RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
CHAIR OF COMPUTER SCIENCE 5
INFORMATION SYSTEMS
PROF. DR. MATTHIAS JARKE
PROF. DR. STEFAN DECKER

BACHELOR THESIS

Anonymization and De-Anonymization of Heterogeneous Graphs Containing Personal Identifiable Information

Felix Hermsen

Matr. Nr: 322697

July 25, 2017

SUPERVISOR : PROF. DR. STEFAN DECKER

ADVISOR : DR. BENJAMIN HEITMAN

REVIEWERS:
PROF. DR. STEFAN DECKER
PROF. DR. BERNHARD RUMPE

Eidesstattliche Versicherung

Hermsen, Felix	322697
Name, Vorname	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Bachelorarbeit mit dem Titel

Anonymization and De-Anonymization of Heterogeneous Graphs Containing Personal Identifiable Information

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum	Unterschrift

Acknowledgements

I wish to express my sincere thanks to my advisor Dr. B. Heitman, for his knowledge and his help for my thesis. His guidance, attitude and keen interest is mainly responsible for completing my work. Without him this thesis would not have been possible.
I also would like to place on record, my sincere thank you to Prof. Dr. S. Decker who took time to answer some of my questions which was of great help.
Finally I would like to express my appreciation to my friends Christa Schenk, Niklas Rieken, Carsten Stoffels and Ken Schimanski and father Joerg Hermsen for spell and speech checking, guidance and personal support.
To all relatives and friends who in one way or another shared their support – physically, morally or financially – thank you.

Abstract

We provide a new way for anonymizing a heterogeneous graph containing personal identifiable information.
The anonymization algorithm is called $k - \texttt{RDF}$-neighborhood anonymity, because it changes the one hoop neighborhood of at least k persons inside a RDF graph so that they cannot be distinguished. This preserves their privacy.
It allows us to control the loss of information in different parts of the graph to adjust the trade-off between full privacy and data utility.
We conducted an experiment, which shows that the overall loss of information is less, if the anonymization is concentrated on multiple parts of the graph instead of solely one.
Moreover, we provide a guessing based de-anonymization algorithm to test how good the anonymization protects the identity of individuals.
The result is that the anonymization algorithm protects well against the threat of re-identification, if the anonymization is strong. However, the data utility decreases, if it is strong. If the data utility is high the re-identification rate is high as well.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 Motivation

Nowadays data gets shared in an enormous amount by individuals and saved by various hosts, providing services for these individuals. How the data is saved is determined by the host. This data is shared for different reasons. For example, a subset of collected data is sold to advertising partners and other third parties by the service host for economic reasons [NS09]. Another reason is the publication of medical data by medical institutes for research purposes [RKKT14] [NS09]. This sharing of potentially sensitive information of a person creates privacy concerns and creates rising public demand for privacy protection.
To meet this demand, data gets anonymised, such that individuals have a certain degree of anonymity.
However, this is not a trivial task, because on the one hand data needs to fulfill the desired amount of privacy and on the other hand the data needs to be useful for data mining. Otherwise third parties would lose interest in that data and for some services like social network sites the sharing of data is the essential business model such that the service remains free for users [NS09]. Therefore, there needs to be a trade off between full privacy and data utility [ZG08].
Depending on the dataset format different approaches have been developed, see e.g [ZPL08] [SWE02]. We distinguish between two kinds of dataset formats. The first dataset format does not have a graph structure. Therefore, this dataset is a collection of isolated data points. Throughout this thesis we refer to those datasets as relational datasets. An example for a relational dataset is a health care dataset saving the name, age, zip-code and diseases of multiple persons.
The second dataset format has a graph structure. For instance, a dataset representing a social network has a graph structure, since relationships between users have to be projected.
Research regarding the anonymization of relational datasets is a well researched area [LT08] [SWE02] [LLV07]. Less well researched is the anonymization of datasets having a graph structure. [ZG08] [LT08].
To test anonymization techniques researchers have started to develop de-anonymization techniques to verify the claim that these approaches grant sufficient privacy or to test how vulnerable anonymized data is [AMS+16].
Therefore, we decided to make this bachelor thesis primarily about the anonymization of datasets having a graph structure to contribute to better understanding about the problem of anonymization and test our approach with a de-anonymization approach.

Furthermore, as for the different dataset formats we distinguish between two classes of graphs. The first class is called homogeneous graphs. Homogeneous graphs are undirected graphs, which only have one node type and one edge type. An example is a simple modeled social network, where a node refers to a person and an edge to a friendship relation.
The second class is called heterogeneous graphs. Heterogeneous graphs are labeled directed graphs. Moreover, edges and nodes can have different types. Hence, heterogeneous graphs have the advantage that much more information can be expressed with them.
These graphs are for example used in Linked Data. Specifically, the Resource Description Framework as a part of Linked Data is a data model, which can be interpreted as a heterogeneous graph. One of the core ideas of Linked Data is to build a globally accessible heterogeneous graph based on standards like `RDF` to build a semantic web [HB11]. Some scientists envision the semantic web as the future of the current web [RKKT14].
An analysis of social networks in the semantic web has resulted in the knowledge that there is a dramatic increase in the amount of social information shared [DFJ04]. Moreover, the authors of this study believe that social information in the semantic web will evolve to a scale free network, which is the topology of a social network. Therefore, people who want to share their social information but do not want to disclose their identity are potentially at risk to be de-anonymized. Therefore, anonymization is important to protect those individuals.
This combined with the observation that social networks are growing in popularity and the rising demand for privacy makes research connecting these two fields very interesting.
Hence, we choose as our topic the anonymisation and de-anonymisation of heterogeneous graphs containing personal identifiable information.

1.2 Thesis Goals

The first goal is to do a proper literature research such that we have a good understanding about the topic. After this we plan to achieve the goal for the anonymization and then the goal for the de-anonymization.

1.2.1 Use Case

The Resource Description Framework as part of Linked Data can be used to build an open accessible heterogeneous `RDF` graph. Therefore, if this graph contains personal identifiable information it is sometimes desirable to anonymize only a part of the graph. Hence, we aim to enable partial anonymization as well.

For example, imagine a heterogeneous RDF graph, which describes people and their political views. This graph could be of public benefit, because it could be used to automate the process of calculating statistics. For instance, what percentage of people in a certain area agrees with gay marriage. However, this is sensitive information about an individual and should not be public knowledge. As a consequence individuals should be anonymized so that their privacy is preserved and data mining on the anonymized heterogeneous graph is still enabled. The heterogeneous graph may contain individuals, which are public figures like celebrities or politicians. Especially the political views of politicians should be of public knowledge so that regular people can look them up and potentially vote for them, if they agree with their political views. Thus it is desirable to only anonymize regular people but not politicians and celebrities.

1.2.2 Goal for the anonymization

The goal for the anonymization is to develop and implement an anonymization approach to anonymize heterogeneous RDF graphs containing personal identifiable information with regard to the trade-off between data utility and full privacy. The anonymization approach will be based on existing research.
The trade-off will be measured by a metric we call information loss. By running an experiment, which executes the anonymization approach on different datasets and different input parameters, which change the anonymization output, we want to study what influence these parameters have. In addition to this we hope to provide a better understanding about the problem of anonymization of heterogeneous graphs and contribute to a sparsely researched area.
The anonymization of heterogeneous graphs is more challenging than the anonymization of homogeneous graphs, because heterogeneous graphs have more than one edge and vertex type.

1.2.3 Goal for the De-anonymization

The goal for the de-anonymization is to develop and implement a de-anonymization approach which the anonymization approach is supposed to protect against. The success of this de-anonymization approach is measured by a metric we call re-identification success rate.
During the experiment each anonymized dataset will be de-anonymizsed with different parameters. By doing so we want to test how good the anonymization and the de-anonymization is and what possible weaknesses and strengths each approach has.

1.3 Outline

The rest of this document is organized as follows:
In chapter 2 we give an overview over related work. Furthermore, we present papers on which approaches we have developed are based upon and give key definitions we extracted from it.
Chapter 3 illustrates our conceptual approach. Its purpose is to give a broad overview about what we have done. Chapter 4 explains the data generation approach, Chapter 5 the anonymization approach and chapter 6 the de-anonymization approach.
Afterwards chapter 7 explains briefly how we implanted the previous presented approaches and which tools we used to achieve the goals.
In addition to this, chapter 8 is going to explain the experiment and the results of thereof.
We close with chapter 9 which summarizes this thesis and presents ideas for future work.

2 Related Work

This chapter gives an overview about some of the research we have reviewed and which is important to understand the topic. Specifically it covers research on anonymization and de-anonymization of datasets having a graph structure.
We refer to this research as anonymization and de-anonymization of graphs. Structurally it is divided into two fields. The first field covers research on homogeneous graphs. The second field covers heterogeneous graphs. In this field there is not as much research conducted as in the first field.
To the best of our knowledge most available research in the area of anonymizing and de-anonymizing graphs has been conducted on social network graphs. Therefore, we present only research trying to preserve the privacy of users inside a social network graph or disclose it. We begin this section by giving some background about an anonymization approach, which is the foundation for the majority of anonymization approaches. Then the concept of background knowledge is introduced. After this we define homogeneous graphs. Based on this definition we illustrate research covering anonymization and de-anonymization of homogeneous social network graphs. We do the same for heterogenous graphs.

2.1 Background

2.1.1 Anonymization

Anonymization emerged as relational datasets had to be shared and the privacy of entities inside those datasets had to to be protected. [SWE02].
The LINDDUN methodology [DWS+11] can be used to find important privacy threats in relation to releasing data containing private information. Those are identifiability, link-ability and disclosure of information of an entity. The most important one in this setting is identifiability.
In addition to that, the most famous idea protecting against the identifiability threat is the idea of k-anonymity [SWE02]. The input is a single table, where a row represents an entity and a column an attribute. For instance, a relational health care dataset could be composed out of 4 attributes: the name, the age, the living place and the disease of a person.
The idea is that k rows inside a table cannot be distinguished from each other. This means from the perspective of an adversary k rows look identical [SWE02]. This process groups k rows

together and forms an equivalence class. Therefore, the identity of an entity is protected, its privacy preserved and the ability to do data mining on this dataset enabled.
A dataset is said to be k-anonymous, if and only if each row is indistinguishable from at least $k-1$ other rows. For this, a row is divided into 3 parts, into its identifiable attributes (IA), quasi identifiable attributes (QIA) and sensitive attributes (SA) [SWE02]. IAs are suppressed (removed), since they can identify the entity directly. IAs are for example the name or the social security number of a person.
QIAs are generalized. Generalization is the process to blur QIAs of a row such that the data has still some utility left. This is done because a QIA does not directly identify an entity. However, QIAs can identify an entity when they are combined. For instance, a study conducted in the United States of America found that 87% of the population is uniquely identified by the combination of three attributes [SWE02]. These attributes are the gender, the date of birth and the 5-digit zip-code. That is why it is important to generalize QIAs and it is not sufficient to just suppress identifiable attributes.
The idea of k-anonymity ignores sensitive attributes, because these attributes are important for the dataset to be released in the first place. For example, if a medical institution wants to conduct research on disease patterns in certain areas the attribute "disease" is very important. However, the identity of a person is not important. Therefore, hospitals or other institutions releasing such information to the public in order to enable public institutions to do research, ignore sensitives attributes and they remain unchanged. The effect is that items inside an equivalence class can be distinguished based on their sensitive attributes.
By suppressing and generalizing rows information is lost. The computation of the anonymization of a dataset with k-anonymity, where the information loss is minimized is NP-complete [MGKV04]. However, there are approximation algorithms and algorithms, which are exponential only in the number of attributes [CDFS07] and can be used, if information loss needs to be reduced.
ℓ-diversity [MKGV07] and t-closeness [LLV07] are anonymization approaches, which change sensitive attributes in each equivalence class, such that the privacy of entities is even more protected. This is done because advanced de-anonymization approaches can exploit the fact that sensitive attributes are unchanged. On the contrary this has the effect that the data utility is going down. The final output of an anonymization is called released data [ZPL08].

2.1.2 De-anonymization

As mentioned above advanced de-anonymization approaches exploit the fact that sensitive attributes remain unchanged. In this thesis we are focusing on approaches exploiting the fact that quasi identifiable attributes are generalized and not suppressed.
Specifically we are focusing on an attack called background knowledge attack to which k-anonymity is vulnerable against [MKGV07]. Background knowledge in this setting is a union between at least one identifiable attribute and one quasi identifiable attribute. This information

is not generalized or suppressed.
By combining background knowledge and anonymized data in different ways identifiable attributes which have been suppressed in the process of anonymization can be reconstructed. We refer to this process as re-identification, since identifiable attributes uniquely identify an entity.

2.2 Homogeneous Graph: Definition

Homogeneous graphs are simple representations of datasets having a graph structure.
The core of a homogeneous graph is that it has one node type and one edge type. Under the assumption that we cover only social network graphs we define a homogeneous social network graph as follows:

Definition 1
$G_{ho} = (V, E, A_v, A_e, f_v, f_e)$ is said to be a homogeneous social network graph, if V is a set of persons represented by vertices and E a set of friendship connections between persons represented by edges. A_v is the set of all attribute values a person can have, while A_e is the set of attribute values an edge can hold. $f_v : V \rightarrow P(A_v)$ is a labeling function assigning attribute values to a vertex. $f_e : E \rightarrow P(A_e)$ is a labeling function assigning attributes to an edge.

This definition is extracted from "A Brief Survey on De-anonymization Attacks in Online Social Networks " [DZWG10]. In addition, it also suits different definitions we have found in the related work for homogeneous anonymization.
We note that by this definition, it is possible to have different types of edges and nodes. This is classic for a social network and for example modeled in the anonymization approach of [ZG08]. For instance, in a social network there could be different types of friendships like starred friends and casual friends. Similarly you can think of a case for users. However, they always represent persons or friendships, which is one of the key differences between homogeneous social network graphs and heterogeneous graphs.

2.3 Anonymization of Homogeneous Graphs

Some research in this section assumes that $A_v = A_e = \emptyset$. This means that persons represented by vertices have no attributes and all edges are interpreted as the same relationship. This is done to solely focus on the structural part. Consequently, the task to anonymize a social network is simplified.
Therefore, the main research focus of almost all explained approaches in this section is to change the structure of the graph in a way to strengthen the privacy of nodes or sensitive relationships.

The paper "A brief Summary on Anonymization Techniques for Privacy Preserving Publishing of Social Network data" [ZPL08] gives a broad overview about available research and divides research regarding the topic of this section into two categories. The first category is called clustering based approaches and the second one is called graph modification approaches.

2.3.1 Clustering Based Approaches

The idea of clustering based approaches is to summarize vertices to one (clustered) super vertex. These approaches generally will shrink the graph but also remove all structure between them [ZPL08].
For instance, the work of Hay et al. [HMJT08] assumes that $A_v = A_e = \emptyset$. Their vertex clustering approach mimics k-anonymity and wants to protect against re-identification attacks. They achieve this by clustering vertices, which have structural similarity. They then publish graph metrics like the number of vertices in each cluster (super vertex) and densities inside and between clusters [ZPL08]. Zheleva et al. [ZG08] propose two clustering based anonymization approaches. However, the goal of these approaches is not to prevent re-identification but to protect sensitive relationships. The corresponding threats are called link-ability and disclosure of information [DWS+11]. A combination has been proposed by Campan and Truta [CT08].

2.3.2 Graph modification approaches

A graph modification approach relies on the deletion and addition of nodes or edges to change the structure of a graph to satisfy different conditions.
[ZPL08] divides graph modification approaches into **randomized graph construction approaches** [LT08] and **greedy graph modification approaches** [ZP08]. For example the idea of Liu et al. [LT08] is called k-degree anonymity. They assume $A_v \cup A_e = \emptyset$. This idea demonstrates one potential adjustment of k-anonymity. It is a randomized graph construction approach, which aims to modify the graph, such that every vertex has the same degree as at least $k-1$ other vertices. This is achieved by computing the smallest number of edges which have to be added so that k vertices have the same degree. Afterwards that number of edges is randomly added to satisfy the condition. This approach minimizes the loss of information in an efficient way.
Another example is k-neighborhood anonymity by Zhou et al. [ZP08], which is satisfied, when at least k nodes have the same one-hop neighborhood. We later on define the one-hop neighborhood. They assume that $A_v \neq \emptyset$ and $A_e = \emptyset$. They limit it to the one-hop neighborhood because of complexity reasons connected to the graph isomorphism problem.
Their approach is categorized as a greedy graph modification approach, since they greedily add edges to similar neighborhoods to make them the same. In addition to this, they search for

similar neighborhoods to reduce information loss.
The work of Feder et al. [FNT08] extends the idea of k-neighborhood anonymity. Their idea is to call a graph (k, ℓ)-anonymous, if for every vertex in the graph there exist k other vertices, which share at least ℓ of its neighbors.
To achieve this, they present optimal algorithms and approximation algorithms. Optimal algorithms are NP-complete.
A different approach by Das et al. [DEE10] not stated in [ZPL08] covers weighted graphs and also proposes a version of k-anonymity for graphs. They call the algorithm k-possible anonymity.

2.4 De-anonymization of Homogeneous Graphs

To find associated work we reviewed two survey papers presenting de-anonymization approaches. The first survey paper "A Literature Survey and Classifications on Data Deanonymisation" [AMS+16] is primarily about a de-anonymization framework to classify de-anonymization approaches. Based on that framework they created a list of de-anonymization approaches. The second survey paper "A Brief Survey on De-anonymization Attacks in Online Social Networks" [DZWG10] discusses de-anonymization approaches targeting social networks and defines them either as mapping based approach or a guessing based approach.
Mapping based approaches map an entity in the background knowledge (BK) to an entity in the released data (RD). Guessing based approaches on the other hand map an entity in the RD to an entity in the BK.
Based on that knowledge, we selected three different de-anonymization attacks, which we present in the following. Moreover, we group them based on their LINDDUN [DWS+11] privacy threat.
Furthermore, we assume that homogeneous de-anonymization approaches take an anonymized social network graph as input.

2.4.1 Link-ability

The idea of Backstrom et al. [BDK07] is often stated as the reason to develop structural anonymization techniques.
This is because they showed that structural analysis of a social network graph with $A_v = A_e = \emptyset$ is enough to find a previously inserted subgraph. This subgraph is hidden for the data owner so that it cannot be removed easily. Out of this the adversary can disclose hidden links between two entities. This is achieved by connecting entities from the inserted subgraph to some entities and disclose their hidden friendship relations after the release of the entire graph.
These kinds of attacks are called analogous to cryptanalysis *active attacks*. This means the

adversary tries actively to affect the data in order to facilitate deciphering [BDK07].
On the other hand, there are *passive attacks* in which the adversary is passive. This means that the adversary is not actively trying to affect the data. In passive attacks the adversary tries to find himself, other individuals or subgraphs. These approaches intuitively yield on average worse results than active attacks.

2.4.2 Disclosure of information

Zheleva et al. [ZGAM09] uses the correlation that friends often share same or similar interests. Their graph model is that V is divided into two parts, into public nodes and private nodes. Public nodes serve as background knowledge while private nodes represent anonymized nodes. Furthermore, they assume that $A_v \neq \emptyset$ and $A_e = \emptyset$. For each public node $|f_v(v)| = 1$ and for each private node $|f_v(v)| = 0$.
Their algorithm tries to predict the correct attribute value for a private node based on its public neighborhood. This is done by computing a score for each possible attribute value. Consequently the most fitting value represents the disclosure of information. We note that this in general does not re-identify private nodes. However, if this algorithm is called multiple times, a set of disclosed information can be produced, which then ultimately can lead to re-identification as explained in the background section.

2.4.3 Identifiability

The idea of Narayanan et. al [NS09] is classified as a seed and grow graph matching attack by the first survey paper [AMS+16]. In this approach two social network graphs are linked together with the help of a machine learning algorithm. One graph serves as background knowledge and the other as the released graph.
Their algorithm is divided into 2 phases. In the first phase, the attacker utilizes the previously stated fact that a known subgraph can be found in the released dataset. This subgraph is called the seed network. It is important that the background knowledge contains the seed network.
Then the second part starts. It is called the propagation phase. In this phase all entities connected to the seed network in the background knowledge are mapped to the best matching entities connected to the seed network in the released graph. This is done by computing a matching score. If the matching score is below or equal a threshold, an entity will be matched.
By matching thus entities all information from mapped entities in the background knowledge is transferred to mapped entities in the released dataset. Therefore, the seed network is extended. The propagation step is recalled until convergence. This re-identifies all mapped entities.

2.5 Heterogeneous Graph: Definition

We previously defined homogeneous social network graphs based on research we have found. Now we do the same for heterogeneous graphs. We propose a definition for a heterogeneous Resource Description Framework (RDF) graph, based on the work of [RKKT14] [Li16], [AMSO14], [RGG15] and [HB11].
The reason we have focused on research dealing with the anonymization and de-anonymization of RDF graphs, is that we chose this data model to represent a heterogeneous graph. Therefore, we model a heterogeneous graph as a RDF graph and refer to it as a heterogeneous RDF graph. Before defining a heterogeneous RDF graph we define a statement.

Definition 2
A statement is a triple $t = (s, p, o)$, where $s \in (I \cup B)$, $p \in P$ and $o \in (I \cup B \cup L)$. s represents a subject, p represents a predicate and o represents an object. I denotes a set of unique resource identifiers, P denotes a set of predicates , B the set of blank nodes and L the set of literals.

A unique resource identifier (URI) or sometimes international resource identifier (IRI) is a string representing a HTTP link [HB11]. HTTP links are used so that people can look them up [1]. We refer to IRIs also as URIs, because it does not make a difference in the scope of this thesis. A predicate is also a string representing a HTTP link and used for edges, while URIs are used for vertices.
For example, http://www.w3.org/People/Berners-Lee/card.cm is an URI.
We call subjects resources and predicates properties. Blank nodes are special resources, which have no URIs. However, they can have blank node identifiers. These ones are disjoint from URIs [HB11]. A blanked node is also called an anonymous resource.
Furthermore, a literal is a string and can be allocated to a different data type. We note that it is not a resource. Therefore, a literal can never be a subject. It can only be an object. For example, a literal can be an integer, double or simply just a string. To make the definition easier we say without loss of generality that all literals are strings.
For instance, a statement describing the name with the predicate p of a specific person uniquely identified by the subject s could look like this:

$s =$ <http://www.w3.org/People/Berners-Lee/card#cm>,
$p =$ <http://xmlns.com/foaf/0.1/name>,
$o =$ "Coralie Mercier".

[1] https://www.w3.org/DesignIssues/LinkedData.html

More formally this means that a statement forms a relationship between a subject, a predicate and an object [HB11]. This relationship can be interpreted as a relationship between two nodes through an edge forming a graph. A graph generally consists of multiple statements. This means a heterogeneous graph is a set of statements.
This graph is defined as follows:

Definition 3
Let $t = (s, p, o) \in T$ be a statement, where T is the set of statements. $G_{he} = (V, E, f_v, f_e)$ is said to be a heterogeneous RDF graph, where f_v, f_e represent labeling functions giving a vertex $v \in V$ or an edge $e \in E$ a label, if and only if the following conditions are satisfied for each $(s, p, o) \in T$

- *For s there exists a vertex $v_1 \in V$, such that $f_v(v_1) = s$.*
- *For p there exists an edge $e \in E$, such that $f_e(e) = p$.*
- *For o there exists a vertex $v_2 \in V$, such that $f_v(v_2) = o$.*
- $f_v(v_1) \neq f_v(v_2)$
- *v_1 is connected with v_2 through e.*

It is important to note that a resource can be subject and object in different statements.
Moreover, each vertex associated with a resource has a type. The type of a resource is given by a statement. This statement has the resource as a subject, a special edge labeled type as a predicate, and another resource as an object, which label is the type of the subject. In addition to that, each edge has a type, too. The type of an edge is given by its label.
All types are defined in a vocabulary. A vocabulary is a collection of properties and resources. A graph can have multiple vocabularies so that node and edge types are theoretically not limited. In the previous example the type of the subject is person.

We call this graph heterogeneous, because it is a graph composed of different types of edges and vertices. On the other hand a homogeneous graph is composed of one vertex type and one edge type.

2.6 Heterogeneous Graph: Anonymization

The research paper from Radulovic et al. [RGG15] proposes an anonymization framework for the Resource Description framework. They state that since an increasing amount of RDF data is shared, privacy issues are expected. Moreover, there are already existing privacy concerns.
For example, the paper “Privacy Concerns of FOAF-Based Linked Data” by Decker et al. [NHD09] discusses that spam e-mails could be generated out of an FOAF based dataset. One of

the concerns is that these e-mails are so specific about details of a person's life that they have a high chance to trick this person into clicking a malicious link.
This is done by abusing the fact that a dataset using the (Friend of a friend) FOAF vocabulary contains personal identifiable information.
Radulovic et al. [RGG15] proposed k-RDFanonymity. The idea is similar to k-anonymity. The idea is that a resource can not be distinguished from at least $k-1$ other resources. The work does not state any pseudo code and does not refer to any implementations.
However, they describe different anonymization operations, which can be used to implement k-RDFanonymity [RGG15].
The approach targets a subset of resources. They refer to them as entities of interest (EOI). They state that anonymizing heterogeneous RDF graphs are more complex than homogeneous graphs, because information regarding the EOI can occur in the different places and forms [RGG15]. They can occur in the subject URI, in the data type value, subject URI, object property value and more complex scenarios [RGG15].
Furthermore, they describe one information loss metric similar to the one used for k-neighborhood anonymity [ZP08].

The master thesis of Zhuyan Lin with the title "From Isomorphism-Based Security for Graphs to Sematics-Preserving Security for the Resource Description Framework (RDF)" [Li16], discusses four different anonymization strategies for RDF. One of them is k-neighborhood anonymity. Another one is k-automorphism. The idea is that k vertices are isomorphic. They prove that it is the strongest one. This means k-automorphism satisfies every other approach they present, including k-neighborhood anonymity. However, they also prove that k-automorphism is NP-hard and therefore not usable for big data.

Finally, we want to present the work of Rachapali et al. [RKKT14]. They extended SPARQL with a new query form called SANITIZE, which consists of a set of sanitization operations and used to sanitize an RDF graph.
They specifically make the following statement: The provided language will help in implementing privacy features for RDF data. For this they present pseudo codes for node, edge and path sanitization.
To achieve sanitization, two important steps have to be taken. The first step is to automate the generation of replacement values and the second step is to apply the sanitization consistently in the entire graph.

2.7 Heterogeneous Graph: De-anonymization

The paper "Deanonymisation in Liked Data: A research Road map" [AMSO14] identifies ares of significant concerns. The authors of this paper are the same persons, who created the de-anonymization framework mentioned in section 2.4.
The biggest concern is that de-anonymization attacks will work even better in the semantic web than for example on a social network graph. This is because de-anonymization attacks try to match data [NS09]. Moreover, they state the fact that data discovery is a core feature in Linked Data. It uses co-reference resolution to match `URIs`describing the same entity. This principle is known in the context of de-anonymization as a linkage attack.
Therefore, we can deduct that some of the core principles of Linked Data lead to better de-anonymization.
Since the principles of Linked Data are not debatable, we have to find a way to ensure privacy regardless of this contradiction.

The research of Qian et al. [QLZC16] transfers a social network graph to a heterogeneous `RDF` graph. They call a heterogeneous `RDF` graph a knowledge graph.
This is done by taking a social network and interpret attribute values as vertices connected to the user vertex.
The significance of their work lies in providing a comprehensive and realistic model of the attackers background knowledge. They justify their research based on the observation, that most research done in the field of de-anonymization has a strict definition of what the adversary's background knowledge is.
Their background knowledge model is the construction of a prior attack graph, which is a knowledge graph modeling different types of background knowledge. They classify background knowledge to be common sense, statistical information, personal information and structural graph information. All this kind of information can be stored in a knowledge graph. They then show that matching the prior knowledge graph with an anonymized transformed social network graph yields successful results. This approach is equivalent to a graph matching attack on heterogeneous `RDF` graphs.

2.8 Summary of the Related Work

Overall related research stresses the importance and the growing demand for privacy. However, it also lays out that sharing datasets containing personal information can be of benefit. Therefore, it is important to protect privacy through anonymization.
In particular research in the section of anonymizing homogeneous social network graphs shows the feasibility of anonymiztaion techniques and that there are several ways how to do it. Furthermore most anonyization approaches are dedicated to do anonymization in a manner to reduce

the loss of information. On the contrary, complexity issues arise due to problems connected to the general graph isomorphism problem. Therefore, algorithms are greedy or random.
These approaches reduce the threat of re-identification based on the trade off between data utility and full privacy and are effective in certain situations.
However, research in the homogeneous de-anonymization section proves that anonymized data is still vulnerable. This means, that re-identification and information disclosure is possible. To achieve this, background knowledge is necessary. The success of an attack depends on its correctness and completeness.

Researchers concerned about the privacy in the semantic web community noticed that the strength of linked data is also the strength of de-anonymization. This is because, de-anonymization attacks match and link data, which is one of the core principles of Linked Data. Nevertheless, research in the field of anonymizing heterogeneous `RDF` graphs suggest that ideas used for the anonymization of homogeneous social network graphs like k-anonymity can be transferred to `RDF` graphs.

We identified that most anonymization approaches try to reduce the loss of information in only one structural part. However, in general social network users also have attributes. We call this the attribute part. One part of research ignores that user inside a social network have attributes. The other part of research generalizes them based on the previously computed structural anonymization. To the best of our knowledge, Zhou et al. [ZP08] is the only approach taking the attribute part and structural part simultaneously into consideration to anonymize a homogeneous social network.
We realized that the question, if anonymization should be based on only the structural part or the attribute part and the structural part or just the attribute part is still unanswered.
As we will show, to anonymize heterogeneous graph data so that the overall loss of information is the least, which is composed of multiple structural parts and multiple attribute parts, it is beneficial to take all parts simultaneously into consideration.

3 Conceptual Approach

In this chapter we present our conceptual approach.
This thesis consist of three parts:
A data generation approach, an anonymization approach we call $k-$RDF-Neighborhood-Anonymity and lastly a de-anonymization approach, which can be classified as a guessing based neighborhood attack.
Figure 3.1 gives an overview on how these approaches are connected and what they require.
In the following sections we explain this figure in more detail and why we chose this specific setting.

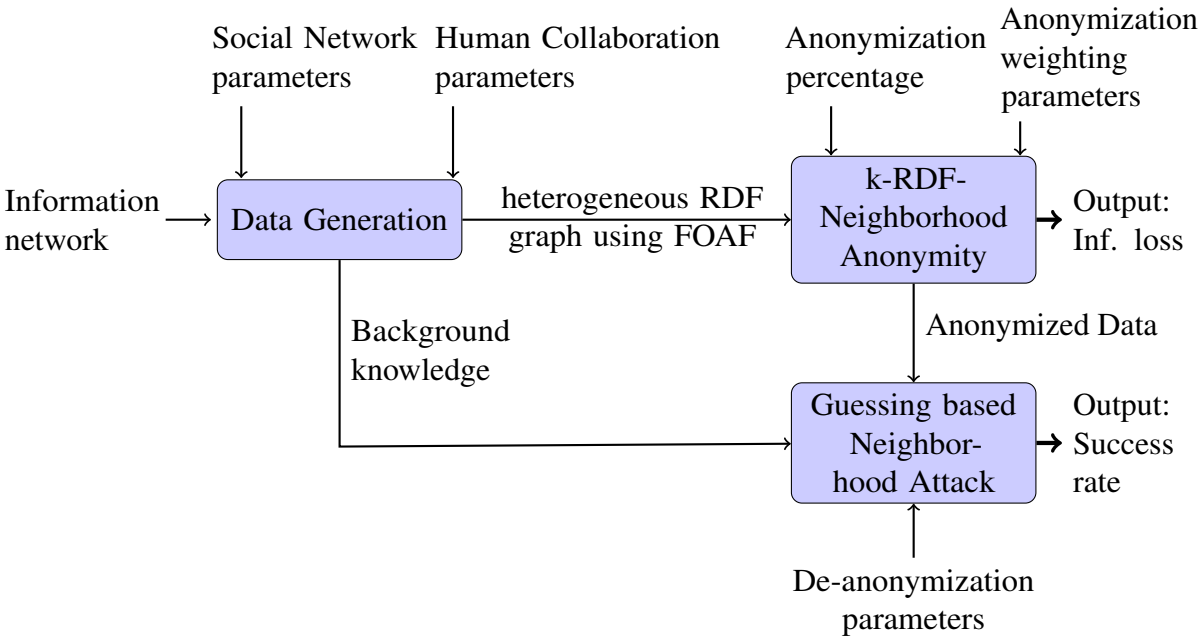

Figure 3.1: Conceptual Approach: Incoming edges depict the input and outgoing edges the output

3.1 Data Generation

As depicted in Figure 3.1 our anonymization approach requires a heterogeneous RDF graph using the Friend of a Friend (FOAF) vocabulary.
We specifically state the FOAF vocabulary, because a graph using FOAF links people and information together. It can be seen as a combination of social networks, representational networks and information networks [1].
Two key specifications in FOAF is the class "person" and the property "knows". They belong to the FOAF core, which is a list of essential classes and properties to describe a person. There are more classes and properties beyond the FOAF core. Based on our heterogeneous RDF graph definition, the property "knows" relates to an edge and the class "person" to a vertex. In addition to this, these are exactly the types a homogeneous social network consists of.
Our main objective is to develop an anonymization approach protecting the identity of a person in a heterogeneous RDF graph. The identity is revealed, because to much personal identifiable information is in that graph. Therefore, we looked for a graph containing personal identifiable information revealing the identity of a person. And since, datasets using FOAF are created to describe persons and link them and information together, we started looking for such a dataset. We planned to use a real world dataset using FOAF.
The most promising dataset we found is the Billion Triples Challenge 2009 Dataset (BTC). As the name says, the entire dataset contains at least 1 billion triples and is 17GB large. Therefore, we restricted to the reduced version with still 2.3 GB. The website we downloaded the dataset from says that this dataset is useful for testing [Har09]. In addition to this the dataset contains a lot of triples which aren't FOAF triples. Hence, we wrote a shell script using the Abstract Window Toolkit (AWT) to reduce the 2.3 GB large file to a 400 MB file only containing FOAF triples.
Since social network analysis is a difficult research branch [DFJ04] we tried to find some resources with type person having some quasi identifiable attributes and a social network structure between each other to get an overview of that dataset.
The btc dataset contains approximately 290.000 resources defined as "*persons*" and 310.000 triples having the property "*knows*" as predicate. Moreover, there are only about 100 triples having the porperty age and 300 triples having the property based_ner. We tried to find a subgraph, such that this subgraph has social network structure, and persons in it have an identifiable attribute and some quasi indetifiable attributes. The fact that we did not find such a subgraph made us reject this dataset.
In addition to that the paper Ding et al. "Analyzing Social Networks on the Semantic Web" [DFJ04] analyzed two datasets, which are different from the BTC, but also using FOAF. They found that resources type person are disjointed star-alike connected components. However, they hypothesize that with the time these datasets become scale free networks. Therefore, the work

[1] http://xmlns.com/foaf/spec

we present may become relevant in the future. Subsequently we decided to generate our own data.

The conceptual approach is to generate a synthetic FOAF based dataset describing persons. In our dataset a person has a name, an age and a living place. The name serves as an identifiable attribute, while the other two serve as quasi identifiable attributes.
Moreover, persons know each other. This forms a social network and is generated based on the social network parameters. In addition to that, persons are engaged in projects. We generate much more persons than projects meaning multiple persons are participating in one project. A person can be engaged in multiple projects. This forms a bipartite graph between persons and projects. This graph is generated based on the human collaboration parameters.
The Information network is a heterogeneous `RDF` graph using the geonames vocabulary representing city regions, districts and the federal states of Germany. This kind of information is used to give a person a living place and is important for the anonymization, since it represents a semantic hierarchy of places.
We decided two generate two types of networks to make the graph truly heterogeneous and thereby different from a homogeneous social network graph.
The background knowledge is a modified version of the previously generated graph. We assume that the background knowledge does not include false information. This is not very realistic but simplifies a lot.
However, it may be incomplete. This means that a certain percentage of edges is deleted to simulate that an adversary might not have perfect background knowledge. Identifiable attributes are not removed. This means that quasi identifiable attributes, edges belonging to the social network and edges belonging to the bipartite graph are removed.
The specific way how we generate this data is explained in detail in chapter 4.

3.2 Anonymization Approach

As stated earlier the idea for the anonymization approach is to protect the identity of resources representing persons.
The anonymization percentage parameter specifies how many resources are anonymized. Each approach we have reviewed always anonymizes the entire dataset. But for example, if a dataset contains public and private figures, the data about public figures should not be anonymized. To be even more specific a dataset containing politicians and their stand on certain political issues and private citizens and their stand on these issues is a concrete example. The identity of private citizens does not matter but the stand of politicians should be publicly accessible. Therefore, the ability to anonymize a certain percentage is desirable when talking about data which contributes to a global database.

The disadvantage is that private figures connected to public figures could be de-anonymized much easier. However, due to simplification reasons we do not design our de-anonymization approach around this.
Secondly, the anonymization approach takes anonymization weighting parameters as input. These ones are used to concentrate the anonymization on only one part or multiple parts of the input graph. The theory is that this reduces the loss of information in these parts of the graph, while still satisfying the anonymization criteria. For example, if the social network part is the most important one, the anonymization approach will try to disturb this part at least. As an expected consequence, loss of information in other parts will rise.

The conceptual approach is to develop a greedy heterogeneous graph modification approach. We did not choose a graph clustering approach, because a graph clustering approach removes all information between clustered nodes. Moreover, our objective is to release a graph instead of just graph metrics.
By reviewing the related work we learned that anonymizing complex social structures is very difficult. Therefore, we decided to restrict the anonymization to the one-hop neighborhood of resources with type person. This idea was proposed by Zhou et al. [ZP08] and was presented in 2.3.2 . In addition to that, they also introduced the idea of anonymization weighting parameters. However, their approach is not designed for heterogeneous graphs and modifications are required. To make these modifications, we include ideas presented by Radulovic et al. [RGG15]. They proposed k-RDF-Anonymity and presented different anonymization operations, which can be used to implement the idea of k-anonymity for heterogeneous graphs. That is why we call our approach k-RDF- Neighborhood Anonymity.
The idea of this approach is that the one-hop neighborhood of a resource which is going to be anonymized is the same as at least the one-hop neighborhood of $k - 1$ other resources. The anonymized data is then compared with the input graph to measure the loss of information. The specific way how we anonymize the input graph and measure the loss of information is explained in detail in chapter 5.

3.3 De-anonymization Approach

The goal for the de-anonymization is to develop a de-anonymization approach, which the anonymization approach is supposed to protect against. This serves the purpose to test if the anonymization approach protects the one-hop neighborhood. The anonymization approach protects against re-identification.
To achieve re-identification of anonymized resources, background knowledge is essential. As we presented in the related work section 2.4 and 2.7, there are two ways to do a background

knowledge de-anonymization attack. The first option is to do a mapping based attack. The second option is a guessing based attack.
We decided to do a guessing based attack, because a mapping based attack requires a search for a known sub graph. The search of such a sub graph is very complex on its own [NS09]. This subgraph then serves as a starting point for a machine learning algorithm. Due to the complexity of such an attack, we decided to take the more simple option.
In addition to that, mapping based approaches are very well studied and are proven to be very effective [NS09] [DZWG10] [QLZC16].
To the best of our knowledge, guessing based approaches are not as much studied as mapping based approaches. Therefore, we might contribute something to this research branch.

The conceptual approach is to compute probabilities on how likely it is that a resource from the input graph has a certain identity. The identity is represented by the identifiable attribute of a resource in the background knowledge. The likelihood is the difference between the one-hop neighborhood of an anonymized resource and the one-hop neighborhood of a resource from the background knowledge. The likelihood is 1, if they match perfectly and 0 if they don't match.

In the case of a guessing based de-anonymization attack multiple identities are computed. We compute the most likely candidate. However, if there are multiple identities with the same likelihood, the de-anonymization approach has to guess between those ones. If the set which is formed out of these identities is too big, the probability to do a correct de-anonymization drops significantly. To prevent such a bad guess the approach can be tuned. And instead the de-anonymization is terminated for this resource. Perfect de-anonymization takes place, if the set of identities has only one element.
Moreover, we compute the most likely candidate and add a threshold parameter to its liklyhood. Then we select all candidaates within this range. This increases the set of possible candidates and may result in a better de-anonymization success rate.
The success rate is the percentage of correct re-identified individuals. The specific way how we de-anonymize the anonymized input graph and measure the re-identification rate is explained in detail in chapter 6.

4 Data Generation

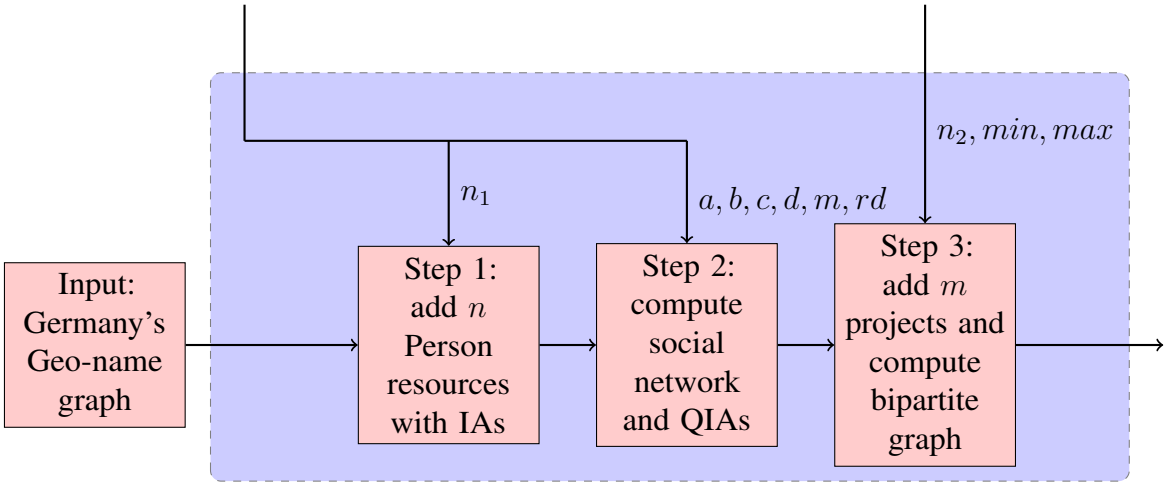

Figure 4.1: Data Generation Approach

Figure 4.1 is connected to figure 3.1 and depicts the process of our data generation approach. The process is divided into three steps.
The input is a heterogeneous `RDF` graph. We call this graph Germany's geo-name graph.
As described in the conceptual approach Germany's geo-name graph describes city regions, districts and federal states of Germany. It also includes a representation of Germany itself.
For example, city region Aachen is a city region in the Cologne district, which is in the federal state of North Rhine-Westphalia, one of the 16 states of Germany.
This graph represents a hierarchy tree. This tree has 3 levels. The level describes how many hoops a vertex is away from the root. The root or level 0 is Germany. Level 1 collects all 16 states of Germany. Level 2 is a set of all districts of all states of Germany. Lastly level 3 is the set of all cities of Germany.
Each element is represented by a resource. This resource can be interpreted as a vertex in a graph. Each vertex has the type feature and is part of the vocabulary Geonames (GN). For convenience we write `namespace:property` to indicate from which vocabulary a type is. `GN:parent_feature` is a property and describes a directed connection from a vertex of level i to a vertex of level $(i-1)$. This ultimately describes what cities, districts and states Germany has. In addition to this, the main purpose of this graph is to describe where entities of Germany represented by resources of type feature are located.

We obtained this graph by the combination of RDF documents we crawled from the geographical database. This database is available at http://www.geonames.org/ (last checked 20.6.2017). Specifically, we crawled RDF documents associated with the Geotree of Germany to the depth of three. For crawling the data we wrote a shell script using basic UNIX commands, the wget program for downloading files, and the Stream Editor (SED), which we used to modify files. How we combined RDF documents to obtain a graph is explained in the implementation chapter.

In the following sections we explain the three steps depicted in the rectangle of figure 4.1.
In the first step we generate a number of resources representing persons, which are connected among each other with an algorithm called R-Mat in the second step. The third step generates project resources and creates a random bipartite graph between persons and projects.

4.1 First Data Generation Step

The first step is to create $n = 2^{(n_1)}$ new resources and add them to Germany's geo-name graph. Each resource has an URI and is subject in a statement. Therefore, to create n resources we have to create n distinct URIs. We generate n distinct URIs as following:

1. define a preamble
2. generate n random real world human names
3. generate n distinct numbers
4. combine preamble, random name, and distinct number to build n distinct person URIs

The following URI is an example of this process and is representative for all URIs we generate

<file://Ba/generated/graph/Subgraph#1/Aleen_Ollmann#153>

The preamble is file://Ba/generated/graph/Subgraph#1/ and does not change. To avoid confusion, *Subgraph#1* comes from a different implementation for the social network structure and has become unnecessary because of the approach we present in the next section.
With the set of distinct resources we create $2n$ statements and add them to Germany's geo-name graph. In this chapter we call this graph the set of statements, since a heterogeneous graph is a collection of statements.
For each generated resource we create statements t_1 and t_2. The parameter t_1 describes the type of the resource and t_2 describes the name of the resource.

We use the property `RDF:type` from the `RDF` schema to describe the type. The `RDF` schema provides a data-modeling vocabulary for `RDF` data [1] .
The type of the generated resource r is "*person*" and since we are using FOAF to describe people its statmenet is t_1. The second statment t_2 is the statement for the name. The name is part of the `URI` of r.

$$t_1 = (s = r, p = \texttt{RDF:type}, o = \texttt{FOAF:person})$$
$$t_2 = (s = r, p = \texttt{FOAF:name}, name)$$

4.2 Second Data Generation Step

The second step is the most complex one. We generate a social network between resources of type `FOAF:person`, while simultaneously assigning quasi identifiable attributes to these resources. The input parameters for this step are a, b, c, d, m and rd.

A social network has two characteristics: a power law vertex degree distribution and a small-world characteristic [CZF].
Chakrabarti et al. [CZF] proposed an algorithm they call Recursive Matrix. This algorithm is designed to construct an adjacent matrix in a way to represent a homogeneous graph with a certain topology. The topology is specified by four input parameters. These input parameters are a, b, c, d and represent probabilities. The condition for these parameters is the sum of them to be 1.
Chakrabarti et al. [CZF] states that $\frac{a}{b} \approx 3$, $\frac{a}{c} \approx 3$ and $a > d$ generate a network which has the above mentioned properties.
The experiment by Zhou et al. [ZP08] also used this method. They used the parameters $a = 0.45, b = 0.15, c = 0.15, d = 0.25$ to generate a social network. Since these parameters achieve exactly what we want, we adopted them.
The algorithm starts with a $n \times n$ matrix, where all entries are 0. A 0 in the matrix means that there is no connection between the vertex associated with the column number and the vertex associated with the row number. A 1 means that there is one.
The algorithm divides the matrix in four equal sized matrices: a top left matrix a top right matrix a bottom left matrix and a bottom right matrix. Then the algorithm draws a random number between 0 and 1. If the random number is less equal than a the algorithm is recursively called with the top left matrix as input. If the random number is less equal than $a + b$ and bigger a it is recalled with the top right matrix. If the random number is less equal than $a + b + c$ and bigger $a + b$ it is recalled with the bottom right matrix. And finally, if the random number is less equal

[1] https://www.w3.org/TR/rdf-schema/#ch_introduction

than $a + b + c + b = 1$ and bigger $a + b + c$ it is recalled with the bottom right matrix.
The recursion stops, if the the input matrix is a 2×2 matrix. A last time a random number is drawn and if the value of that cell is 0, it is changed to 1. If it is 1, the algorithm starts again until a 1 has been assigned. In addition to that, a vertex can not be connected to itself through an edge. This means that the diagonal of the output matrix is always 0.
This process is repeated m times, which means that the output matrix represents a graph with m edges.
For each 1 in the matrix we generate two statements, let r_1 be the resource associated with the column number and r_2 the resource associated with the row number.
The generated statements are:

$$t_1 = (s = r1, p = \texttt{FOAF.knows}, o = r2)$$

$$t_2 = (s = r2, p = \texttt{FOAF.knows}, o = r1)$$

The property `FOAF:knows` describes that two persons know each other.
Moreover, this algorithm automatically creates communities. The top left matrix and the bottom right matrix are associated with distinct communities. Therefore, the recursion depth is an indicator for the size of a community.
We create a statement representing the living place at recursion depth rd for each person associated with the input matrix . The living place lp is a random level 3 resource from Germany's geo-name graph and is described by a statement using the property `FOAF:based_near`. Therefore, the statement for the living place is:

$$t_3 = (s = r, p = \texttt{FOAF.based\_near}, o = lp)$$

The living place is the same for all those persons meaning there are at least $2^{(n_1 - rd)}$ persons having the same living place.
In addition to that, we assign a random age which is based on a uniform distribution between 20 and 80 to each of those persons. The associated statement is:

$$t_4 = (s = r, p = \texttt{FOAF.age}, o = age)$$

Lastly all generated statements are added to the set of statements.

4.3 Third Data Generation Step

The last step is to generate n_2 project resources and compute a bipartite graph between persons and projects. The input parameters for this step are n_2, min and max.
We generate `URIs` representing projects as follows:

1. define a preamble
2. generate n_2 distinct numbers
3. combine preamble and distinct number to build n_2 distinct project `URIs`

The following `URI` is an example of this process and is representative for all project `URIs` we generate.

<file://Ba/generated/graph/Subgraph#1/project#1>

For each generated project resource r_p we construct a statement t specifying the type. The constructed statement, which is added to the set of statments is:

$$t = (r = r_p, p = \texttt{RDF:type}, o = \texttt{FOAF:project})$$

The bipartite graph is generated as follows:
For each generated resource r_1 with the type FOAF:person we draw a random number x between min and max. The number of projects r_1 involved is x. Therefore, we construct x statements $t_1, ..., t_x$ for r_1. For each of those statements $t_i : i\ in\{1, ..., x\}$ we select a random distinct resource r_2 with the type FOAF.project. The resulting statement, which is added to the set of statements is

$$t_i = (s = r_1, p = \texttt{FOAF.currentProject}, o = r_2).$$

4.4 Summary of the Data Generation

We generated 2^{n_1} persons and assigned them a random real world name. Then we computed a social network among them, which consists of m edges. To achieve this, we used an algorithm called Recursive Matrix. In particular this algorithm uses four input variables, which can be adjusted to generate different types of networks. We adjusted them so that it computes a small world network having a power law vertex degree distribution. In addition to that, the R-Mat algorithm naturally calculates communities. In a community, which exactly consists of $2^{(n_1 - rd)}$ persons, each person has the same living place and a random age based on a uniform distribution between 20 and 80.
Furthermore, we generated a random bipartite graph between persons and projects. Each person is involved in at least min projects and at most max projects.

5 Anonymization Approach: k-RDF-Neighborhood Anonymity

This chapter describes our anonymization approach k-RDF-Neighborhood anonymity in detail. In general all ideas presented in this section are applicable to any heterogeneous RDF graph. However, this approach is designed to anonymize graphs generated with the data generation approach we described in chapter 4.
We say a heterogeneous RDF graph $G_{he} = (V, E, f_v, f_e)$ is anonymized, if all entities of interest (EOI) are anonymized.
We define an EOI to be a vertex associated with the type FOAF:person.
In addition to that, we say an EOI is anonymized, if there are at least $k-1$ other EOIs having the same one-hop neighborhood, where $k \in \mathbb{N}$. We call the set of EOIs having the same one-hop neighborhood an equivalence class.
The one-hop neighborhood of a vertex $v \in V$ is defined as follows:

Definition 4
The heterogeneous graph $G_{1hop}(v) = (V_{1hop}, E_{1hop}, f_v, f_e)$ is said to be the one-hop neighborhood of a vertex $v \in V$ of $G_{he} = (V, E, f_v, f_e)$, if $G_{1hop}(v)$ is a subgraph of G_{he}.
Specifically, let V_{1hop} be the set of all vertices $v_o \in V$ that are directly connected with v including v itself. Moreover, let E_{1hop} be the set of edges connecting v with $v_o \in V_{1hop}$. In addition to that, E_{1hop} also contains all edges, which connect vertices in V_{1hop} among one another.

As we stated in chapter 3 the approach we propose is a modified version of the approach proposed by [ZP08]. Their approach orders EOIs descending based on their degree and computes for each EOI a neighborhood code (NHC). This code is a string representing the one-hop neighborhood of an EOI. Its purpose is to reduce the complexity of comparing one-hop neighborhoods .
After this the approach computes a similarity value between the NHC of the highest degree vertex, which is marked as not anonymized, and every other NHC of not anonymized vertices. The similarity value is a value indicating how similar two one-hop neighborhoods are.
Then the smallest $k-1$ values are searched and the corresponding one-hop neighborhoods are changed so that they form an equivalence class and are marked as anonymized. This is repeated until all one-hop neighborhoods are anonymized.

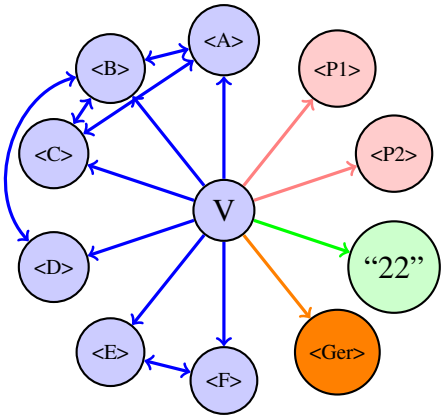

Figure 5.1: Example of an one-hop neighborhood of vertex V: blue edges represent edges labeled `FOAF:knows`, red edges `FOAF:current_project`, green edges `FOAF:age` and an orange edge `FOAF:based_near`.
Blue vertices are of type `FOAF:persons`, red of type `FOAF:project`, orange vertices of type `GN:feature` and green vertices have no type since they are literals.

This approach relies on three key algorithms:

1. The **full neighboirhood computation algorithm**, which computes a neighborhood code representing the one-hop neighborhood of a vertex.
2. The **similarity value computation algorithm**, which computes a similarity value between two one-hop neighborhoods by using neighborhood codes .
3. The **graph modification algorithm** and is used to change the one-hop neighborhoods of input vertices so that they form an equivalence class.

In the next three sections we explain our modified versions of these algorithms and in the fourth section we combine these algorithms to anonymize a heterogeneous `RDF` graph.

5.1 Full Neighborhood Code Computation Algorithm

Since our approach frequently compares one-hop neighborhoods, it would be inefficient to frequently conduct isomorphism tests. The reason for that is that there is no known polynomial time algorithm for the general graph isomorphism problem [ZP08]. Therefore, we generate a string we call the full neighborhood code ($FNHC$), which use to compare neighborhoods in a more efficient way. However, the computation of that string is still very time consuming.
To compute the $FNHC$ we divide the input graph $G_{1hop}(v) = (V_{1hop}, E_{1hop}, f_v, f_e)$ into three neighborhoods:

1. **The attribute neighborhood** $G_a(v) = (V_a, E_a, f_v, f_e)$ is composed of all vertices connected to v through edges $e \in E_{1hop}$, where $f_e(e) =$ `FOAF:age` or $f_e(e) =$ `FOAF:based_near`. The set of edges E_a is empty. Assuming that G_{he} is the output of the approach described in chapter 4, each `EOI` has only one age and one living place. Hence, $G_a(v)$ is composed of 2 isolated vertices.
2. **The human collaboration neighborhood** $G_{hc}(v) = (V_{hc}, E_{hc}, f_v, f_e)$ is composed of all vertices connected to v through edges $e \in E_{1hop}$, where $f_e(e) =$ `FOAF:current_project`. The set of edges E_c is empty.
3. **The social network neighborhood** $G_s(v) = (V_s, E_s, f_v, f_e)$ is composed of all vertices connected to v through edges $e \in E_{1hop}$, where $f_e(e) =$ `FOAF:knows`. $E_s \subset E_{1hop}$ is composed of all edges $(v_i, v_j) \in E_{1hop}$, where $v_i \neq v$ and $v_j \neq v$. Moreover, $f_e(e) =$ `FOAF:knows`.

For each neighborhood we generate a neighborhood code (NHC). Afterwards, we put them together in a defined order to obtain the $FNHC$. Each neighborhood is in general a forest. This means, one of the neighborhoods is divided into multiple unconnected subgraphs.
The $NHCs$ for $G_a(v)$ and $G_{hc}(v)$ are trivial, because these neighborhoods are composed of isolated vertices.
We call NHC_a the corresponding NHC for the attribute neighborhood, which is the resulting string by ordering all strings $f_v(v_i)$ lexicographically, where $v_i \in V_a$.
The NHC for the human collaboration neighborhood we call NHC_{hc}, which is the resulting string by ordering all strings $f_v(v_i)$ lexicographically, where $v_i \in V_{hc}$.
By NHC_s we denote the string for the social network neighborhood $G_s(v)$. Its computation is much more challenging, since vertices in V_s are connected among each other. $G_s(v)$ is in general a forest composed of multiple subgraphs called $G_{s_i}(v)$.
In Figure 5.1 for example, vertices A, B, C, D and E, F build a forest of two subgraphs.
Zhou et al. [ZP08] picked up the idea developed by Xifeng et al. [YH02], which proposed a way how to calculate a unique string representing such a neighborhood.

Their idea is to calculate all possible *depth first search trees* (DFS-trees) for each subgraph G_{s_i} and vertex $v \in V$. After that they are encoded and the minimum DFS code is searched. The ordered concatenation of the minimum DFS codes of each subgraph results in the string NHC_s.
$T_i(G_{s_i}, f_v(v)) = (V, E_t, f_v, f_e, l_v, l_e)$ represents one possible DFS-tree of graph G_{s_i} and vertex $v \in V(G_{s_i})$. The labeling functions l_v, l_e are computed with a DFS algorithm.
The labeling function l_v assigns each vertex an index $i \in \mathbb{N}$, indicating when a vertex was discovered by the DFS algorithm. In addition, the set of vertices E_t is a subset of $E(G_{s_i})$. The labeling function l_e assigns to each edge the attribute *forward* or *backward*. An edge $e = (v_i, v_j)$ has the label "*forward*", if $l_v(v_i) < l_v(v_j)$ and "*backward*", if $l_v(v_i) > l_v(v_j)$. Figure 5.2 shows three possible DFS-trees of neighborhood subgraph G_{s_1} of the example graph depicted in figure

5.1. T_1 and T_2 show different DFS-trees from the same vertex, while graph T_3 is an example of a different DFS-tree from different vertex of G_{s_1}.

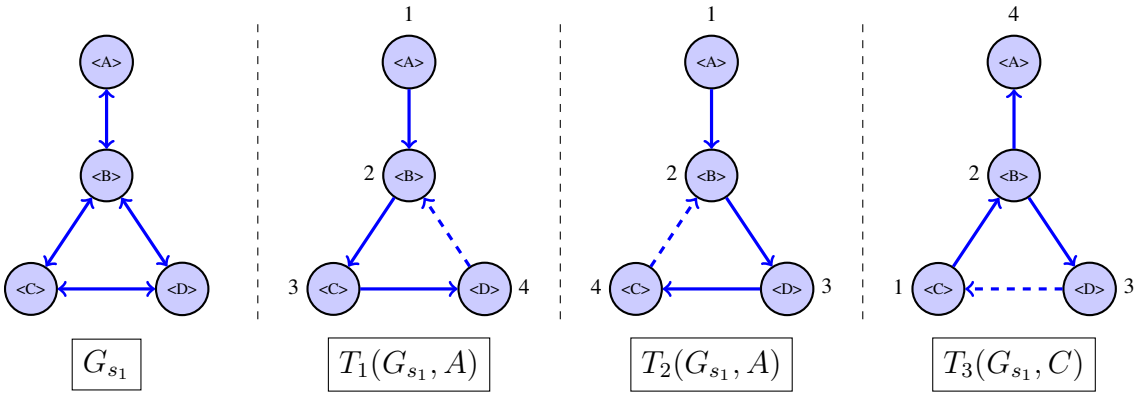

Figure 5.2: This figure depicts three depth first search trees and its origin. The origin G_{s_1} is a subgraph of the social network neighborhood of figure 5.1. Dotted edges represent backward edges, while others are forward edges. The first two are computed from the same vertex A, while the third is computed from vertex C.

A depth first search tree $T_i(G_{s_i}, f_v(v)) = (V, E_t, f_v, f_e, l_v, l_e)$ is encoded as the sequence of its linear ordered edges. We call this sequence $code(T_i)$. An encoded edge $e = (v_i, v_j)$ is the string $s = ((l_v(v_i), l_v(v_j), f_v(v_i), f_v(v_j))$, where $v_i, v_j \in V(T_i)$. If the depth first search tree is just a single vertex the encoding is the label of the vertex, which is given by the function f_v .

Example 5.1.1
This example presents the DFS codes for the DFS-trees T_1 and T_3, which are depicted in figure 5.2.

- $code(T_1) = (1, 2, A, B)(2, 3, B, C)(3, 4, C, D)(4, 2, D, B)$
- $code(T_3) = (1, 2, C, B)(2, 3, B, D)(3, 1, D, C)(2, 4, B, A)$.

The **linear order** $\prec$ proposed by [YH02] is defined as follows:
The linear order $\prec$ is a relation between two encode edges $e_1 = (v_i, v_j)$ and $e_2 = (v'_i, v'_j)$.
We say $e_i \prec e_j$, if and only if one of the following conditions is true:

- $l_e(e_i)$ and $l_e(e_j)$ equals $forward$ and $l_v(v_j) < l_v(v'_j)$ or $(l_v(v_i) > l_v(v'_i) \wedge l_v(v_j) = l_v(v'_j))$
- $l_e(e_i) = backward$ and $l_e(e_j) = forward$ and $l_v(v_j) < l_v(v'_i)$
- $l_e(e_i) = forward$ and $l_e(e_j) = backward$ and $l_v(v_i) \leq l_v(v'_j)$
- $l_e(e_i)$ and $l_e(e_j)$ equals $backward$ and
  $l_v(v_i) < l_v(v'_i)$ or $(l_v(v_i) = l_v(v'_i) \wedge l_v(v_j) < l_v(v'_j))$

The intuition behind the linear orderer $\prec$ is that edges are sorted in the order in which they have been discovered by the depth first search algorithm. However, if the algorithm finds a backward edge it is placed before the next forward edge. If there are multiple backward edges the backward edge, which goes furthest back is placed before the other backward edges. This means backward edges have a higher priority than forward edges. Nevertheless the ordering does not influence the computation of the depth first search tree.

The min code of a subgraph graph G_{s_i} is computed by ordering the set of codes.
We say $code(T_i) \prec code(T_j)$, if there exists an edge e_i at position $p \in \mathbb{N}$ in $code(T_i)$ and an edge e_j at the same position in $code(T_j)$ so that $e_i \prec e_j$. We note that there could be multiple min codes. However, we are only interested in one, because we are just interested in the structure. For example $code(T_3) \prec code(T_1) = code(T_2)$.
The intuition is that the code with the earliest backward edge is the min_code.

We randomly pick one min code for each subgraph G_{s_i} of G_s. The resulting set of min codes are sorted in an ascending way by the following order:

We say $min\_code(G_{s_i}) < min\_code(G_{s_j})$, if at least one of the following conditions is true:

- $min\_code(G_{s_i})$ is composed of less vertices as $min\_code(G_{s_j})$
- $min\_code(G_{s_i})$ is composed of the same amount of vertices and less edges as $min\_code(G_{s_j})$.
- $min\_code(G_{s_i}) \prec min\_code(G_{s_j})$ and $min\_code(G_{s_i})$ is composed of the same amount of vertices and edges as $min\_code(G_{s_j})$.

Two social network neighborhoods are isomorphic, if their NHC_s codes are the same according to [YH02] . This also applies to the $FNHC$, which is the concatenation of NHC_a,NHC_{hc} and NHC_s (proof omitted).

Example 5.1.2
This example presents a min_{code} of the one-hop neighborhood graph, which is depicted in figure 5.1.

$$(\{[\text{“}22''], [< Ger >]\}, \{[< P1 >], [< P2 >]\}, \{[(1,2,E,F)], [(1,2,C,B)(2,3,B,D)(3,1,D,C)(2,4,B,A)]\})$$

- $NHC_a = \{[\text{“}22''], [< Ger >]\}$
- $NHC_{hc} = \{[< P1 >], [< P2 >]\}$
- $NHC_s = \{[(1,2,E,F)], [(1,2,C,B)(2,3,B,D)(3,1,D,C)(2,4,B,A)]\})$

5.2 Similarity value computation algorithm

The computation of the similarity value is a simulated anonymization, in which no values are changed. The value indicates the expected information loss. A lower value of loss of information indicates a bigger similarity of neighborhoods.
The inputs for this algorithm are two $FNHCs$ we call $FNHC(G_{1hop}(v))$ and $FNHC(G_{1hop}(u))$. Moreover, four weighting parameters called $\alpha, \beta, \gamma, \delta$.
The similarity value is composed of four parts. The **age similarity**, the **based_near similarity**, **the project similarity** and the **knows similarity**.
In the next four subsections we explain each similarity part.
The combined similarity value for two one-hop neighborhoods is computed by the following function:

$$sim(FNHC(G_{1hop}(v)), FNHC(G_{1hop}(u))) = \alpha \cdot sim_{age} + \beta \cdot sim_{based\_near} + \gamma \cdot sim_{project} + \delta \cdot sim_{knows}$$

5.2.1 Age Similarity

We obtain the age of v by selecting NHC_a from the associated full neighborhood code $FNHC(G_{1hop}(v))$. The age is the first substring of NHC_a. We call this string age_v. The same is done to obtain age_u.
We assume that age_v and age_u represent two random natural numbers in the interval $[20 - 80]$. The similarity value is based on the distance to the nearest interval both ages fall into. To calculate the distance to the nearest interval we developed a hierarchy tree, which is defined in the following.
The age hierarchy tree $H = (V, E, l_v)$ for the interval $20 - 80$, consists of 7 levels $0 - 6$. Each level is a set of vertices. Level 0 consists of $|80 - 20| = 60$ vertices each labeled with a distinct age contained in the interval set. Each remaining level $l \in \{1, ..., 6\}$ has $|v_l| = (7 - l)$ vertices. Each vertex in level l, where $l > 0$, is labeled with a distinct interval of size $10l$. Interval i at level l starts at $20 + 10i$, where $i \in \{1, ..., |v_l|\}$.
A directed edge $(v_i, v_j) \in E(H)$ exists only, if the following three conditions are satisfied:

- $v_i \in V$ is associated with level i and $v_j \in V$ is associated with level j
- $j = i + 1$
- $l_v(v_i) \subset l_v(v_j)$

The following figure illustrates an example for a concept hierarchy tree representing the interval $20 - 50$:

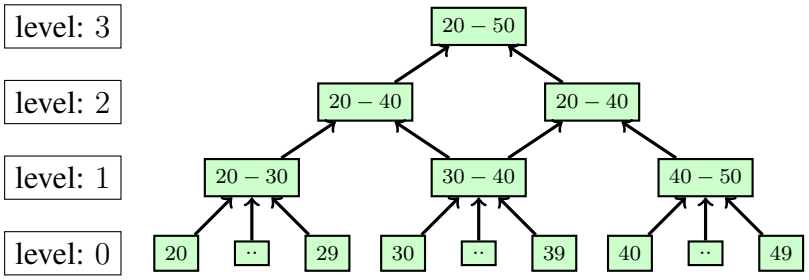

Figure 5.3: This tree illustrates an example for a numerical concept hierarchy tree representing the age in the interval $20 - 50$

Algorithmus 5.1 Pseudo Code for the Distance algorithm

```
1: procedure D_a(l(v_i), l(v_j), H = (V, E, l_v))
2:     distance ← 0
3:     while l(v_i) ≠ l(v_i) do
4:         v_i ← v_i' where (v_i, v_i') ∈ E
5:         v_j ← v_j' where (v_j, v_j') ∈ E
6:         distance ← distance + 1
7:     end while
       return distance
8: end procedure
```

The distance is computed by **algorithm 5.1** and works as follows for computing the age similarity:
The distance computation takes age_i, age_j and the age hierarchy tree $H = (V, E, l_v)$ as input. It starts by selecting two vertices $v_i, v_j \in V$, where $l(v_i) = age_i$ and $l(v_j) = age_j$. Then it checks whether these labels are the same ones. If not, the algorithm follows the only edge (v_i, v_i') to the next higher level and replaces v_i with v_i'. The same is done for v_j. At this point $l_v(v_i)$ and $l_v(v_j)$ represent intervals. This procedure is repeated until $v_i = v_j$. This results in a vertex representing the smallest interval, which includes age_i and age_j, since vertex labels are unique. The distance is the level of that interval or the amount of edges followed divided by 2.
With that distance we compute the fraction $sim_{age} = \frac{distance}{6}$. If the distance to the root element $20 - 80$ is 6 we have the highest amount of information loss. If the distance is 0, then age_i and age_j have the same value.

5.2.2 Based_near Similarity

The based_near similarity sim_{based_near} is computed like sim_{age}. However, $H = (V, E, l_v)$ is Germany's geo-name graph, which was explained in chapter 4. Moreover, the distance algorithm does not use two ages but two URIs representing two locations. These URIs are obtained by selecting the second substring of the associated FNC_a.
We note that Germany's geo-name graph is not a mathematical hierarchy like the numerical one previously defined, but a hierarchy with a predefined semantic.

5.2.3 Project Similarity

The project similarity is the easiest to compute. The string vertex v for a person is $FNHC(G_{1hop}(v))$. The second substring of this is NHC_{hc} and represents the projects v is involved in. This substring consists of URIs describing project resources. We call the set of project URIs for v P_v. In a similar way we call the set of project URIs for person u P_u. The similarity value $sim_{project}$ is the amount of how many project resources have to be deleted so that $P_v = P_u$.
Deletion of edges in this context means that we copy the $FNHC$ and modify this code instead, because we just want to simulate the change and not apply it.

5.2.4 Knows Similarity

The most challenging similarity to compute is the knows similarity, which takes the third substring of $FNHC(G_{1hop}(v))$ and $FNHC(G_{1hop}(u))$as input. We previously referred to that string as NHC_s.
Moreover, we explained in section 5.1, that $NHC_s(v)$ is composed of multiple substrings, each representing a subgraph of a social network neighborhood.
The challenge is to compute how many edges have to be deleted so that $G_s(v)$ is isomorphic to $G_s(u)$. This is the case, if $NHC_s(v) = NHC_s(u)$.
We note that we are only interested in the structure and do not demand a perfect neighborhood match. This means that $NHC_s(v)$ is isomorphic to $NHC_s(u)$, if they are equal in the sense of the definition of the linear order $\prec$. To be even more specific, only the first two entries in an edge encoding are of interest. It is not required that $G_s(v)$ and $G_s(u)$ connect to the same vertex URIs.

We say $NHC_s(v)$ is composed of n substrings and $NHC_s(u)$ is composed of m substrings. We call NHC_{s_i} to be the ith substring of NHC_s.
We compute the number of removed edges by matching isomorphic parts and remove the edges, which can not be matched.
We start with the biggest substring in $NHC_s(v)$ and $NHC_s(u)$, which are $NHC_{s_n}(v)$ and $NHC_{s_m}(u)$, since $NHC_s(v)$ and $NHC_s(u)$ are sorted in an ascending order.
If both substrings are composed of edges, meaning they represent subgraphs and not single vertices, we compute matching edges between $NHC_{s_i}(v)$ and $NHC_{s_j}(u)$.

The first case is that all edges match. Therefore, the neighborhoods are isomorphic and we can map them together and move on with the next pair of substrings.

The second case is that not all edges match. This case has two sub-cases.
The first sub-case is, that there exist two edges $e_v \in NHC_{s_i}(v)$ and $e_u \in NHC_{s_j}(u)$, which do not match. If $e_v \prec e_u$ then we remove edge e_v from $NHC_{s_i}(v)$. If $e_u \prec e_v$ we remove edge e_u from $NHC_{s_j}(u)$. Otherwise we say that e_v matches e_u.
Removing an edge from a substring NHC_{s_i} means, it gets split in two parts, if and only if the removed edge is a $forward$ edge. Splitting a substring into two parts requires an update to the corresponding NHC_s. The update results in an increase of substrings in NHC_s, if the removed edge splits the associated subgraph into two.
If the removed edge is a $backward$ edge it is just removed. We note that the update of a removed backward edge does not split the graph.

The second sub-case is that one substring is larger then the other one and all edges in the smaller substring have been matched. We resolve that by splitting the larger substring one part matches with the smaller substring. Then we update NHC_s.
If one substrings is composed of a vertex and the other one of edges, we remove one vertex from the substrings composed of edges. Removing a vertex from a substring means that a number of edges is removed to isolate a vertex. This can lead to splitting the substring composed of edges into multiple substrings.

In **the last case** both substrings are composed of a vertex, which means we can just match them together.
By iterating through substrings pairs, we can at last completely match all edges of one NHC_s. If this is the case it means that $n > m$ or $m > n$. By removing all other edges, which are not mapped we end up with the number of removed edges.

5.3 Graph Modification Algorithm

The graph modification algorithm generalizes n vertices by changing the heterogeneous graph $G_{he} = (V, E, f_v, f_e)$ so that vertices $T = \{v_1, ..., v_n\} \subseteq V$ form an equivalence class.
The graph modification algorithm starts by changing G_{he} so that v_1 and v_2 are generalized and create an equivalence class $EQ = \{v_1, v_2\}$. After this the equivalence class is expanded by one additional vertex $v \in T$, where $v \notin EQ$. Next we change G_{he} so that v_1 and v are generalized. This process generally changes $G_{1hop}(v_1)$. To adjust to the new changes we have to update all other one-hop neighborhoods of vertices $v_i \in EQ$, where $v_i \neq v_1$, such that v_1 and v_i are generalized. We expand the equivalence class until each vertex of T has been added, such that $EQ = \{v_1, ..., v_n\}$. If $n \geq k$ we can call each vertex in EQ anonymized.

We note that this process might change the one-hop neighborhood of previously anonymized vertices. This means that a previously computed equivalence class could consist of less than k vertices having the same one-hop neighborhood.
On the one hand, this strengthens the anonymization instead of making it weaker. On the other hand, it increases the loss of information. Due to simplicity we say that once an equivalence has been computed all vertices in it are anonymized and the anonymization criteria is satisfied for these vertices.

We say $v, u \in T$ with $v \neq u$ are generalized, if the `age` attributes, the `based_near` attributes, the `current_project` structures and the `knows` structures are generalized so that $FNHC(G_{1hop}(v)) = FNHC(G_{1hop}(u))$. We generalize attributes by using modified versions of the algorithms explained in the similarity value section. In the next subsections we explain the generalization of attributes and structural components.

5.3.1 Generalization of Attributes

In this section we discuss the generalization of `age` and `based_near` attributes.
Generalization of attributes is the process of computing one output variable for two input variables.
The *distance algorithm* proposed in section 5.2.1 computes such a value as a byproduct. However, we made the assumption that both input variables are leaf vertices in the associated hierarchy tree $H = (V, E, l_v)$. We now extend the algorithm that it can handle arbitrary vertices. Nevertheless, the principle of the distance algorithm does not change.

We say that $l_v(a)$ and $l_v(b)$ are input variables, where $a, b \in V(H)$. Moreover, let a be associated with level x and b with level y. If $x > y$ or vice versa, the algorithm will first follow $|x - y|$ edges in H so that a and b represent vertices at the same level. If $a \neq b$ the algorithm proceeds like the distance algorithm explained in 5.2.1. The result is the generalization $l_v(a) = l_v(b)$.

Let $v \in G_{He}$ be an entity of interest (`EOI`). The vertex label representing the age of v is the vertex $o_1 \in V(G_{1hop}(v))$, where the edge $(v, o) \in E(G_{1hop}(v_1))$ is labeled `FOAF:age`. Similar let $f_v(o_2)$ be the age of `EOI` $u \in G_{He}$.
The age is generalized by replacing the labels $f_v(o_1)$ and $f_v(o_2)$ with the generalization of them as described above, where H is the age hierarchy tree. Moreover, we update $FNHC(G_{1hop}(v))$ and $FNHC(G_{1hop}(u))$.
The same is done for the living place except that the edge label is `FOAF:based_near` and H is Germanys geo-name tree.

5.3.2 Generalization of the Structure

In this section we discuss the generalization of structural elements.
Generalization does not include false information. Hence, we delete edges and not add edges as it is proposed by [ZP08]. Moreover, this idea is in compliance with the open world assumption. The open world assumption is that a missing statement can also be true, if it is not contained in the dataset.
The algorithm computing the generalization of the project neighborhood is similar to the algorithm computing the project similarity value. The algorithm computing the generalization additionally removes edges and associated full neighborhood codes are updated.
The generalization of the social network neighborhood is almost equal to the algorithm computing the knows similarity value. The difference is that every time an edge is deleted in $FNHC(G_{1hop}(v))$ or $FNHC(G_{1hop}(u))$ by the knows similarity algorithm, we also delete the associated edge in $G_{1hop}(v)$ or $G_{1hop}(u)$.
In addition to that, by actually deleting an edge it maybe effects other neighborhoods. Therefore, to keep the computation consistent we have to update all full neighborhood codes, which contain the deleted edge.
We assume the deleted edge labeled with `FOAF:knows` connects vertices $a, b \in V$. Moreover, let $G_{1hop}(a)$ be the one-hop neighborhood of a and $G_{1hop}(b)$ the one-hop neighborhood of b. Then we have to update all full neighborhood codes of $V(G_{1hop}(a)) \cup V(G_{1hop}(b))$.

5.4 Combined Anonymization Algorithm

In this section we describe how the previously explained approaches work together to anonymize a heterogeneous RDF graph $G_{he} = (V, E, f_v, f_e)$.
The inputs for this algorithm are: an equivalence class size parameter $k \in \mathbb{N}$, weighting coefficients $\alpha, \beta, \gamma, \delta \in \mathbb{R}_+$, anonymization percentage parameter $p \in [0, 1]$ and a heterogeneous RDF graph G_{he}. The following algorithm depicts the anonymization:

Algorithmus 5.2 Pseudo-code for the Anonymization algorithm: k-RDF-Anonymity

procedure $k\_RDF\_Anonymity(G_{he}, k, p, \alpha, \beta, \gamma, \delta)$
    $EOIs \leftarrow$ select all vertices with type FOAF:persons of G_{he} and randomly select p percent.
    for all $v \in EOIs$ **do**
        compute $FNHC(G_{1hop}(v))$
        remove identifiable attributes
        change the label of v such that the name in it is removed.
    end for
    while EOIs.$size > 2(k-1)$ **do**
        target $\leftarrow v \in EOIs$ with $G_{1hop}(v)$ such that there is no $G_{1hop}(v')$ with higher degree
        EOIs.$remove(v)$
        sim_list $\leftarrow$ empty list
        for all $u \in EOI's$ **do**
            sim_list $\leftarrow$ add $sim(FNHC(G_{1hop}(target)), FNHC(G_{1hop}(u)))$
        end for
        best_fitting_vertices $\leftarrow$ empty list
        for all $i \in \{1, ..., (k-1)\}$ **do**
            u $\leftarrow$ find minimum in sim_list and save associated vertex u
            best_fitting_vertices $\leftarrow u$
            sim_list.$remove(u)$
        end for
        $graph\_modification(best\_fitting\_vertices \cup target)$
        EOIs.$remove(best\_fitting\_vertices)$
    end while
    $graph\_modification(EOI's)$
    $return\ G_{he}$
end procedure

The algorithm first selects all vertices representing persons of G_{he} and then selects a subset of them based on the input p. This simulates that there are some individuals, who want all their private information released. All other vertices should be anonymized. We call these vertices entities of interest (EOI).

The next step is to compute for each vertex v in `EOI` a full neighborhood code as explained in section 5.1.
We also remove all identifiable attributes v has. Moreover, there might be some identifiable information in the resource `URI`. Therefore, we change the label of the vertex so that this kind of identifiable information is removed. Specifically we remove the name, because in our data generation approach we included the name in the `URI`.
After this we greedily select k vertices and anonymize them. We select the vertex $target \in$ `EOIs`, which has the biggest one-hop neighborhood, because this is the neighborhood, which could introduce the highest amount of information loss. We compute a similarity value between $target$ and all other vertices in `EOI` and select $k-1$ vertices with the smallest similarity value, because they reduce the information loss. We save them in a list called sim_list. We do this as explained in section 5.2. sim_list is most likely not the set introducing the smallest information loss, since correlations between vertices in sim_list are ignored due to complexity reasons. The corresponding decision problem is the substructure similarity search problem and proven to be NP-hard [ZP08]
The last step is to generalize the set of vertices $sim\_list \cup target$ with the graph_modification algorithm proposed in section 5.3. We repeat this until each vertex in `EOI` is anonymized.

5.5 Loss of Information

The loss of information is calculated by comparing the output of the anonymization algorithm $G_{he_{out}}$ with the input graph $G_{he_{in}}$. Let $G_{1hop}(v)$ be the one-hop neighborhood of $v \in V(G_{he_{in}})$ and $G_{1hop}(u)$ the one-hop neighborhood of $u \in V(G_{he_{out}})$, where $l_v(v) = l_v(u)$. Moreover, the type of v and u is `FOAF:Person` and u has been anonymized.
The Information loss is the similarity value between the $G_{1hop}(v)$ and $G_{1hop}(u)$. We sum all similarity values and divide the output by the number of vertices having type `FOAF:persons` in one graph.
However, we normalize the similarity value of the project similarity and the knows similarity so that they are not the number of removed edges.
The project similarity is divided by the number of vertices in the project neighborhood of $G_{1hop}(v)$.
The knows similarity is divided by the number of edges in the knows neighborhood of $G_{1hop}(v)$.
This means the project similarity and the knows similarity are values between 0 and 1. Therefore, the loss of information between $G_{he_{out}}$ and $G_{he_{in}}$ is a value between 0 and 4.
The final output is the average loss of information and also a value between 0 and 4.

5.6 k-RDF-Neighborhood Anonymity vs k–Neighborhood Anonymity

We introduced k-RDF-neighborhood anonymity, which is based on the approach proposed by Zhou et al. [ZP08]. They proposed $k-$neighborhood anonymity. Moreover, our approach additionally uses ideas of research presented in the related work section.
This section lists the core differences between our approach and the approach of Zhou et al.

- Our approach is designed to anonymize a *heterogeneous graph*. The approach of Zhou et al. is designed to anonymize a *homogeneous graph*.
- We deal with directed graphs and they deal with undirected graphs.
- We assume that each vertex has 1 identifiable attribute, 2 quasi identifiable attributes and is part of 2 networks. They assume that each vertex has 1 quasi identifiable attribute and is part of 1 network.
- Our approach is designed to do partial and full anonymization. Their approach is designed to do full anonymization.
- We delete edges and they add edges to satisfy the anonymization criteria. We note that deleting edges is more complicated, since the associated neighborhoods could be spitted. We do this because the idea of generalization is to not introduce false information. Therefore, to be consistent in that respect we delete edges. In addition it is in compliance with the open world assumption.
- We use a semantic hierarchy and a mathematical hierarchy to implement the generlaization. They only use a semantic hierarchy. In addition, our semantic hierarchy is due to the nature of the heterogeneous graph a part of it. The advantage is that in a global heterogeneous graph already defined hierarchies can be used to automate the process of generalization.
- We assume that each vertex is uniquely identified by its vertex label. They assume that the vertex label is the quasi identifiable attribute. Their assumption reduces the complexity.

5.7 Summary of the anonymization approach

In this chapter, we have introduced k-RDF-Anonymity, a new anonymization algorithm that can be used to anonymize a heterogeneous RDF graph containing personal identifiable information. The algorithm is a modification of the algorithm proposed by Zhou et al. [ZP08] and additionally uses ideas presented in the related work section.

The difference between our algorithm and the one proposed by Zhou et al. is that our algorithm can deal with heterogeneous RDF graphs. This means, it can compute different anonymizations reducing the loss of information in different parts of the input graph, while still satisfying the anonymization criteria. The anonymization criteria is that at least k vertices representing persons have the same one-hop neighborhood so that they form an equivalence class.

We note that the anonymization algorithm might change previously computed equivalence classes so that the anonymization criteria is not satisfied anymore. On the one hand, this process strengthens the anonymization and does not make it weaker. On the other hand it introduces more loss of information. In order to simplify things, we say that the anonymization criteria is still satisfied. Yet the algorithm can be adjusted to avoid this.

The one-hop neighborhood is represented by a unique string we call full neighborhood code ($FNHC$). This string is composed of three parts. The attribute part, the human collaboration part and the social network part. The computation of the social network part is the most complex one and the bottleneck of this algorithm. We compute such a string, because we otherwise would have to conduct a lot of isomorphism tests. By computing this string we can simply compare strings. If two strings of two graphs are identical in the sense of the defined linear order, this means the associated graphs are isomorphic. Moreover, it simplifies the calculation of a value indicating the difference between two one-hop neighborhoods.
The similarity value between two one-hop neighborhood graphs is the sum of the similarity of each substring of the full neighborhood code. We call the substring neighborhood component code (NHC). As stated in section 5.2 $sim(FNHC(G_{1hop}(v)), FNHC(G_{1hop}(u)))$ is the weighted sum of four different similarity values. Thereby we control which part is important and influences the anonymization.

The algorithm starts by computing for each vertex representing a person a full neighborhood code. It then selects all Entities of interest (EOIs). All vertices in the EOI set are anonymized. For our evaluation in section 8 we randomly select 0.95 percent of all persons as $EOIs$. However, the selection criteria would be based e.g on resource type depending on the domain of the dataset.
Then the algorithm greedily selects one vertex of the EOI set and computes similarity values between its one-hop neighborhood and every other one-hop neighborhood associated with a vertex in the EOI set. Afterwards the algorithm selects at least $k-1$ similar neighborhoods based on the calculated similarity values and generalizes their one-hop neighborhood, such that they cannot be distinguished. The algorithm repeats this process until all vertices in the EOI set have been anonymized, such that at least k vertices are grouped into an equivalence class.

6 De-anonymization Approach: Guessing Based Neighborhood Attack

This chapter describes our de-anonymization approach. It can be classified as a guessing based neighborhood attack.
As for the anonymization, in general all ideas presented in this section are applicable to any heterogeneous `RDF` graph. However, this approach is designed to de-anonymize an anonymized generated graph G_{he} with the data generation approach we described in chapter 4.
The objective for this attack is to re-identify anonymized `FOAF:persons` $V_{ano} \subset V(G_{ano})$ in the anonymized heterogeneous `RDF` graph G_{ano}.
To achieve this, background knowledge is essential. In chapter 4 we described that the background knowledge graph G_{bk} is a subgraph of G_{he}.
Each `FOAF:person` in G_{bk} is identified, but the attribute neighborhood, human collaboration neighborhood and social network neighborhood may be incomplete. Moreover, we assume that G_{bk} does not include false information.
The idea is to find for each vertex $v_{ano} \in V_{ano}$ a set of possible candidates $V_{pc} \subseteq V_{bk}$, where $V_{bk} \subset V(G_{bk})$. V_{pc} is a set of persons. If $|V_{pc}(v_{ano})| = 1$ we say we have perfectly re-identified v_{ano}. If the set is bigger than 1 we guess.
This approach has two input parameters. A probability parameter $p \in [0, 1]$ and a threshold parameter $t \in [0, 4]$.

The de-anonymization approach consists of two algorithms:
The calculation of perfect matches for v_{ano} and the calculation of similarity values.
The similarity value computation is equal to the similarity calculation for the anonymization approach.
Based on these two approaches we developed two de-anonymization algorithms, which essentially are doing the same but should be different in their success rate.
In the following section we explain the calculation of perfect matches. Then we combine both algorithms, such that we obtain two de-anonmymization algorithms.

6.1 Perfect Match Algorithm

The perfect match algorithm has as input an anonymized vertex v_{ano} and a background knowledge graph G_{bk}. The objective for this algorithm is to calculate a set of possible candidates $V_{pc} \subset V_{bk}$. We compute this set by checking for each vertex $v_{bk} \in V_{bk}$, if $G_{1hop}(v_{ano})$ is a subgraph of $G_{1hop}(v_{bk}))$. If this is true we add vertex v_{bk} to the set of possible candidates V_{pc}. We refer to this set as the set of perfect matches.
Let $G_1 \sqsubseteq G_2$ denote that G_1 is subgraph of G_2.
We say $G_{1hop}(v_{ano}) \sqsubseteq G_{1hop}(v_{bk}))$, if the following conditions are satisfied:

- $G_a(v_{ano}) \sqsubseteq G_a(v_{bk})$, where $G_a(v)$ is the attribute neighborhood
- $G_{hc}(v_{ano}) \sqsubseteq G_{hc}(v_{bk})$, where $G_{hc}(v)$ is the human collaboration neighborhood
- $G_s(v_{ano}) \sqsubseteq G_s(v_{bk})$, where $G_s(v)$ is the social network neighborhood

To calculate, if $G_a(v_{ano}) \sqsubseteq G_a(v_{bk})$ we use the distance algorithm presented in section 5.3.1. We use it to calculate the generalization of the age attribute and the based_near attribute. However, we do not modify any of the graphs. If both generalizations are equal to the labels of the associated vertices in $G_a(v_{ano})$ we say $G_a(v_{ano}) \sqsubseteq G_a(v_{bk})$.
To calculate if $G_{hc}(v_{ano}) \sqsubseteq G_{hc}(v_{bk})$ we check whether $V(G_{hc}(v_{ano})) \subset V(G_{hc}(v_{bk})))$.
Lastly to check if $G_s(v_{ano}) \sqsubseteq G_s(v_{bk})$, we compute the knows similarity of $G_s(v_{ano})$ and $G_s(v_{bk})$. We are interested in the computational process and not in the value. If the process does not delete an edge in $NHC_s(v_{ano})$ we say $G_s(v_{ano}) \sqsubseteq G_s(v_{bk})$. This means we have found a subgraph in $G_s(v_{bk})$, which is isomorphic to $G_s(v_{bk})$.

Figure 6.1 presents an example of a perfect match, where each neighborhood of $G_{1hop}(v_{ano})$ is a subgraph of each neighborhood in $G_a(v_{bk})$. Figure 6.2 presents an example of not a perfect match, where each neighborhood of $G_{1hop}(v_{ano})$ is not a subgraph of each neighborhood in $G_a(v_{bk})$.

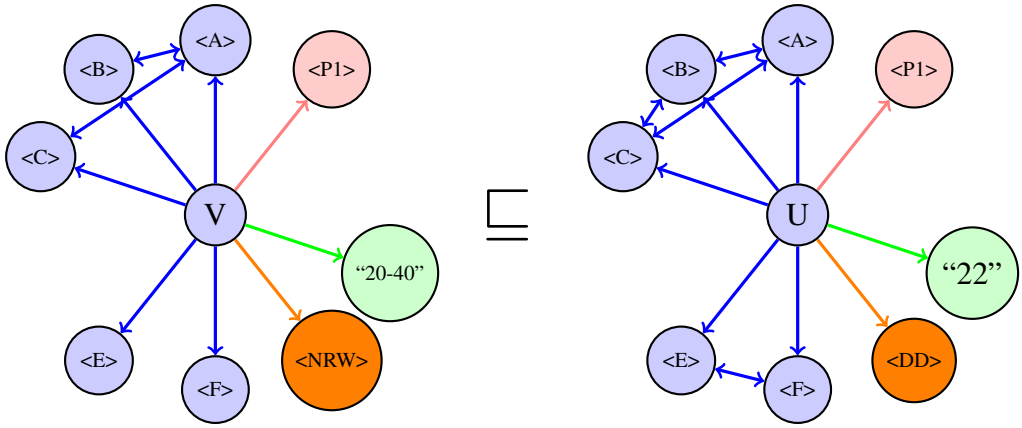

Figure 6.1: This figure presents an example of a perfect match, where each neighborhood of the left graph G_{ano} is a subgraph of the associated neighborhood of the right graph G_{bk}. The blue neighborhood represents the social network neighborhood, the red neighborhood the human collaboration neighborhood and the green and orange neighborhood form the attribute neighborhood.

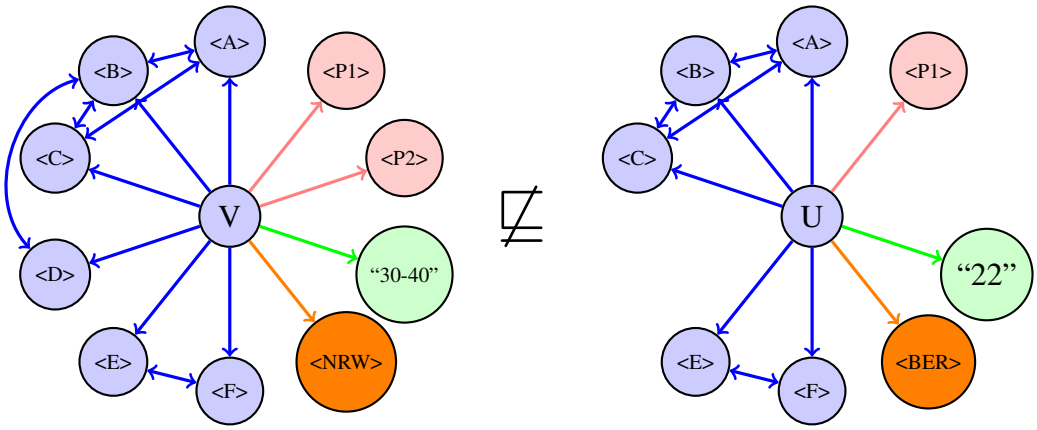

Figure 6.2: This figure presents an example of **not** a perfect match, where each neighborhood of the left graph G_{ano} is **not** a subgraph of the associated neighborhood in the right graph G_{bk}. The blue neighborhood represents the social network neighborhood, the red neighborhood the human collaboration neighborhood and the green and orange neighborhood form the attribute neighborhood.

6.2 Combined De-anonymization Algorithm

We developed two de-anonymization attacks, which are called de-anonymization-min and de-anonymization-value. We start by presenting the de-anonymization-min algorithm which is depicted by algorithmus 6.1.
Based on this we later on explain the second algorithm, because it is almost equal to the de-anonymization-min algorithm. The second algorithm is equal to the first one except for the red line (18). This means the de-anonymization algorithms compute different candidate sets.

Algorithmus 6.1 Pseudo-code for the De-anonymization algorithm

1: **procedure** $De\_anonymization\_min(G_{ano}, G_{bk}, p)$
2: $ano\_list \leftarrow$ list of anonymized vertices in G_{ano} having type `FOAF:person`
3: $bk\_list \leftarrow$ list of vertices in G_{bk} having type `FOAF:person`
4: **for all** $v \in ano\_list$ **do**
5: compute $FNHC(G_{1hop}(v))$
6: **end for**
7: **for all** $v \in bk\_list$ **do**
8: compute $FNHC(G_{1hop}(v))$
9: **end for**
10: $success\_rate \leftarrow 0$
11: **for all** $v \in ano\_list$ **do**
12: possible_candidate_set $\leftarrow perfect\_matches(v, G_{bk})$
13: sim_list $\leftarrow$ empty_list
14: **for all** $u \in possible\_candidate\_set$ **do**
15: sim_list $\leftarrow$ add normalized $similarity(FNHC(G_{1hop}(v)), FNHC(G_{1hop}(u)))$
16: **end for**
17: min $\leftarrow$ search minimum value in sim_list
18: $candidate\_set \leftarrow$ select all vertices in sim_list associated with min
19: // success rate calculation
20: **for all** $v_{bk} \in candidate\_set$ **do**
21: **if** $v.id == v_{bk}.id$ **then**
22: **if** $\frac{1}{candidate\_set.length} \geq p$ **then**
23: $success\_rate \leftarrow success\_rate + \frac{1}{candidate\_set.length}$
24: **end if**
25: **end if**
26: **end for**
27: **end for**
28: $return \frac{success\_rate}{ano\_list.size}$
29: **end procedure**

The de-anonymization algorithms are computing a success rate. Therefore, they technically do not reconstruct the anonymized input graph, which could be used for data mining. We calculate the expected success rate, since the objective of the de-anonymization in this thesis is to test how good the previously computed anonymization is.

The de-anonymization-min algorithm gets as input an anonymized graph G_{ano}, a background knowledge graph G_{bk} and a percentage parameter p.
The algorithm begins by selecting all vertices in the anonymized graph having type `FOAF:person` and saves them in ano_{list}. Likewise the background knowledge list bk_{list} is computed.
For each vertex in those lists the associated full neighborhood code is computed, as explained in section 5.1.
Afterwards, for each vertex $v \in ano_{list}$ all perfect matches are computed, by using the algorithm explained in section 6.1.
Then for each perfect match we compute the normalized similarity value as explained in section 5.5, which we used to calculate the information loss. We store the output in the similarity list.
Now we calculate the minimum in the similarity list and select all vertices having this minimum as similarity value. We save those in the candidate set.
We proceed by checking if the de-anonymization was successful. We say it was successful, if the correct identity of v is in the candidate set.
We check this by selecting the id of anonymized vertex v in the candidate set. During the anonymization process we changed labels of vertices by removing the name. However, as explained in the data generation chapter, we also added an id to the label for the purpose of later checking, if we successfully de-anonymized.
This means, if we found the id in the set, we say we successfully de-anonymized. We note that successful de-anonymization is not equal to a successful re-identification. Successful re-identification is always the case, when we have a perfect successful de-anonymization. This means the candidate set is only composed of one vertex and we add a 1 to the success rate.
If the candidate set is bigger than one, we have to guess. Therefore the chance to successful re-identification is $\frac{1}{candidate\_set.size}$. If $\frac{1}{candidate\_set.size} \leq p$ we say that we have a chance to correctly re-identify and thereby add $\frac{1}{candidate\_set.size}$ to the success rate.
After having finished with each anonymized vertex we normalize the success rate and end up with a value between 1 and 0.

The de-anonymization-value algorithm works exactly the same. However, we calculate a different candidate set. For this the algorithm has an additional input. This value is the previously mentioned threshold t.
More specifically we increase the candidate set expecting to successfully de-anonymize more vertices and end up with a higher re-identification success rate. The difference to the de-anonymization-min algorithm is that we select all vertices in the similarity list having a value between min and $(min + t)$.

6.3 Summary of the de-anonymization approach

In this chapter, we have introduced de-anonymization-min and de-anonymization value, de-anonymization algorithms, which can be used to de-anonymize an anonymized heterogeneous RDF graph with the help of background knowledge. The background knowledge is also a heterogeneous graph.
The de-anonymization algorithms can be classified as guessing based neighborhood attacks. They map each vertex from the anonymized graph to a set of possible vertices in the background knowledge graph. If the set of possible vertices contains only one vertex, we say it is a perfect re-identification. Otherwise the algorithm guesses a vertex in the possible vertex set.
The de-anonymization-min and de-anonymization value algorithm are the same algorithms except for one computational step.
The algorithm starts by computing a full neighborhood code for each vertex in the background knowledge and each vertex in the anonymized graph
Then the algorithm computes a set of perfect matches for each anonymized vertex in the anonymized data. The set of perfect matches consists of vertices in the background knowledge, where each associated full neighborhood code is a superset of the full neighborhood code of the anonymized vertex.
Afterwards, the algorithm computes similarity values for all perfect matches.
The de-anonymization min algorithm selects the minimum and guesses, if there are multiple minimums.
The de-anonymization value algorithm selects the minimum and adds a threshold parameter. This has generally the effect that the candidate set gets larger and therefore the success rates drops. However, this algorithm might be able to de-anonymize more vertices, which compensates the success rate drop.

7 Implementation

In this chapter we describe what we used to implement the presented algorithms.
Moreover, we give a broad overview how we organized the source code, such that a person which wants to review the source code, knows where to look for certain algorithms.

7.1 General Infrastructure

We implemented everything in Java.
We chose Java, because it is the most popular programming language in developing semantic web applications. This was shown in a study conducted by Heitman et al. [HCHD12]. Moreover, Heitman et al. [HCHD12] claim that the two most mature and popular `RDF` libraries are Jena and Sesame. We chose Apache Jena, because it is an open source Java framework and we modeled our heterogeneous graph as an `RDF` graph. In addition to that, Jena supports read and write operations.
We used Eclipse as our development environment.
Jena has a special class called *model*. This *model* represents a graph we defined as a heterogeneous RFD graph. Jena supports reading in already existing datasets. This is done with a class called *RDFDataMgr* and is used to add `RDF` data to an already existing *model*. Jena also supports different `RDF` formats. Two of these formats are *RDF/XML* and *turtle*.
In the data generation chapter we talked about `RDF` documents. These documents have the specific format RDF/XML. Therefore we used the *RDFDataMgr* to read in all *RDF/XML* documents we downloaded. To add a triple to a *model*, Jena implements many different ways. The method we used the most is called *add statement*.
Jena specifically restricts modifying existing statements associated with a *model*. Therefore, to modify a statement we have to delete the statement we want to modify and add a new statement, which represents the modified version.
To evaluate the results we generated in our experiment, which is explained in the following chapter, we used the *Apache Commons Mathematics Library*. More specifically we used the class DescriptiveStatistics to calculate the mean, average and the standard deviation of a list of numbers. The experiment results also include saved models for generated graphs and anonymized graphs. We saved them in the data format turtle.

7.2 Code organization

We implemented 23 classes in 7 packages. We explain what each package is for and briefly state what each class does. The 7 packages are called default, setup, anonymization, de-anonymization, helping_classes, R_mat and Vocabularies.
The **default package** has three classes. The first one, the *Test*, implements the main method and is called to execute the code. The second class is *Experiment*. It implements the experiment, which we describe in the next chapter. The third class *Graph_compare* implements the calculation of the information loss as explained in chapter 5.5.
The setup package includes 5 classes and the **R_mat** 1 class. In general these two packages are used to implement chapter 4. The setup package includes a class called *Data_Generator*. This class is used to create a model and uses the classes *Person_Generator*, *Project_Generator*, *Structure_Generator*, *Attribute_Generator* to generate the input data. The class *RMat* is used in the Structure Generator to generate the social network structure.
The anonymization package has 5 classes. The class *Full_Neighborhood* calculates the one-hop neighborhood of a vertex. To achieve this, the class uses the classes *Neighborhood* and *Neighborhood_Component*. The first class represents either an attribute neighborhood, a human collaboration neighborhood or a social network neighborhood. The class Neighborhood component implements chapters 5.1. The superclass of *Anonymizer* is *Neighborhood_Anonymizer*. Neighborhood_Anonymizer implements 5.2,5.3 and 5.4. **The de_anonymization package** has 3 classes. *NH_attack* is the superclass of *neighborhood_attack* and they implement together algorithms of chapter 6. The class *Model_changer* uses methods from the setup package to generate background knowledge graphs. **The helping class package** is a collection of classes used for formating data and saving calculated data to the hard drive. Finally **the Vocabularies package** defines resources and properties.

8 Experiment

In this chapter we describe the experiment, which executes the implementation on different input parameters.
We executed the experiment on a computer with 8GB internal memory and an intel core i7 processor.
The first section describes the experiment and what parameter values we used.
In the second section we present the results of the experiment and discuss interpretations of them.

8.1 Experiment Structure

In general, the experiment is designed to compute different anonymizations and to measure the information loss of each one. Additionally, to test how vulnerable the resulting anonymizations are, we de-anonymized each anonymization with different background knowledge graphs. For each de-anonymization we measured the success rate.
We divided the experiment into 3 phases: the data generation phase, the anonymization phase and the de-anonymization phase.

8.1.1 Data Generation Phase

In the data generation phase we generated 10 different graphs for the anonymization phase.
For each graph we generated $n = 2^8 = 256$ person resources and $m = 3 * 256 = 768$ `FOAF:knows` edges.
The reason why these parameters are so small is that the full neighborhood code computation algorithm computes so many depth first search trees for high degree resources, such that our program runs out of allocated memory, if we increase these parameters.
This problem could be solved by running the program on a server or by changing the implementation so that the minimum code is not computed by saving all depth first search trees. Moreover, the calculation of minimum depth first search trees can be parallelized.
In addition we generated $n_2 = 15$ different projects and associated a person with at least

$min = 3$ projects and a maximum of $max = 7$ projects. We chose the recursion depth $rd = 5$, such that at least $2^3 = 8$ people are associated with the same living place.
We saved each graph as a turtle file to the hard-drive. Furthermore, we computed for each vertex its number of edges labeled `FOAF:knows`.

8.1.2 Anonymization phase

In the anonymization phase, we computed for each graph $7 * 5 = 35$ different anonymizations with the algorithm $k - \texttt{RDF} - anonymity$ presented in 5.4.
The algorithm has six different input parameters beside the input graph:

- p: simulates that in each anonymization there are some individuals, which want there entire personal information to be released
- k: indicates how strong the anonymization is
- α: weighting parameter for sim_{age}
- β: weighting parameter for sim_{based_near}
- γ: weighting parameter for $sim_{current\_project}$
- δ: weighting parameter for sim_{knows}

We chose the parameter p to be 0.95 for all the anonymizations. The result is that $0.95 * 256 = 243$ individuals are anonymized and $0.05 * 256 = 13$ not. In addition to that, for each anonymization associated with the same graph we chose the same 243 resources with type `FOAF:person`.

To distinguish anonymizations we use the $5-$tuple $(k, \alpha, \beta, \gamma, \delta)$. Each $5-$tuple represents a unique anonymization. The combination of $\alpha, \beta, \gamma, \delta$ defines on which part or parts the anonymization algorithm is concentrated on and the parameter k is used to adjust the strength of the anonymization. We use 5 different strengths, where $k \in \{3, 4, 5, 6, 7\}$ and seven distinct combinations of $\alpha, \beta, \gamma, \delta$, to obtain 35 different $5-$tuples.
In the following we present seven distinct combinations of $\alpha, \beta, \gamma, \delta$ and their meaning.

1. $(k, 1, 0, 0, 0)$: computes an anonymization solely based on the age attribute.
2. $(k, 0, 1, 0, 0)$: computes an anonymization solely based on the living place attribute.
3. $(k, 0, 0, 1, 0)$: computes an anonymization solely based on the human collaboration network.
4. $(k, 0, 0, 0, 1)$: computes an anonymization solely based on the social network.

5. $(k, 1, 1, 1, 1)$: computes an anonymization mostly based on structural components. We note that γ and β are weighting coefficients for structural components and increase of decrease the number of edges removed. α and β are used to increase or decrease the weight of attribute similarities, which is a value between 0 and 1. Therefore, if we choose for each weighting coefficient the same value, structural components have generally a bigger influence than attribute components.

6. $(k, 10, 10, 1, 1)$: computes an anonymization equally based on structural components and attribute components. We calculated the average number of removed edges by conducting anonymizations with the first four anonymization values. The average number of removed edges was approximately 10.

7. $(k, 10, 10, 0.5, 0.5)$ computes an anonymization mostly based on attribute components.

The anonymization experiment procedure is to compute for each $5-$tuple an anonymization with $k - \texttt{RDF} - anonymity$ and save the output.
The output metric is the loss of information, which is a value between 0 and 4. In addition we saved the loss of information for each neighborhood, which is a value between 0 and 1. On top of that we calculated the number of edges labeled `FOAF:knows` for each vertex.
Therefore, we calculated 350 anonymizations in total. However, there are only 35 different combinations of input parameters, if the input graph is excluded.
This means we computed the same anonymization on 10 different graphs. To analyse the data, we calculated the average, mean and standard deviation for each distinct combination of input parameters.

8.1.3 De-anonymization phase

In the de-anonymization phase we computed for each anonymization $9 * 8 = 72$ de-anonymizations. We call G_{in} the graph which was anonymized.
8 de-anonymizations were computed with the de-anonymization-min algorithm, while the rest was computed with the de-anonymization-value algorithm as explained in 6.2.
In this phase, we first generated 8 different background knowledge graphs. We call the first graph the perfect background knowledge and it is G_{in}. The other 7 graphs are incomplete versions of the perfect background knowledge graph. To obtain them, we randomly removed $10\%, 20\%, 30\%, 40\%, 50\%, 60\%$ and 70% percent of quasi identifiable attributes in the perfect background knowledge.
For all de-anonymizations we chose p to be 0.1. This means that we try to guess, if the chance to re-identify a `FOAF:person` correctly is at least 10%.
The de-anonymization-min algorithm has no additional input parameters.
However, the de-anonymization-value algorithm has one additional input parameter called

threshold. For the threshold we chose the parameters $0.1, 0.2, 0.3, 0.4, 0.5, 0.8, 1.0, 1.5$.
The de-anonymization experiment procedure is to compute the success rate with the de-anonymization min algorithm for each generated background knowledge graph and the anonymized graph. Moreover, the algorithm computes a success rate with the de-anonymization value algorithm for each additional threshold t.

The output of this procedure is the success rate of each de-anonymization, which is a value between 0 and 1.
Since we calculated 350 anonymizations we calculated 25200 de-anonymizations in total. We also calculated the average, mean and standard deviation for the set of success rates associated with the same anonymization excluding the input graphs.

8.2 Experiment Evaluation

In this section we present the experiment data and our interpretations.
The first thing we noticed was that the difference between the average and the mean of almost all calculated values is very small. Therefore, we restrict to the average.
We first present the results for the loss of information for each of the 35 distinct anonymization combinations.
After this we present only the output for the de-anonyization-min algorithm. The reason for this is stated in the associated subsection.

8.2.1 Evaluation of the Anonymization Phase

Table 8.1 presents the output of the anonymization phase. Each column except the last column, represents a different average loss of information metric as explained in the previous section. If a value is 1, this means all information is lost, whereas 0 means no information is lost.
The table is divided into seven sections. Each section presents the results of the anonymization with the same weighting parameters. However, the parameter k changes in each row of a section. All 35 $5-$tuples are shown in the first column.
The $2nd$ and $3rd$ column show the average loss of information in the attribute neighborhood. Column 4 shows the average loss of information in the human collaboration neighborhood and column 5 shows the average loss of information in the social network neighborhood.
The last column is divided into two sub-columns. The left sub-column shows the average of the combined loss of information. We note that this value may be different from the real sum of the presented values, because the combined loss of information value was calculated with the true values and not with the rounded ones. The right sub-column shows the average standard deviation, since one row represents the average of 10 anonymizations.

	Age	**Based near**	**Current Project**	**Knows**	**Sum**	
$(k, \alpha, \beta, \gamma, \delta)$	**avg**	**avg**	**avg**	**avg**	**avg**	**sd**
$(3, 1, 0, 0, 0)$	0.06	0.81	0.75	0.71	2.36	0.07
$(4, 1, 0, 0, 0)$	0.1	0.9	0.89	0.81	2.7	0.10
$(5, 1, 0, 0, 0)$	0.14	0.96	0.94	0.89	2.36	0.1
$(6, 1, 0, 0, 0)$	0.16	0.98	0.96	0.92	3.04	0.12
$(7, 1, 0, 0, 0)$	0.19	1.0	0.98	0.94	3.11	0.12
$(3, 0, 1, 0, 0)$	0.61	0.05	0.74	0.7	2.1	0.06
$(4, 0, 1, 0, 0)$	0.72	0.05	0.87	0.82	2.46	0.05
$(5, 0, 1, 0, 0)$	0.77	0.09	0.94	0.86	2.67	0.05
$(6, 0, 1, 0, 0)$	0.81	0.10	0.97	0.91	2.78	0.07
$(7, 0, 1, 0, 0)$	0.85	0.13	0.98	0.94	2.89	0.06
$(3, 0, 0, 1, 0)$	0.60	0.81	0.33	0.72	2.44	0.10
$(4, 0, 0, 1, 0)$	0.71	0.93	0.49	0.84	2.93	0.10
$(5, 0, 0, 1, 0)$	0.77	0.97	0.61	0.88	3.25	0.10
$(6, 0, 0, 1, 0)$	0.82	0.98	0.72	0.92	3.42	0.13
$(7, 0, 0, 1, 0)$	0.84	0.99	0.79	0.95	3.57	0.12
$(3, 0, 0, 0, 1)$	0.62	0.77	0.75	0.29	2.42	0.10
$(4, 0, 0, 0, 1)$	0.73	0.88	0.87	0.42	2.86	0.10
$(5, 0, 0, 0, 1)$	0.79	0.93	0.94	0.52	3.20	0.11
$(6, 0, 0, 0, 1)$	0.81	0.95	0.97	0.57	3.27	0.14
$(7, 0, 0, 0, 1)$	0.86	0.97	0.98	0.64	3.47	0.15
$(3, 1, 1, 1, 1)$	0.56	0.64	0.42	0.47	2.02	0.13
$(4, 1, 1, 1, 1)$	0.66	0.77	0.60	0.54	2.58	0.13
$(5, 1, 1, 1, 1)$	0.72	0.83	0.73	0.63	2.94	0.14
$(6, 1, 1, 1, 1)$	0.78	0.86	0.81	0.69	3.13	0.12
$(7, 1, 1, 1, 1)$	0.81	0.89	0.86	0.75	3.31	0.11
$(3, 10, 10, 0.5, 0.5)$	0.24	0.01	0.64	0.58	1.58	0.15
$(4, 10, 10, 0.5, 0.5)$	0.32	0.15	0.79	0.68	1.97	0.18
$(5, 10, 10, 0.5, 0.5)$	0.36	0.21	0.88	0.79	2.21	0.19
$(6, 10, 10, 0.5, 0.5)$	0.38	0.21	0.93	0.84	2.37	0.22
$(7, 10, 10, 0.5, 0.5)$	0.38	0.26	0.95	0.87	2.52	0.23
$(3, 10, 10, 1, 1)$	0.30	0.16	0.57	0.53	1.58	0.15
$(4, 10, 10, 1, 1)$	0.38	0.23	0.75	0.63	2.01	0.18
$(5, 10, 10, 1, 1)$	0.42	0.28	0.83	0.71	2.27	0.20
$(6, 10, 10, 1, 1)$	0.49	0.26	0.90	0.79	2.52	0.24
$(7, 10, 10, 1, 1)$	0.53	0.32	0.92	0.82	2.60	0.22

Table 8.1: Experiment data of the anonymization phase.

The parameter k is an indicator how strong the anonymization is. Therefore, a section gives an overview how the anonymization output based on one combination of weighting parameters evolves, if the anonymization is strengthened by increasing the paramter k.

The first four sections show anonymizations, which solely concentrate on one part of the graph. More specifically, section 1 concentrates on the age attribute in the attribute neighborhood. Section 2 concentrates on the based_near attribute in the attribute neighborhood. Section 3 concentrates on the human collaboration neighborhood, while section 4 concentrates on the social network neighborhood. We stated previously that we expect, if the anonymization is concentrated solely on one part, the loss of information should be small in that part and high in every other part.
For example, the first section solely concentrates on the age attribute. Therefore, we expect the loss of information to be small in the age column and high in the next 3 columns. As the data proves this is exactly what happens and what we expected. Section 2,3 and 4 show similar results.
We note that section 4 shows the anonymizaton, which solely focuses on the social network neighborhood. This is the anonymization type of most homogeneous social network anonymization approaches.

If the algorithm is concentrated on the attribute neighborhood the loss of information drops significantly more, as if it is concentrated on one of the structural neighborhoods.
We hypothesize that this is the case, because we generated a small network due to complexity issues. Therefore, if the graph had more vertices and a more dense structural part there would be more similar structural neighborhoods, and we expect the loss of information in these neighborhoods to drop.
Nevertheless, the data proves that the loss of information is low in a graph part, if the anonymization is solely concentrated on this part and high in every other part.

Section 5 of the table shows the anonymization, which mostly concentrates on the structural part. However, the anonymization is influenced by the attribute part, too. By comparing section 5 with section 4, which concentrates solely on the social network part, we observe that the loss of information in the attribute part of section 5 is not as high as in section 4. Moreover, in section 5 we concentrate on both structural parts equally. On top, we observe that in the current_project column we have less loss of information as in section 4. However, the loss of information in the knows column in section 5 is higher than in section 4. This is expected, since we do not focus solely on the social network neighborhood.
This mostly results in less combined information loss, which is shown in the left sub-column of the sum column.
Section 6 and 7 always result in less combined information loss, if compared to an anonymization, which concentrates solely on one part.

Therefore, to have good data utility we realize that anonymization should not be concentrated solely on one part.
To the best of our knowledge we are the first ones to make this statement and present experiment data indicating the truthfulness of this statement.

Furthermore, we observe that the loss of information increases in a linear way to 1 with rising k.
We note that it is difficult to say which anonymization is the best one, since it is a matter of priority. However, if we use the combined information loss as a metric to determine the best anonymization, this is to concentrate mostly on the attribute part.

8.2.2 Evaluation of the De-anonymization Phase

Table 8.2 shows some of the results we obtained in the de-anonymization phase.
We present only the results of the de-anonymization-min algorithm. The reason is that the de-anonymization value algorithm mostly yields worse results than the de-anonymization min algorithm.
Therefore, the hope to end up with a higher re-identification rate by using the de-anonymization value algorithm turned out to be false.

Table 8.2 shows $35 * 8 = 280$ success rate values. Each cell in table 8.2 represents the average percentage of correct re-identified persons by the de-anonymization min algorithm. We call this percentage the success rate.
For each anonymization (row) we calculated 8 de-anonymizations. The percentage associated with each column shows how much information has been removed from the perfect background knowledge used by the de-anonymization algorithm.

$(k, \alpha, \beta, \gamma, \delta)$	0%	10%	20%	30%	40%	50%	60%	70%
$(3, 1, 0, 0, 0)$	0.18	0.17	0.15	0.13	0.13	0.09	0.06	0.04
$(4, 1, 0, 0, 0)$	0.10	0.11	0.10	0.09	0.08	0.07	0.049	0.04
$(5, 1, 0, 0, 0)$	0.07	0.07	0.06	0.05	0.06	0.05	0.04	0.03
$(6, 1, 0, 0, 0)$	0.04	0.04	0.04	0.03	0.04	0.03	0.02	0.03
$(7, 1, 0, 0, 0)$	0.02	0.02	0.03	0.03	0.02	0.03	0.03	0.02
$(3, 0, 1, 0, 0)$	0.10	0.10	0.09	0.07	0.06	0.05	0.03	0.02
$(4, 0, 1, 0, 0)$	0.05	0.05	0.05	0.04	0.04	0.05	0.03	0.03
$(5, 0, 1, 0, 0)$	0.04	0.04	0.04	0.03	0.03	0.03	0.02	0.02
$(6, 0, 1, 0, 0)$	0.03	0.03	0.03	0.03	0.02	0.03	0.02	0.02
$(7, 0, 1, 0, 0)$	0.02	0.03	0.03	0.02	0.03	0.02	0.02	0.02
$(3, 0, 0, 1, 0)$	0.07	0.06	0.06	0.04	0.04	0.03	0.02	0.00
$(4, 0, 0, 1, 0)$	0.02	0.03	0.03	0.03	0.02	0.02	0.02	0.01
$(5, 0, 0, 1, 0)$	0.04	0.04	0.04	0.03	0.03	0.03	0.02	0.02
$(6, 0, 0, 1, 0)$	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.00
$(7, 0, 0, 1, 0)$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$(3, 0, 0, 0, 1)$	0.10	0.08	0.04	0.02	0.01	0.00	0.00	0.00
$(4, 0, 0, 0, 1)$	0.03	0.03	0.02	0.01	0.01	0.00	0.00	0.00
$(5, 0, 0, 0, 1)$	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00
$(6, 0, 0, 0, 1)$	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00
$(7, 0, 0, 0, 1)$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$(3, 1, 1, 1, 1)$	0.12	0.12	0.09	0.06	0.04	0.02	0.01	0.01
$(4, 1, 1, 1, 1)$	0.05	0.06	0.05	0.03	0.03	0.02	0.01	0.00
$(5, 1, 1, 1, 1)$	0.02	0.03	0.03	0.02	0.02	0.02	0.00	0.00
$(6, 1, 1, 1, 1)$	0.01	0.01	0.1	0.1	0.01	0.00	0.00	0.00
$(7, 1, 1, 1, 1)$	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00
$(3, 10, 10, 0.5, 0.5)$	0.19	0.17	0.17	0.13	0.10	0.07	0.04	0.02
$(4, 10, 10, 0.5, 0.5)$	0.09	0.10	0.10	0.09	0.07	0.05	0.04	0.3
$(5, 10, 10, 0.5, 0.5)$	0.05	0.06	0.06	0.06	0.05	0.05	0.03	0.02
$(6, 10, 10, 0.5, 0.5)$	0.03	0.04	0.4	0.4	0.04	0.04	0.03	0.02
$(7, 10, 10, 0.5, 0.5)$	0.02	0.03	0.03	0.03	0.04	0.02	0.02	0.02
$(3, 10, 10, 1, 1)$	0.16	0.16	0.15	0.11	0.08	0.04	0.03	0.01
$(4, 10, 10, 1, 1)$	0.08	0.09	0.09	0.08	0.06	0.04	0.03	0.01
$(5, 10, 10, 1, 1)$	0.04	0.05	0.6	0.5	0.05	0.04	0.03	0.01
$(6, 10, 10, 1, 1)$	0.02	0.03	0.03	0.03	0.03	0.03	0.02	0.01
$(7, 10, 10, 1, 1)$	0.02	0.02	0.03	0.03	0.03	0.03	0.02	0.02

Table 8.2: Experiment data of the de-anonymization phase

We can observe that with rising k and worse background knowledge the success rate drops significantly.
The structural part is much more effected by the anonymization, because we generated a small social network. Therefore, we see that the de-anonymization algorithm is not very useful, if the anonymization is solely concentrated on the structural part.
As for the loss of information we hypothesize, if the social network is denser the loss of information will be lower and therefore the success rate will increase.
The highest success rates we achieved occurred by de-anonymizing anonymizations, which were concentrated on the attribute neighborhood, and by using the perfect background knowledge.
The highest success rate we achieved is 18%.
The success rate is usually cut in half, if 50 percent of the perfect background data has been deleted to simulate imperfect background knowledge.

To say when a de-anonymization was successful, is a matter of perspective.
Nevertheless the de-aonymization output proves that our anonymization algorithm works as intended. This means, by increasing the strength of the anonymization by increasing the size of k, we clearly observe that the success rate of the de-anonymization drops significantly.

8.3 Summary of the Experiment

In this chapter, we presented our experiment. The experiment has three parts: a data generation phase, an anonymization phase and a de-anonymization phase.
In the data generation phase we generated 10 different graphs, which serve as the input for the anonymization phase.

In the anonymization phase we generated for each graph 35 anonymizations and saved the loss of information.
The evaluation of that data resulted in the knowledge that an anonymization, which solely concentrates on one part of the network, achieves low loss of information in that part but high loss of information in every other part.
This decreases the data utility, because the overall loss of information is high.
In general the loss of information in the attribute part is smaller than the loss of information in the structural part. We affiliate this with the fact that we generated a smaller social network as intended because of complexity issues.
However, we observed that the overall loss of information is always smaller, if the anonymization is concentrated on multiple parts.
This observation answers the research question, if anonymization should be based on solely one part or multiple parts.

In the de-anonymization step we computed for each anonymization 8 de-anonymization with the de-anonymization-min algorithm and 64 de-anonymizations with the de-anonymization-value algorithm. Surprisingly the de-anonymization-min algorithm yields always better results on average.
Therefore, we presented only the output of the de-anonymization min algorithm.
By analyzing the output data, we observed that our anonymization algorithm achieves the desired feature.
The desired feature is, if the anonymization is strengthened by increasing k, the success rate of a de-anonymization attack drops.
Moreover, we noticed a correlation between loss of information and success rate. If the loss of information is high the success rate of the de-anonymization is poor and vise versa.

9 Conclusion

In this thesis we developed an anonymization algorithm for heterogeneous RDF graphs containing personal identifiable information. It protects the identity of persons by generalizing the graph. More specifically, our greedy anonymization algorithm finds k similar one-hop neighborhoods of persons and changes them so that they cannot be distinguished. This step is repeated until each person, which wants to stay private, is anonymized. The algorithm is mostly based on the algorithm proposed by Zhou et al. [ZP08]. However, it also uses ideas of research papers we presented in the related work chapter.

The anonymization results in the loss of information. The loss of information is measured by comparing the input graph with its anonymized version.
The difference between a heterogeneous graph and a homogeneous graph is that a heterogeneous graph consists of multiple edge types and vertex types. Therefore, the graph has multiple parts. For example a heterogeneous graph can contain a social network and a human collaboration network. Moreover, attributes of persons are modeled as a subgraph.
Our anonymization algorithm can concentrate solely on one part or multiple parts of the input graph. On which part or parts the algorithm is concentrated on we specify with weighting parameters. The result is that the loss of information drops in the parts that have been targeted. We proved that the algorithm works as intended by presenting the data of a conducted experiment. Moreover, we observed in the experiment data that anonymizations, which are concentrated on solely one part of the input graph, achieve low loss of information in that part but high loss of information in every other part. This results in an overall high amount of loss of information.
The overall loss of information is lowered by targeting multiple parts of the input graph. Therefore, we hypothesize that the anonymization of a heterogeneous graph should always be concentrated on multiple parts to achieve a good trade-off between data utility and loss of information. However, the priority could be to reduce the information loss in a specif part of a heterogeneous graph as much as possible with diregard to the overall loss of information. Therefore, an anonymization algorithm should have the ability to do so. Our anonymization algorithm provides this feature and is much more flexible than just that.

One problem we faced was that the computation of similar one-hop neighborhoods is very inefficient, because there is no known polynomial time algorithm to solve the general graph isomorphism problem. The idea of [ZP08] was to calculate a string for each neighborhood and thereby avoid the computation of many isomorphism tests. However, the computation of that

string turned out to be a bottleneck for us, because the anonymization algorithm computes so many depth first search trees that our program ran out of allocated memory, although we only focused only on the one-hop neighborhood to avoid complexity issues.
This problem could be mitigated by improving the implementation so that not all depth first search trees are saved in the memory to compute the minimal depth first search tree and running the algorithm on a server. In addition, the algorithm can be parallelized.

Moreover, in this thesis we developed two de-anonymization algorithms. They can be classified as guessing based neighborhood attacks. We call the first algorithm de-anonymization-min and the second one de-anonymization-value. They serve the purpose to re-identify persons in the anonymized graph, since the anonymization protects the identity of persons.
This is achieved by mapping each person in the anonymized graph to a set of identified persons in the background knowledge. Then the algorithm computes a set of candidates in that set. If the candidate set consists of more than one person the algorithm guesses between them.
The de-anonymization-min and de-anonymization-value algorithm are the same, except that they compute the set of candidates in different ways.

In the last phase of our experiment we de-anonymized each anonymization and measured the success rate. This is the percentage of correct re-identified persons. To our surprise the de-aonymization-min algorithm was on average always the better one. Therefore, we presented only its output data.
By analyzing the output data, we observed that our anonymization algorithm achieved the desired feature. The desired feature is, if the anonymization is strengthened by increasing k the success rate of a de-anonymization attack drops. Moreover, we noticed a correlation between information loss and success rate. If the information loss is high the success rate is poor and vise versa. The best success rate we achieved was 19%

All in all we showed that ideas developed for the anonymization of homogeneous graphs can be transfered to heterogeneous graphs. Moreover, we proved that anonymization helps against the threat of re-identification. However, if the anonymization is too strong, the data utility is low and if the data utility is too high de-anonymization attacks are very successful.
We believe that there is no best trade-off, because in the end it comes down to the priority of the data owner and how much the data owner wants to protect the identities of the individuals in the data that is shared with third parties. However, privacy becomes more and more a public demand. Therefore, data owners have to adapt and have to develop better anonymization approaches, which protect the privacy and keep the loss of information low, to stay relevant.

Bibliography

[AMS+16] D. Al-Azizy, D. Millard, I. Symeonidis, K. O'Hara, N. Shadbolt. "A Literature Survey and Classifications on Data Deanonymisation." In: *Risks and Security of Internet and Systems: 10th International Conference, CRiSIS 2015, Mytilene, Lesbos Island, Greece, July 20-22, 2015, Revised Selected Papers*. Ed. by C. Lambrinoudakis, A. Gabillon. Cham: Springer International Publishing, 2016, pp. 36–51. ISBN: 978-3-319-31811-0. DOI: 10.1007/978-3-319-31811-0_3. URL: http://dx.doi.org/10.1007/978-3-319-31811-0_3 (cit. on pp. 17, 25, 26).

[AMSO14] D. Al-Azizy, D. Millard, N. Shadbolt, K. O'Hara. "Deanonymisation in Linked Data: A Research Roadmap." In: *Internet Security (WorldCIS), 2014 World Congress*. 2014 (cit. on pp. 27, 30).

[BDK07] L. Backstrom, C. Dwork, J. Kleinberg. "Wherefore Art Thou R3579x?: Anonymized Social Networks, Hidden Patterns, and Structural Steganography." In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Banff, Alberta, Canada: ACM, 2007, pp. 181–190. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242598. URL: http://doi.acm.org/10.1145/1242572.1242598 (cit. on pp. 25, 26).

[CDFS07] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, P. Samarati. "κ-Anonymity." In: *Secure Data Management in Decentralized Systems*. Ed. by T. Yu, S. Jajodia. Boston, MA: Springer US, 2007, pp. 323–353. ISBN: 978-0-387-27696-0. DOI: 10.1007/978-0-387-27696-0_10. URL: http://dx.doi.org/10.1007/978-0-387-27696-0_10 (cit. on p. 22).

[CT08] A. Campan, T. M. Truta. "A clustering approach for data and structural anonymity in social networks." In: *Proceedings of the 2nd ACM SIGKDD International Workshop on Privacy, Security, and Trust in KDD (PinKDD'08), in Conjunction with KDD'08, Las Vegas, Nevada, USA*. 2008 (cit. on p. 24).

[CZF] D. Chakrabarti, Y. Zhan, C. Faloutsos. "R-MAT: A Recursive Model for Graph Mining." In: *Proceedings of the 2004 SIAM International Conference on Data Mining*, pp. 442–446. DOI: 10.1137/1.9781611972740.43. eprint: http://epubs.siam.org/doi/pdf/10.1137/1.9781611972740.43. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611972740.43 (cit. on p. 41).

[DEE10] S. Das, Ö. Eğecioğlu, A. El Abbadi. "Anonymizing weighted social network graphs." In: *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*. IEEE. 2010, pp. 904–907 (cit. on p. 25).

[DFJ04] L. Ding, T. Finin, A. Joshi. "Analyzing social networks on the semantic web." In: *IEEE Intelligent Systems (Trends & Controversies)* 8.6 (2004), pp. 815–820 (cit. on pp. 18, 34).

[DWS+11] M. Deng, K. Wuyts, E. Scandariato, B. Preneel, W. Joosen. "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements." In: *Requirements Eng (2011) 16: 3. doi:10.1007/s00766-010-0115-7*. 2011 (cit. on pp. 21, 24, 25).

[DZWG10] X. Ding, L. Zhang, Z. Wan, M. Gu. "A Brief Survey on De-anonymization Attacks in Online Social Networks." In: *2010 International Conference on Computational Aspects of Social Networks*. Sept. 2010, pp. 611–615. DOI: 10.1109/CASoN.2010.139 (cit. on pp. 23, 25, 37).

[FNT08] T. Feder, S. U. Nabar, E. Terzi. "Anonymizing Graphs." In: 2008 (cit. on p. 25).

[Har09] A. Harth. *Billion Triples Challenge data set Downloaded from*. 2009. URL: http://km.aifb.kit.edu/projects/btc-2009/ (cit. on p. 34).

[HB11] T. Heath, C. Bizer. "Linked Data: Evolving the Web into a Global Data Space (1st edition)." In: *Synthesis Lectures on the Semantic Web: Theory and Technology*. Morgan and Claypool, 2011, pp. 1–136. URL: http://linkeddatabook.com/editions/1.0/ (cit. on pp. 18, 27, 28).

[HCHD12] B. Heitmann, R. Cyganiak, C. Hayes, S. Decker. "An empirically grounded conceptual architecture for applications on the web of data." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.1 (2012), pp. 51–60 (cit. on p. 67).

[HMJT08] M. Hay, G. Miklau, D. Jensen, D. Towsley. "Resisting structural identification in anonymized social networks." In: *In Proceedings of the 34th International Conference on Very Large Databases (VLDB'08)*. 2008 (cit. on p. 24).

[Li16] Z. Li. "From Isomorphism-Based Security for Graphs to Semantics-Preservubg Security for the Resource Description Framework RDF." In: *Master Thesis at the University of Waterloo, Ibtario Canada*. 2016 (cit. on pp. 27, 29).

[LLV07] N. Li, T. Li, S. Venkatasubramanian. "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity." In: *2007 IEEE 23rd International Conference on Data Engineering*. Apr. 2007, pp. 106–115. DOI: 10.1109/ICDE.2007.367856 (cit. on pp. 17, 22).

[LT08] K. Liu, E. Terzi. “Towards Identity Anonymization on Graphs.” In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’08. Vancouver, Canada: ACM, 2008, pp. 93–106. ISBN: 978-1-60558-102-6. DOI: 10.1145/1376616.1376629. URL: http://doi.acm.org/10.1145/1376616.1376629 (cit. on pp. 17, 24).

[MGKV04] A. Machanavajjhala, J. Ghrke, D. Kifer, M. Venkitasubramaniam. “On the complexity of optional k-anonymity.” In: *Proceedings of the ACM Symposium on Principles of Database Systems*. 2004 (cit. on p. 22).

[MKGV07] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam. “L-diversity: Privacy Beyond K-anonymity.” In: *ACM Trans. Knowl. Discov. Data* 1.1 (Mar. 2007). ISSN: 1556-4681. DOI: 10.1145/1217299.1217302. URL: http://doi.acm.org/10.1145/1217299.1217302 (cit. on p. 22).

[NHD09] P. Nasirifard, M. Hausenblas, S. Decker. “Privacy concerns of FOAF-based linked data.” In: *Trust and Privacy on the Social and Semantic Web Workshop (SPOT 09) at ESWC09, Heraklion, Greece*. 2009 (cit. on p. 28).

[NS09] A. Narayanan, V. Shmatikov. “De-anonymizing Social Networks.” In: *2009 30th IEEE Symposium on Security and Privacy*. May 2009, pp. 173–187. DOI: 10.1109/SP.2009.22 (cit. on pp. 17, 26, 30, 37).

[QLZC16] J. Qian, X. Li, C. Zhang, L. Chen. “De-anonymizing Social Networks and Inferring Private Attributes Using Knowledge Graphs.” In: *IEEE INFOCOM - The 35th Annual IEEE International Conference on Computer Communications*. 2016 (cit. on pp. 30, 37).

[RGG15] F. Radulovic, R. García-Castro, A. Gómez-Pérez. “Towards the Anonymisation of RDF Data.” In: KSI Research, 2015 (cit. on pp. 27–29, 36).

[RKKT14] J. Rachapalli, V. Khadilkar, M. Kantarcioglu, B. Thuraisingham. “Towards Fine Grained RDF Access Control.” In: *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies*. SACMAT ’14. London, Ontario, Canada: ACM, 2014, pp. 165–176. ISBN: 978-1-4503-2939-2. DOI: 10.1145/2613087.2613092. URL: http://doi.acm.org/10.1145/2613087.2613092 (cit. on pp. 17, 18, 27, 29).

[SWE02] L. SWEENEY. “k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY.” In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570. DOI: 10.1142/S0218488502001648. eprint: http://www.worldscientific.com/doi/pdf/10.1142/S0218488502001648. URL: http://www.worldscientific.com/doi/abs/10.1142/S0218488502001648 (cit. on pp. 17, 21, 22).

[YH02] X. Yan, J. Han. “gSpan: graph-based substructure pattern mining.” In: *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* 2002, pp. 721–724. DOI: 10.1109/ICDM.2002.1184038 (cit. on pp. 47–49).

[ZG08] E. Zheleva, L. Getoor. “Preserving the Privacy of Sensitive Relationships in Graph Data.” In: *Privacy, Security, and Trust in KDD: First ACM SIGKDD International Workshop, PinKDD 2007, San Jose, CA, USA, August 12, 2007, Revised Selected Papers*. Ed. by F. Bonchi, E. Ferrari, B. Malin, Y. Saygin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 153–171. ISBN: 978-3-540-78478-4. DOI: 10.1007/978-3-540-78478-4_9. URL: http://dx.doi.org/10.1007/978-3-540-78478-4_9 (cit. on pp. 17, 23, 24).

[ZGAM09] E. Zheleva, L. Getoor, L. Arbuckle, B. Malin. “To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles.” In: 2009, pp. 531–540 (cit. on p. 26).

[ZP08] B. Zhou, J. Pei. “Preserving privacy in social networks against neighborhood attacks.” In: *Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE’08)*. 2008, pp. 506–515 (cit. on pp. 24, 29, 31, 36, 41, 45–47, 55, 57, 58, 79).

[ZPL08] B. Zhou, J. Pei, W. Luk. “A Brief Survey on Anonymization Techniques for Privacy Preserving Publishing of Social Network Data.” In: *SIGKDD Explor. Newsl.* 10.2 (Dec. 2008), pp. 12–22. ISSN: 1931-0145. DOI: 10.1145/1540276.1540279. URL: http://doi.acm.org/10.1145/1540276.1540279 (cit. on pp. 17, 22, 24, 25).