Requena[1], Rudolf Mayer[1] ✉, and Andreas Ekelhart[1]

SBA Research, Vienna, Austria {rmayer,aekelhart}@sba-research.org

Abstract. Anonymisation is a strategy often employed when sharing and exchanging data that contains personal and sensitive information, to avoid possible record identification or inference. Besides the actual attributes contained within a dataset, also certain other aspects might reveal information on the data subjects. One example of this is the structure within a graph, i.e. the connection between nodes. These might allow to re-identify a specific person, e.g. by knowledge of the number of connections for some individuals within the dataset.
Thus, anonymisation of the structure is an important aspect of achieving privacy. In this paper, we therefore present an algorithm that extends upon the current state of the art by considering multiple types of connections (relations) between nodes.

Keywords: Graph Structure Anonymisation · Multiple Relational Types

1 Introduction

The amount of data collected is ever increasing, and data represented as graphs, e.g. social networks or processes e.g. in the knowledge work domain [5], are no exception. Several interesting data analysis tasks utilise such graphs, which represent connections between individuals, organisations, and other entities. As this data is highly personal, data protection becomes an important aspect. Historically, tabular data was among the first types to be addressed, with methods such as k-anonymity or differential privacy being developed.

In graphs, besides the values within nodes, which could be treated in a similar manner as tabular data, also the structural information encoded in the connections (edges) is of concern. Depending on the background knowledge of an attacker, it might be possible to re-identify individuals based on this structure alone, e.g. especially those individuals that have unusual patterns of connections [6]. Thus, recent years have also shown an increase in works addressing structural anonymisation, such as adaptations of the concept of k-anonymity [2,7], and combinations of node and structure anonymisation, in various types of graphs [9,1,8].

* SBA Research (SBA-K1) is a COMET Centre within the framework of COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMDW, and the federal state of Vienna; COMET is managed by FFG.
This work is supported by FFG under Grant No. 871299 (project KnoP-2D).

In this paper, we specifically expand on previous approaches for achieving structural anonymity for graphs with multiple types of edge connections. While most existing works consider homogeneous graphs, i.e. with only one type (e.g. *foaf:knows*), in a heterogeneous graph, nodes can be linked by varying types of connections. Heterogeneity complicates structure anonymisation, as attacks may take advantage of this addition information. Based on the ideas from [3], our goal is to develop a method to anonymise heterogeneous Resource Description Framework (RDF)[1] graphs. The idea developed in [3] is that the one-hop neighbourhood of any resource to be anonymised should be indistinguishable from the one-hop neighbourhood of at least k-1 other resources. For that purpose, they developed a greedy heterogeneous graph modification algorithm for a simplified RDF graph which includes only 4 types of semantic connections. However, although the general guidelines of the algorithm are stated in pseudo code, no implementation is publicly available. Based on their approach, our contributions in this work consist of (i) an extension to the approach of [3] to increase flexibility and usability of the anonymisation method, and (ii) an open-source implementation in Python[2].

2 k-RDF-Neighbourhood Anonymisation with Multiple Edge Types

In this section, we describe our method and extensions of the anonymisation method [3]. Since the focus of our method is to anonymise heterogeneous RDF graphs, we first present the formal definition of a heterogeneous graph by [4]:

Definition 1. *A heterogeneous graph is defined as a directed graph* $G = (V, E, A, \Delta)$ *where each node* $\nu \in V$ *and each edge* $\epsilon \in E$ *are associated with their type mapping functions* $\theta(\nu) : V \rightarrow A$ *and* $\omega(\epsilon) : E \rightarrow \Delta$*, respectively.*

In the following, we describe our method on an example heterogeneous RDF graph utilising FOAF[3], with vertices of type *foaf:Person*, representing people, edges of type *foaf:knows*, representing relations between individuals, and edges of type *foaf:CurrentProject*, indicating projects an individual is working on. Other edges primarily serve to describe properties, such as *foaf:Age*, or *foaf:Name*, while *custom:has_disease* is an example of a custom property outside the FOAF specification. Following the procedure of [10] and [3], we will demonstrate the anonymisation on the one-hop neighbourhood of *foaf:Person* resources [4].

Our method initially gets a list of connection types to consider for structure anonymisation, all other attributes will be removed[5]. Edge connections in the

[1] https://www.w3.org/RDF/
[2] https://github.com/sbaresearch/graph-anonymisation
[3] http://www.foaf-project.org/
[4] Note that apart from reducing computational complexity, it is logical to target individuals, since it is the most common setting when facing an anonymisation task
[5] Node value anonymisation, if necessary, is a pre-requisite step and not covered by our structure anonymisation method

one-hop-neighbourhood of any node ν of a target graph can be classified into three different categories, depending on the type of information they describe:

- **Attribute connections:** edges connecting a node (e.g. *foaf:Person*) to a descriptive characteristic of this individual which is stored as a Literal.
- **Unidirectional connections:** directed edges connecting a node to other entities (e.g. *foaf:Person* to a project via *foaf:CurrentProject*).
- **Bidirectional connections:** edges symmetrically connecting nodes (e.g. *foaf:Person* via *foaf:knows*).

Definition 2. *A **heterogeneous RDF graph is said to be k-anonymous** if there are at least k identical one-hop-neighbourhoods in the target graph for each node $\nu \in N$. We consider that two attributes of the one-hop-neighbourhood of a pair of nodes x and y, are identical if they are generalised to the same level. We consider two unidirectional connections of the one-hop-neighbourhood of a pair of nodes x and y to be identical if they point exactly to the same resources. We consider the bidirectional connections of the one-hop neighbourhood of a pair of nodes x and y to be identical if their one-hop-neighbourhoods are isomorphic.*

In order to fulfil the anonymisation criteria defined above, we rely on three different algorithms (similar to [3]):

The Neighbourhood Code Extraction Algorithm compares one-hop-neighbourhoods of target nodes (e.g. *foaf:Person*) across the target graph. For this purpose, we encode the node neighbourhood information into a more efficient data structure than the raw RDF graph. We chose a hashtable due to its low indexing complexity ($O(n)$). Depending on the type of edge connection, the information contained in the one-hop-neighbourhood of a node ν is stored in a different way:

- **Attribute connections**: the attributes of each individual are stored as *key value* pairs (i.e., *foaf:Age* "40").
- **Unidirectional connections**: the resources to which each unidirectional connection of an individual points to are stored in a list. The type of connection is the *key* (i.e., *foaf:CurrentProject*) and the list of resources is the *value* associated with it (i.e., ["Project1", "Project3", "Project7"]).
- **Bidirectional connections**: Multiple isomorphic tests have to be conducted for each bidirectional connection. At this time, no polynomial time algorithm for the general isomorphic problem [10] is known. In our approach, we utilise the same string representation of the edges as proposed in [3] and based on [10]. The main idea is to encode the information of each sub-graph G_{bidi_i} created by considering only one type of bidirectional connection across the one-hop-neighbourhood of a node ν, so that the one-hop-neighbourhood of two *foaf:Person* nodes can be considered isomorphic in terms of that type of bidirectional connections if the generated codes are identical in structure. The way this encoding is constructed consists of finding the *minimum* depth-first search (DFS) tree of each component and concatenating it in a list where all these minimum trees are stored. We simplify the search of the minimum

DFS tree by dynamically discarding candidate paths. In the worst case scenario in which all the DFS trees in the subgraph fulfil the criteria, one of the paths is taken randomly and the encoding algorithm becomes $O(n!)$ which is the same complexity as the original algorithm proposed by [3]. Following the guidelines of [3], we call the dictionary encoding the one-hop-neighbourhood of a node ν the Full Neighbourhood Code of ν ($FNHC_v$).

Dissimilarity Computation Algorithm To compute the dissimilarity between each of the nodes, we use the information stored in the Full Neighbourhood Code hashtable. The dissimilarity between the one-hop-neighbourhood of two nodes x and y is the weighted sum of the dissimilarity of each connection in that neighbourhood:

$$sim(FNHC_x, FNHC_y) = \sum_{i=0}^{N} \alpha_i * sim_i(FNHC_{x_i}, FNHC_{y_i}) \tag{1}$$

where N is the set of connection types present in the one-hop-neighbourhood, α_i is the weight of the dissimilarity of attribute i (sim_i) to the total dissimilarity between the nodes x and y. There are three types of dissimilarity functions, one for each type of edge connection described above:

- **Dissimilarity of attribute connections** is the normalised distance of two attributes x_i and y_i given a defined hierarchy tree. It ranges between 0 (identical) and 1 (reached highest level of hierarchy).
- **Dissimilarity of unidirectional connections** between two nodes x and y, given a set of Literals to which they point, is defined as the number of connections of that type to be deleted so that two nodes x and y are connected to exactly the same Literals or resources.
- **Dissimilarity of bidirectional connections** between a node x and another node y, given the one-hop-neighbourhood, is determined by the amount of edges one needs to delete so that one-hop-neighbourhoods of both nodes are identical (isomorphic).

To compute the complete dissimilarity between two nodes x and y, one needs to calculate the similarity of each of the connections using the corresponding methods explained above and apply the weighted sum provided in Equation (1).

The Graph Modification Algorithm is also based on the ideas presented by [3] with some modifications to improve scalability. The main goal of this algorithm is to transform the one-hop-neighbourhood of a group of k given nodes, so that the anonymisation criteria is fulfilled for all of them. We refer to this group of nodes as *anonymised neighbourhoods* or *equivalent classes*.

- For the **generalisation of attribute connections**, the attributes of each of the k nodes are generalised to the lowest level's possible common value in the hierarchy tree provided.

- When **generalising unidirectional connections**, one should remove the necessary edges, so that each of the k nodes are connected exactly to the same Literals and resources via those unidirectional connections. We follow the idea of only deleting edges to avoid introducing false information (added edges) in the graph. In addition, as pointed out by [3], the approach of deleting edges fits with the open world assumption which suggests that missing statements can also be true.
- The **generalisation of bidirectional connections** is the most complex one. As for unidirectional connections, it relies on the same type of calculations used when computing dissimilarity. That means, for each node in the neighbourhood of size k, one should delete all the necessary edges so that the one-hop-neighbourhood of each of them is isomorphic in terms of each of the bidirectional connections. With our method, it is enough to take one of the nodes as reference and perform a pairwise comparisons to every other node twice (double-pass). At every comparison, the one-hop-neighbourhood of the reference node and the other node under comparison are updated via edge deletion so that they are isomorphic. The idea is that after the first pass, the reference one-hop-neighbourhood takes the minimum isomorphic representation and in the second round, this structure is acquired by all the other nodes.

We would like to point out two of the major challenges that arise when anonymising bidirectional connections. First, when deleting edges during the described double pass, the edges of other one-hop-neighbourhoods may be affected as well, and this can lead to more edge deletions than necessary and hence, additional information loss. To avoid this issue, we only store which edges to delete during the double pass, but they are only deleted when the algorithm has finished. Edge deletion may still cause some additional edges to be deleted in the neighbourhood, and therefore, they might not be isomorphic anymore. However, since the calculation of which edges to delete ensures that they are actually isomorphic in the first place, deleting additional edges of the structure of each of the one-hop-neighbourhoods does not reveal any additional information, and we can still consider them isomorphic in terms of the anonymisation goal.

Secondly, deleting edges may affect the one-hop-neighbourhood of other nodes that are not in the same neighbourhood as the k target nodes: (i) The one-hop-neighbourhood of non-anonymised nodes is affected – then, one needs to simply update the one-hop-neighbourhood or (ii) this affects the one-hop-neighbourhood of anonymised nodes, then this is the exact same situation as in 1).

Due to these improvements, our method is able to deal with larger graphs than the earlier approach by [3]. Furthermore, the consideration of the outlined challenges leads to reduced information loss.

3 Conclusions

Anonymisation of graph data differs from relational data as also the structure of graphs can be utilize by an attacker to perform e.g. a re-identification attack.

In this paper, we have thus presented an algorithm for anonymising the structure of graphs. We extended previous work by allowing on the one hand heterogeneous graph structures with multiple types of edges, and on the other hand also scaled up the algorithm.

Future work will focus on evaluating our approach in diverse settings against benchmark datasets, and measure the effect of the anonymisation on utility.

References

1. Campan, A., Truta, T.M.: Data and Structural k-Anonymity in Social Networks. In: Privacy, Security, and Trust in KDD. vol. 5456, pp. 33–54. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01718-6_4
2. Feder, T., Nabar, S.U., Terzi, E.: Anonymizing Graphs (Oct 2008), arXiv:0810.5578
3. Heitmann, B., Hermsen, F., Decker, Stefan: k - RDF-Neighbourhood Anonymity: Combining Structural and Attribute-based Anonymisation for Linked Data. In: Workshop on Society, Privacy and the Semantic Web - Policy and Technology (PrivOn). Vienna, Austria (2017), http://ceur-ws.org/Vol-1951/PrivOn2017_paper_3.pdf
4. Hu, Z., Dong, Y., Wang, K., Sun, Y.: Heterogeneous Graph Transformer. In: The Web Conference 2020. pp. 2704–2710. WWW, ACM, Taipei Taiwan (Apr 2020). https://doi.org/10.1145/3366423.3380027
5. Hübscher, G., Geist, V., Auer, D., Ekelhart, A., Mayer, R., Nadschläger, S., Küng, J.: Graph-based managing and mining of processes and data in the domain of intellectual property. Information Systems **106**, 101844 (May 2022). https://doi.org/10.1016/j.is.2021.101844
6. Ji, S., Mittal, P., Beyah, R.: Graph Data Anonymization, De-Anonymization Attacks, and De-Anonymizability Quantification: A Survey. IEEE Communications Surveys & Tutorials **19**(2), 1305–1326 (2017). https://doi.org/10.1109/COMST.2016.2633620
7. Liu, K., Terzi, E.: Towards identity anonymization on graphs. In: ACM SIGMOD international conference on Management of data. p. 93. SIGMOD, ACM Press, Vancouver, Canada (2008). https://doi.org/10.1145/1376616.1376629
8. Mohapatra, D., Patra, M.R.: Anonymization of attributed social graph using anatomy based clustering. Multimedia Tools and Applications **78**(18), 25455–25486 (Sep 2019). https://doi.org/10.1007/s11042-019-07745-4
9. Zheleva, E., Getoor, L.: Preserving the Privacy of Sensitive Relationships in Graph Data. In: Privacy, Security, and Trust in KDD. pp. 153–171. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78478-4_9
10. Zhou, B., Pei, J.: Preserving Privacy in Social Networks Against Neighborhood Attacks. In: International Conference on Data Engineering. pp. 506–515. ICDE, IEEE, Cancun, Mexico (Apr 2008). https://doi.org/10.1109/ICDE.2008.4497459