Linked Data and Graph Theory: Gaining Knowledge through the Structure of Heterogeneous Data

Gerard J. de Jong
Department of Computing Science, Utrecht University, The Netherlands
Q42, The Netherlands

Abstract

Linked data has become an integral part of modernizing cultural heritage collections. Similarly, the Rijksmuseum has transformed its digital collection into a linked data format. In partnership with Q42, the museum is developing a search and browse engine that allows both casual and professional users to explore this rich dataset. This thesis explores the intersection of linked data and graph theory to extract knowledge from the dataset's topology. By utilizing community detection algorithms, we identify clusters of similar actors which can be used in a recommendation engine to facilitate users' ability to explore unknown parts of the cultural heritage collection. This thesis proposes several data aggregation methods, several community detection algorithms, and a novel iterative approach to community detection. In our experiments, these techniques are applied to real-world cultural heritage data from the Rijksmuseum. The resulting clusters are evaluated against a validation set and shown to real-world users in a survey. The findings indicate our approach is promising, with the resulting clusters suitable for use in the recommendation engine.

2012 ACM Subject Classification Mathematics of computing → Graph theory; Information systems → Resource Description Framework (RDF)

Keywords and phrases Graph Theory, Community Detection, Graph Clustering, Linked Data, Cultural Heritage Data, Iterative Community Detection

Acknowledgements I would like to thank my thesis supervisor, Hans Bodlaender, for his general help in structuring my research and insightful feedback on my thesis. I want to thank my colleagues at Q42, especially Leonard Punt and Joris Bruil, for the support, help and truly amazing time at this company. Furthermore, I want to thank Nara Prasetya and Ismay de Jong for helping me proofread my thesis and structure my results during the final parts of my thesis. Lastly, I want to thank my friends and family for their unwavering support during the entire duration of my thesis research.

Disclaimer Generative AI tools have been used to aid in the writing of this thesis, by helping with structuring sentences and providing automated feedback.

1 Introduction

With the ever-growing importance of a strong online presence for cultural heritage organizations, the Rijksmuseum is increasing its use of linked data for its virtual collection of cultural heritage resources. Over the past few years, the museum has been working on expanding its linked data collection. In collaboration with the company Q42, they are fully embracing this new approach of storing and leveraging data, focusing on the development of a new search and browse engine for the museum's digital collection.

As part of this new search and browse engine a recommendation engine will be integrated, suggesting related search keywords to users based on their existing search keywords. Given the absence of information on which keywords relate to which other keywords, Q42 aims to employ innovative methods of exploring the linked data collection to generate this information.

This study aims to determine the feasibility of utilizing graph theory techniques to extract knowledge from the topology of linked data, by applying multiple community detection techniques on multiple aggregate datasets. More specifically, we focus on creating clusters of actors in the Rijksmuseum linked data collection such that the search and browse engine knows which actors relate to which other actors.

This paper is structured as follows: in Section 2 we discuss relevant literature; in Section 3 we explore the context in which this paper was written as well as dive into the dataset used; in Section 4 we outline the proposed methodology to gain more knowledge from linked data by applying community detection techniques; in Section 5 we outline the experiment setup as well as which metrics are gathered and how; in Section 6 we outline the results collected through the experiments; in Section 7 we discuss the results.

2 Literature review

The utilization of linked data concepts for modelling and disclosing cultural heritage data has a longstanding history. As early as the early 2000s, museums have started to digitize their collections into linked data formats. One notable example is the MuseumFinland project [14], which, at the time, disclosed 4 000 cultural heritage objects from three different museums in a linked data format utilizing seven different ontologies. These cultural heritage objects have remained freely searchable on the MuseoSuomi website[1] ever since.

As more cultural heritage organizations embraced the utilization of linked data themselves, research quickly commenced on leveraging this novel form of data disclosure and modelling within the cultural heritage domain. In 2005, the REACH project [7] introduced a novel hybrid approach integrating content-based visual search with ontology-based search, enhancing the search results. Following this, in 2006, the MultimediaN E-Culture project [21] demonstrated how linked data could enhance the effectiveness of simple text searches on cultural heritage data. It achieved this by mapping search text to related cultural heritage objects through paths in the linked data. In 2008, the CHIP demonstrator [23] further explored the potential of linked data by combining it with user preferences of cultural heritage objects to create personalized museum tours.

The relevance of utilizing linked data to model cultural heritage data continues to be significant. This is shown by the ongoing development and adoption of the Europeana data model [2, 11] and the Linked Art data model[2]. Notably, the Rijksmuseum has conducted extensive research into the utilization of linked data to model cultural heritage data [4, 6, 5] as well, transitioning from multiple data models to adopting the Linked Art data model to disclose its cultural heritage data. This model is collaboratively developed by a global consortium of cultural heritage organizations, including the Rijksmuseum. Further details about the Linked Art data model are provided in Section 3.2.

Research into the use of graph clustering algorithms to cluster semantic data is nothing new. In 2007, Fanizzi and d'Amato [9] proposed a novel distance measure to enable hierarchical clustering of semantic data. Given a feature set, distances between nodes of interest can be calculated to which a hierarchical clustering method can be applied. In 2008, Grimnes et. al [13] proposed three similarity measures for semantic data clustering. The first is quite similar to the method proposed by Fanizzi and d'Amato, a feature-vector-based distance measure. Secondly, they proposed a graph-based distance and lastly an ontology-based distance measure. Traditional community detection methods that solely rely on the graph topology can be applied to linked data as well. In 2012 Giannini [12] compares the use of an

[1] `http://museosuomi.fi/`
[2] `https://linked.art/`

overlapping community detection method [1] as well as a non-overlapping community detection method [3]. The clusters the non-overlapping community detection method found were not an aggregation of homogeneous resources but still, were useful as a way of summarizing linked data. In 2016, Martínez-Rodríguez et. al [15] continued this research into the use of traditional community detection methods by comparing multiple community detection algorithms on a linked data dataset. Their results consist of objective measurements of the communities found such as modularity score, where the multilevel/Louvain community detection algorithm [3] scored best.

3 Background and Dataset

3.1 Thesis context

This research has been conducted as part of a research internship for Q42 and the Rijksmuseum. For the past year, Q42 and the Rijksmuseum have been working on a new search and browse engine for the museum's online collection. The results of this research will be used in the development of this new search engine.

The Rijksmuseum is the largest and one of the most influential museums in the Netherlands, welcoming up to 2.7 million visitors annually. Its collection consists of over a million cultural heritage objects of which 770 000 objects have been digitized and disclosed using Rijksstudio[3].

Q42 is a strategic software development company with about 80 employees and is based in The Hague and Amsterdam. Some of their most notable projects include the HEMA app, the PostNL app, and the software behind Philips Hue. Besides this, they have been working together with The Rijksmuseum for over two decades, creating Rijksstudio, the current website, the Rijksmuseum app, and several other projects.

3.2 Dataset

For this thesis, we will examine the Rijksmuseum collection as a specific instance of linked data. The Rijksmuseum has progressively disclosed its cultural heritage collection data online utilizing linked data techniques as detailed in previous studies [4, 6, 5]. The collection is structured according to the Linked Art data model, a self-prescribed Linked Open *Usable* Data model. The "usable" aspect refers to a set of design principles aimed at making the data more accessible and practical. The data model describes the vocabularies used for storing cultural heritage objects and their related metadata as Resource Description Framework (RDF) triples. RDF triples are formatted as `(subject, predicate, object)`, where the subject and object are entities and the predicate is a relation linking them. The resulting linked data forms a heterogeneous directed graph, with entities as nodes and predicates as edges. This knowledge can be utilized to extract information from the data's structure, rather than its semantics.

The Rijksmuseum Linked Art dataset contains a significant portion of the museum's physical collection, including many items not currently on display. It contains 17 498 421 unique entities, of which 809 656 cultural heritage objects, and a total of 99 770 219 triples. The data is managed within a Blazegraph[4] RDF database, which can be queried using SPARQL[5], a specialized querying language for querying RDF databases.

[3] `https://www.rijksmuseum.nl/nl/rijksstudio`
[4] `https://blazegraph.com/`
[5] `https://www.w3.org/TR/rdf-sparql-query/`

In the Linked Art data model, a cultural heritage object is described as an entity with the type of "human-made object" to give a unified definition that would apply to any object a museum might have in its collection. More specifically, the CIDOC-CRM (a vocabulary used by Linked Art) definition `crm:E22_Human-Made_Object` is used. Similarly, we use the definition `crm:E39_Actor` for any person or organization, and `crm:E12_Production` for production events. Entities can be linked with predicates, just like the entities these predicates are also typed using the CRM vocabulary. An example of what such triples would look like in practice is given in Listing 1 with a visual representation in Figure 1.

Listing 1 Example of RDF triples related to the Night Watch and its creator

```
prefix crm: <http://cidoc-crm.org/cidoc-crm/>
prefix id: <https://id.rijksmuseum.nl/>

id:2005216 rdf:type crm:E22_Human-Made_Object
id:2005216 crm:P1_is_identified_by t11569244
id:2005216 crm:P108i_was_produced_by t11611745

t11569244 rdf:type crm:E33_E41_Linguistic_Appellation
t11569244 crm:P190_has_symbolic_content "... ‘The Night Watch’"

t11611745 rdf:type crm:E12_Production
t11611745 crm:P14_carried_out_by id:2105706

id:2105706 rdf:type crm:E39_Actor
id:2105706 crm:P1_is_identified_by t19497120

t19497120 rdf:type crm:E33_E41_Linguistic_Appellation
t19497120 P190_has_symbolic_content "Rijn, Rembrandt van"
```

crm:E22_Human-Made_Object
→ crm:P108i_was_produced_by →
crm:E12_Production
→ crm:P14_carried_out_by →
crm:E39_Actor
→ crm:P1_is_identified_by →
crm:E33_E41_Linguistic_Appellation
→ crm:P190_has_symbolic_content →
"Rijn, Rembrandt van"

crm:E22_Human-Made_Object
→ crm:P1_is_identified_by →
crm:E33_E41_Linguistic_Appellation
→ crm:P190_has_symbolic_content →
"...‘The Night Watch’"

Figure 1 Linked Art structure

The Linked Art data model is event-centered. This means rather than directly linking a human-made object and its creator, a human-made object is linked to a primary production event, which in turn is linked to a creator. For instance, in the example given before, `t11611745` is the id of the entity resembling the primary production event of the Night Watch. This in turn is linked to its primary maker, Rembrandt van Rijn.

Besides the limited set of object and predicate types shown here, Linked Art includes many more concepts. The ones in use by the Rijksmuseum can seen in Figure 13 in Appendix A or be explored in an interactive graph at `https://blazegraph-graph-vis.denni.dev/`.

3.3 Community Detection

Community detection algorithms are a set of well-known algorithms within the field of graph theory. These algorithms analyze the structure of a graph to identify clusters of nodes, or communities, within the graph [10]. There are various approaches to community detection, including modularity maximization methods, information-theoretic methods, and statistical inference methods [17]. Typically, these methods are applied to social networks, which are homogeneous[6] compared to the heterogeneous[7] graphs we encounter in our dataset.

In this research, we aim to use community detection to identify communities of actors within the Rijksmuseum's linked data. To enable us to do this, we aggregate the data to create a structure similar to a social network. This structure allows community detection algorithms to be applied, resulting in clusters of actors which share similarities based on the aggregation techniques used. Detailed descriptions of the specific data aggregation techniques and community detection algorithms utilized in this study are provided in Section 4.

4 Methodology

This section outlines the methodologies used to apply community detection to the linked data dataset and is structured as follows: in Section 4.1 we discuss methods of data filtration; in Section 4.2 the techniques used to aggregate the linked data into a homogeneous graph are detailed; in Section 4.3 we describe three community detection methods utilized in this study; and lastly, in Section 4.4 we introduce a novel technique of iteratively running community detection algorithms to ensure smaller communities.

4.1 Data filtration

The Rijksmuseum Linked Art dataset contains 809 656 cultural heritage objects created by 155 186 distinct actors. Among these, only 71 264 actors have participated in a production event, and a smaller subset of only 20 137 actors have contributed to the production of at least one cultural heritage object with a picture shown on the website, in "Rijksstudio". This highlights that a significant portion of the actors in the dataset have minimal information available. To mitigate the risk of wrong conclusions drawn from sparse data, we propose two filtration methods:

1. Limiting the dataset to actors with at least one picture available on Rijksstudio, resulting in a filtered dataset of 20 137 actors.

[6] All nodes are the same type
[7] Nodes are of different types

2. Limiting the dataset to actors with at least four pictures available on Rijksstudio, resulting in a filtered dataset of 5 830 actors.

4.2 Data aggregation

Transforming the heterogeneous linked data dataset into a homogeneous graph structure is crucial for the application of traditional community detection techniques. To achieve this, we developed a Python program that interfaces with a Neo4j graph database containing the linked data dataset. This program utilizes the Cypher query language to extract the necessary data, which is then aggregated into an undirected multigraph using one of the four methods mentioned in this section.

Collaboration

In the original dataset, certain production events (denoted by `E12_Production`) are comprised of multiple sub-events through the `P9_consists_of` relationship. Each sub-event has an actor associated with it through the `P14_carried_out_by` relationship. To construct a network of actors who have collaborated on the same cultural heritage object, we iterate over these composite production events and create an edge between any two actors that are part of the same production event. If actors have collaborated in multiple production events, multiple edges are created between their nodes, making them strongly connected. This process is visually represented in Figure 2. The underlying assumption is that actors collaborating on cultural heritage objects will likely produce works that share similarities in style, category, or other attributes.

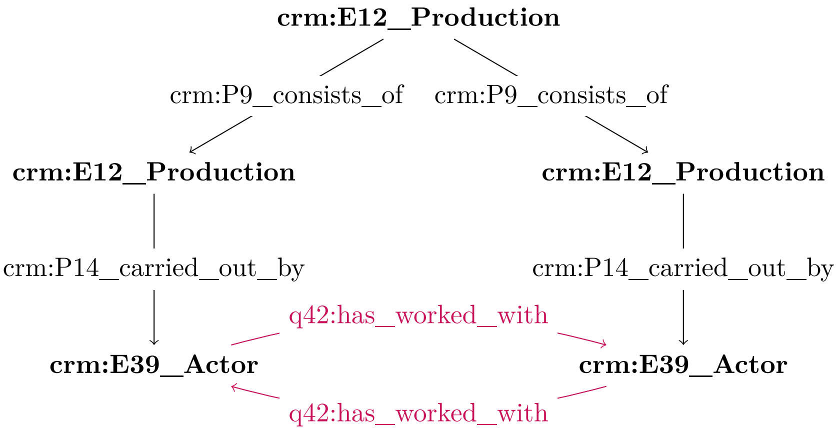

Figure 2 Visualization of collaboration aggregation

The number of edges and nodes in each resulting aggregate graph are detailed in Table 1.

File name	# edges	# nodes
collaboration.nt	1 491 958	55 285
filtered-collaboration.nt	507 782	7 982
filtered4-collaboration.nt	255 524	3 483

Table 1 # of edges and # of unique actors in each collaboration dataset

Technique

In addition to examining the actors themselves, it is also insightful to consider the cultural heritage objects they have created. Each production event may have one or more `E55_Type` nodes associated with it through the `P32_used_general_technique` relationship, representing which technique is used during the production of the cultural heritage object. The dataset contains a total of 1 010 distinct techniques, ranging from broad categories like "painting", "sculpting", and "casting", to more specific techniques like "aizuri-e"[8], "chiaroscuro"[9], and "Façon de Venise"[10]. To construct a network of actors who have used the same techniques, we create an edge between two actors for every technique they share. Consequently, actors who share many techniques will be more strongly connected. This process is visually represented in Figure 3.

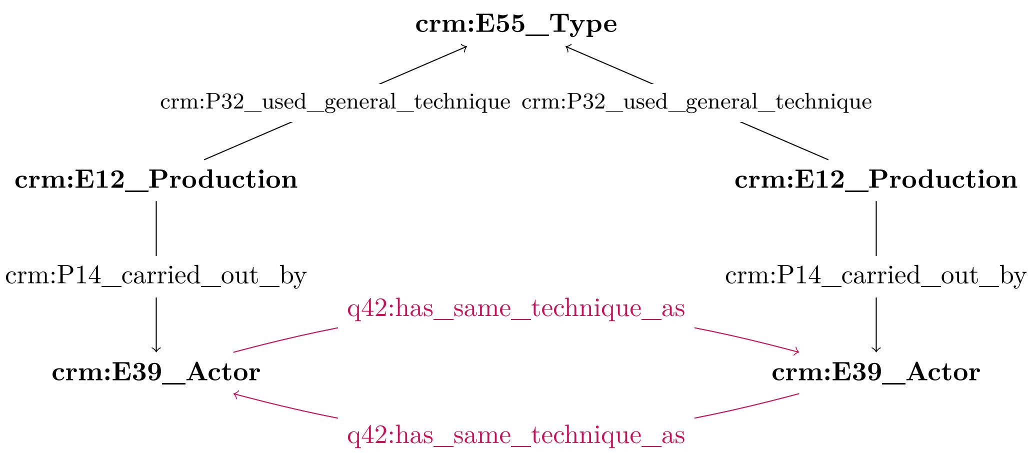

Figure 3 Visualization of technique aggregation

The number of edges and nodes in each resulting aggregate graph are detailed in Table 2.

File name	# edges	# nodes
technique.nt	91 147 760	23 441
filtered-technique.nt	63 786 788	17 972
filtered4-technique.nt	13 789 038	5 624

Table 2 # of edges and # of unique actors in each technique dataset

Location

In addition to information about techniques and collaboration, the Rijksmuseum linked data collection also contains information about the location of production events. A production event may have one or more `E53_Place` nodes associated with it through the `P7_took_place_at` relationship denoting the location where a production event took place. The dataset includes 17 430 distinct locations, ranging from local neighbourhoods and cities

8 Predominantly blue Japanese woodblock prints
9 The use of strong tonal contrasts in lighting to portray depth and volume in paintings
10 Venetian style of glass from the 16th/17th century

to countries, continents and even planets. To construct a network of actors who have worked in the same location, we create a mapping of locations and which actors have been involved in production events in these locations. An edge is created between any two actors for every location they share as detailed in Figure 4. Consequently, actors who have worked in many of the same locations are more strongly connected.

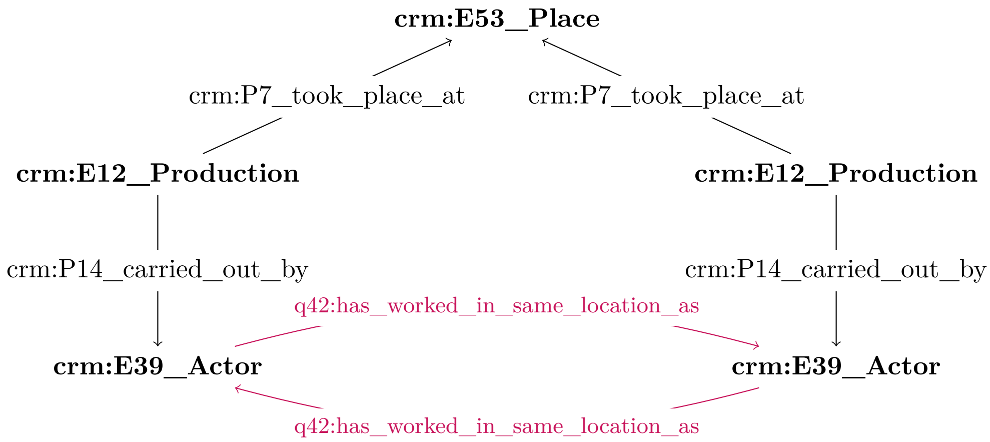

Figure 4 Visualization of location aggregation

The number of edges and nodes in each resulting aggregate graph is detailed in Table 3.

File name	# edges	# nodes
location.nt	98 779 582	43 757
filtered-location.nt	20 817 814	15 691
filtered4-location.nt	5 313 702	5 151

Table 3 # of edges and # of unique actors in each location dataset

Teacher-Student

The RKD (Netherlands Institute for Art History) offers an openly available dataset called RKDartists, which provides data on 257 792 distinct actors[11]. This dataset contains extensive information on actors including the locations in which they were active, their family relations, which actors influenced them, which actors taught them and which actors they have taught, all available in a linked data format. Especially the teacher-student relations offer valuable insights into which actors might be closely related to which other actors.

Given that these actors are only linked within the RKD dataset, a mapping from RKD URI to Rijksmuseum URI based on the actors' names has to be created. To construct a network of actors who have had a teacher-student relationship, we use this mapping and create an edge between any two actors who have had a teacher-student relationship in the RKD dataset as illustrated in Figure 5. In contrast to the aforementioned aggregate networks, this network does not contain any parallel edges between actors. The RKDartists dataset

[11] https://research.rkd.nl/

does not give any information on how strong the teacher-student relationship was between any two actors, thus we consider every teacher-student relationship equally strong.

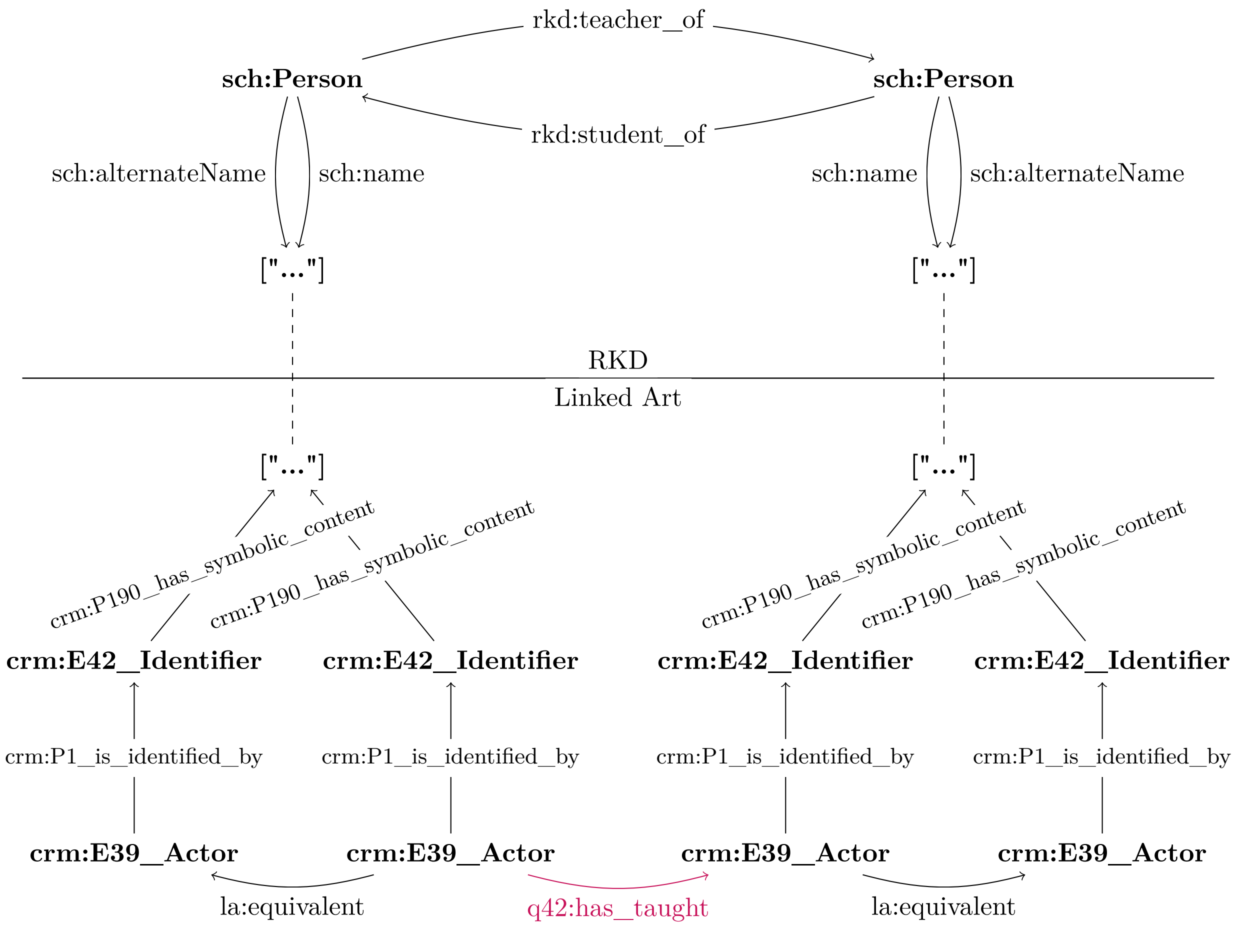

Figure 5 Visualization of teacher-student aggregation

The number of edges and nodes in each resulting aggregate graph is detailed in Table 4.

File name	# edges	# nodes
teacher-student.nt	8 896	7 345
filtered-teacher-student.nt	3 274	2 927
filtered4-teacher-student.nt	1 372	1 265

Table 4 # of edges and # of unique actors in each teacher-student dataset

Combinations

While individual datasets provide valuable insights into the relations of actors based on each particular aggregation technique, it might not be sufficient to find small clusters of similar actors. For example, clustering actors based on technique alone could create clusters of actors who all have created paintings, but with wildly different art styles. To address this issue, we propose a method that combines the aggregate datasets into supersets. This results in $|P(S) \setminus \{\{\}\}| = 2^{|S|} - 1 = 2^4 - 1 = 15$ distinct combinations. The number of edges and unique actors in each resulting combined graph is detailed in Table 5.

	unfiltered		filtered		filtered4	
File name	# edges	# nodes	# edges	# nodes	# edges	# nodes
teacher-student	8 896	7 345	3 274	2 927	1 372	1 265
collaboration	1 491 958	55 285	507 782	7 982	255 524	3 483
technique	91 147 760	23 441	63 786 788	17 972	13 789 038	5 624
location	98 779 582	43 757	20 817 814	15 691	5 313 702	5 151
teacher-student-collaboration	1 500 854	57 338	511 056	8 731	256 896	3 685
teacher-student-technique	91 156 656	26 907	63 790 062	18 356	13 790 410	5 664
teacher-student-location	98 788 478	46 988	20 821 088	16 365	5 315 074	5 280
collaboration-technique	92 639 718	68 452	64 294 570	18 574	14 044 562	5 711
collaboration-location	100 271 540	66 249	21 325 596	16 728	5 569 226	5 391
technique-location	189 927 342	49 240	84 604 602	19 323	19 102 740	5 783
teacher-student-collaboration-technique	92 648 614	69 413	64 297 844	18 715	14 045 934	5 718
teacher-student-collaboration-location	100 280 436	67 688	21 328 870	17 080	5 570 598	5 438
teacher-student-technique-location	189 936 238	51 532	84 607 876	19 544	19 104 112	5 802
collaboration-technique-location	191 419 300	70 249	85 112 384	19 568	19 358 264	5 810
teacher-student-collaboration-technique-location	191 428 196	71 144	85 115 658	19 677	19 359 636	5 814

Table 5 Number of edges and number of unique actors in each combined dataset

As mentioned in Section 4.1, the unfiltered datasets contain many actors of which very little information is available. A 55.5% decrease in edges and a 72.3% decrease in nodes between the largest unfiltered and filtered datasets can be seen in Table 5, indicating actors with little to no information make up significant parts of the unfiltered datasets, which could lead to erroneous conclusions in our experiments. In contrast, the datasets only containing actors with four or more pictures available on Rijksstudio (`filtered4`) eliminate too many actors, potentially missing out on classifying valuable actors by including these datasets in our analysis. Therefore, our analysis only includes the datasets containing actors with one or more pictures available on Rijksstudio (`filtered`). Table 11 in Appendix B provides detailed graph metrics of each combined `filtered` dataset.

4.3 Algorithms

To process these aggregate datasets and create communities of actors who might be similar to one another, we propose three community detection algorithms.

Louvain

The `Louvain` community detection method, introduced in 2008 by Blondel et al. [3], is one of the most widely used community detection methods. It aimed to speed up community detection in large graphs through the local moving of nodes by calculating a difference in modularity score for each move. Modularity is defined as the measure of the density of edges within a community versus the density of edges between communities [16]. The formula for this is as follows:

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta\left(c_i, c_j\right) \tag{1}$$

In Equation (1), $A_{i,j}$ is the sum of the weights of edges between i and j, $k_i = \sum_j A_{i,j}$ is the sum of all weights of edges connected to i, c_i is the community i is in, $\delta\left(u, v\right)$ is the function that returns 1 if $u = v$ and 0 otherwise and $m = \frac{1}{2} \sum_{i,j} A_{i,j}$ is the sum of all weights in the graph.
The Louvain method is a two-part algorithm that is run iteratively:

1. **Local Moving** Each node is initially considered as its own community. For each community, it will calculate a difference in modularity if it were to merge with its neighbouring community. It will then merge the community with the neighbour with the largest positive change in modularity.
2. **Aggregation** An aggregate graph is created by merging all nodes in the same community into a single node in the aggregate graph.

This iterative process is repeated until no improvement in modularity is possible, resulting in a clustering of nodes where modularity is maximized across the entire graph.

Leiden

The `Leiden` community detection method, introduced in 2019 by Traag et al. [22], is a variation of the `Louvain` community detection method designed to ensure well-connected communities. This method modifies the `Louvain` method by adding a refining phase after the local moving phase to prevent disconnected communities from being created.

Due to the deterministic behaviour of the `Louvain` community detection method, a node might be added to a neighbouring community before another node is removed from the same community, causing the community to become disconnected. To prevent this issue, the refining phase in the `Leiden` community detection method re-applies the local moving phase on each community found in the initial local moving phase, breaking up any disconnected communities into separate communities before they are aggregated in the next phase.

Infomap

Unlike the modularity-based `Louvain` and `Leiden` community detection methods, the `Infomap` community detection method has a flow-based approach [20, 19]. This approach aims to minimize the map equation, a measure to calculate the expected description length of a random walk through the graph, measured in bits per step.

A random walk represents a random sequence of connected nodes in a graph such that a path is formed. This path can be described using bits by creating a sequence of bits which is unique to this path. In large graphs, giving each node its own binary representation can lead to extensive path descriptions. The `Infomap` method aims to minimize the length of such a description by applying Huffman coding, grouping often sequentially visited nodes to decrease their description length, decreasing the overall description length of any random walk.

The `Infomap` algorithm simulates random walks through the graph, measuring the visit frequencies of each node. It then applies a deterministic greedy search algorithm to the results, refining them using a simulated annealing approach, to minimize the map equation over the entire graph. This results in communities of nodes, such that the expected binary description, the codelength, of a random walk is minimized.

4.4 Iterative Approach

Given that the resulting communities will be used to generate labels for a recommendation engine, it is impractical for these communities to be excessively large. Labelling an actor as similar to a large number of other actors defeats the purpose of the recommendation engine.

To ensure usable results we introduce a novel iterative strategy that reduces the size of the discovered communities. This strategy iteratively applies community detection algorithms to the largest community, subdividing it into smaller communities. If the algorithm is unable to subdivide the largest community into smaller communities it is skipped and the next largest community is used. This process repeats until one of the following stopping conditions is met: a maximum number of iterations has been reached, the largest community is below a minimum community size, or none of the communities can be subdivided any further. In all of our experiments, we use a maximum of 1 000 iterations and a minimum community size of 10 nodes.

This strategy no longer optimizes for a global measure of community structure but rather optimizes for these locally. The main goal of this strategy is to reduce community sizes to increase the usability of the results, rather than optimizing for a global measure of community structure.

5 Experiments

Given the 15 datasets, three community detection techniques and whether to run the community detection algorithm iteratively, there are $15 \cdot 3 \cdot 2 = 90$ distinct combinations to test. Since these datasets do not have a ground truth, validating the results is quite hard. Firstly, we focus on bringing the number of results down. We will look at a combination of objective measures as well as compare the results to an approximate form of ground truth. Secondly, we will present a select number of results to real-world users in a survey and perform validation this way.

5.1 Implementation

To gather results we have written an application in Python which can run any of the community detection algorithms with any of the datasets, iteratively or not. This program uses CDLib[12] to provide a common interface for all community detection algorithms as well as a common interface for the results (`NodeClustering`). Under the hood, CDLib uses pre-existing implementations of each of the algorithms used in this paper. For the `Louvain` algorithm, this is the `python-louvain` implementation available on PyPi and GitHub[13]. For the `Leiden` algorithm, this is the `leidenalg` implementation, created by the original authors of the Leiden paper [22], available on PyPi and GitHub[14]. And lastly, for the `Infomap` algorithm, this is the `infomap` implementation [8], available on PyPi and GitHub[15]. The program is run on a Dell XPS 15 7950 with an Intel i7-9750H (12) @ 4.500GHz and 64GB of memory and all resulting `NodeClustering`s are saved to the disk for later processing and gathering of metrics.

[12] https://github.com/GiulioRossetti/cdlib
[13] https://github.com/taynaud/python-louvain
[14] https://github.com/vtraag/leidenalg
[15] https://github.com/mapequation/infomap

5.2 Objective metrics

Given a graph $G = (V, E)$ and a set $\mathcal{C} = \{C_1, \dots, C_n\}$ of non-overlapping communities with $V = \bigcup_{i=1}^{n} C_i$, we can calculate a set of metrics which can be used to gain insights into the quality of the results. These metrics are collected from the previously saved `NodeClustering`s using a Python script. We collect the following metrics:

Community count The number of communities found by the community detection algorithm. Calculated as: $|\mathcal{C}|$.

Average community size The average community size is the average size of communities found by the community detection algorithm. A high average community size is a sign of broad general communities whereas a lower average community size is a sign of more specific communities. Calculated as: $\sum_{i=1}^{|\mathcal{C}|}\{|C_i|\}/|\mathcal{C}|$.

Max community size The maximum size of the communities found. A larger max size means there is a large community with a very broad set of actors, whereas a smaller max size means the community detection was able to find more specific communities. Calculated as: $\max_{C_i \in \mathcal{C}} |C_i|$.

5th percentile community size The 5th percentile community size shows the smallest community size, accounting for any outliers.

95th percentile community size The 95th percentile community size shows the largest community size, accounting for any outliers. If this number is substantially lower than the max community size, we can assume there are only very few outlying communities in terms of size.

Modularity Modularity measures the community structure in a graph. Given the graph topology and a set of communities, we can calculate the modularity using Equation (1). A high modularity score is a sign of communities with strongly intra-connected nodes and weak inter-connected nodes between communities.

Codelength The lower bound of the average bits needed to describe a single step in a random walk of the graph. A lower codelength means the partitions allow for efficient mapping of a random walk across the graph, whereas a higher codelength means more bits per step are needed to describe such a path. The codelength is explained more in-depth in Section 4.3.

5.3 Semi-supervised learning

A form of semi-supervised learning has been performed to gain more insights into the results. The Rijksmuseum website includes a feature called Rijksstudio[16], which allows users to create curated sets of cultural heritage items for others to explore and interact with. For our semi-supervised learning, we take these user sets and compare the actors of the works in the set with the communities in our results. Rijksstudio contains a total of 196 976 of these user sets. However, many of these user sets are expected to be of very low quality, e.g. a user simply making collections of works that they would like to see in the museum. Therefore, we have chosen to filter out all user sets which have less than 500 views, contain works of 50 or more distinct actors, or contain works of two or fewer distinct actors. For each dataset, we “equalize” the result set and validation set by taking the intersection of all actors appearing in both sets. The resulting actor counts for each dataset are shown in

[16] `https://www.rijksmuseum.nl/rijksstudio`

Table 6. While this method does not cover the entire result set, we still expect it to give an accurate representation.

Dataset	# nodes	Intersection
teacher-student	2 927	1 615
collaboration	7 982	3 557
technique	17 972	4 846
location	15 691	4 956
teacher-student-collaboration	87 31	3 798
teacher-student-technique	18 356	5 094
teacher-student-location	16 365	5 240
collaboration-technique	18 574	5 199
collaboration-location	16 728	5 297
technique-location	19 323	5 417
teacher-student-collaboration-technique	18 715	5 287
teacher-student-collaboration-location	17 080	5 404
teacher-student-technique-location	19 544	5 560
collaboration-technique-location	19 568	5 567
teacher-student-collaboration-technique-location	19 677	5 633

Table 6 The intersection between each dataset and the validation set

Using these equalized sets we can compare all results using the Rand index [18]. The Rand index measures the similarity between two sets of clusters and is commonly used to compare the results of clustering algorithms. It gives a value between 0 and 1, where 0 indicates randomness and 1 indicates perfect similarity. It is computed by iterating over all pairs of actors and checking if they are in the same cluster in the result set and the validation set. We count the times a pair is either in the same cluster in both sets or not in the same cluster in both sets. These are the true positives and true negatives. From this, we can calculate the Rand index using Equation (2).

$$R = \frac{TP + TN}{n(n-1)/2} \tag{2}$$

In Equation (2), TP is the number of true positives, TN is the number of true negatives and n is the number of actors.

The Rand index assumes that neither of the sets contains overlapping clusters. However, in our case, the validation set *does* have overlapping clusters. This is not necessarily undesirable, because we can still calculate the Rand index. It is irrelevant whether a pair occurs only in one cluster or in multiple clusters in the validation set for the index to work, as long as the result set only contains non-overlapping clusters, which it does.

5.4 Validation survey

The metrics mentioned in Section 5.2 and Section 5.3 give insights into which datasets and techniques provide the most valuable communities to be utilized in a recommendation engine. To validate these results, we survey real-world users to validate whether the results are of sufficient quality. Users are asked to give a subjective rating of a set of actors on a scale from 1 through 5 where 1 means the actors do not match and 5 means the actors match well. Besides this, the users can provide additional textual feedback. Since many actors in the dataset are not well-known, we aid the respondents by showing the four top-rated human-made objects of each actor, obtained from the Rijksstudio platform.

The survey consists of a website created with React and TypeScript and a back-end created solely in TypeScript. A Google Datastore is used to store the questions and the responses from users. Before the survey, the datastore is populated with a set of "questions". Each question is simply a set of actors, which is generated from the communities in our result sets. For every community with at least four actors, we take the six actors whose works cumulatively have the most likes on Rijksstudio. This results in questions with four to six actors. Whenever a new user starts the survey they are pseudo-randomly assigned a total of 24 questions from each result set, such that questions from every result set appear approximately just as often as any other. Screenshots of what the survey looks like in practice can be found in Appendix D. When the user has completed the survey they can request 12 more questions to be assigned to them in the same way as before, ensuring no duplicate questions are assigned. The user can do this as often as they want.

6 Results

This section presents the results of the experiments outlined in Section 5. All results presented in this section are obtained by running the algorithms on the filtered datasets as described in Section 4.1, both using the non-iterative and iterative approach as described in Section 4.4. The figures in this section will use abbreviated dataset names. Table 7 shows a conversion table for each dataset and its abbreviated counterpart. A complete set of all results can be found in Table 12 in Appendix C.

Dataset	Abbr. Dataset
filtered-collaboration	col
filtered-collaboration-location	col-loc
filtered-collaboration-technique	col-tec
filtered-collaboration-technique-location	col-tec-loc
filtered-location	loc
filtered-teacher-student	tea
filtered-teacher-student-collaboration	tea-col
filtered-teacher-student-collaboration-location	tea-col-loc
filtered-teacher-student-collaboration-technique	tea-col-tec
filtered-teacher-student-collaboration-technique-location	tea-col-tec-loc
filtered-teacher-student-location	tea-loc
filtered-teacher-student-technique	tea-tec
filtered-teacher-student-technique-location	tea-tec-loc
filtered-technique	tec
filtered-technique-location	tec-loc

Table 7 A conversion table for the abbreviated dataset names

6.1 Community Structure

The applied algorithms use two distinct measures of community structure for which they optimize. The `Louvain` and `Leiden` algorithms both use the modularity score, whereas the `Infomap` algorithm uses the codelength as calculated by the map equation. Both are detailed in Section 4.3. We expect these measures to be correlated with one another in our output. The modularity and codelength for each of our result sets are plotted in Figure 6. This figure shows an inverse correlation between the modularity and codelength, which indicates both metrics give relevant insights into the quality of the community structure of the given result. The figure also shows how the iterative approaches generally perform worse than the non-iterative approaches. This is to be expected, given that the algorithms are run locally on increasingly smaller sets of nodes and are no longer optimizing their measure of community structure globally across the graph.

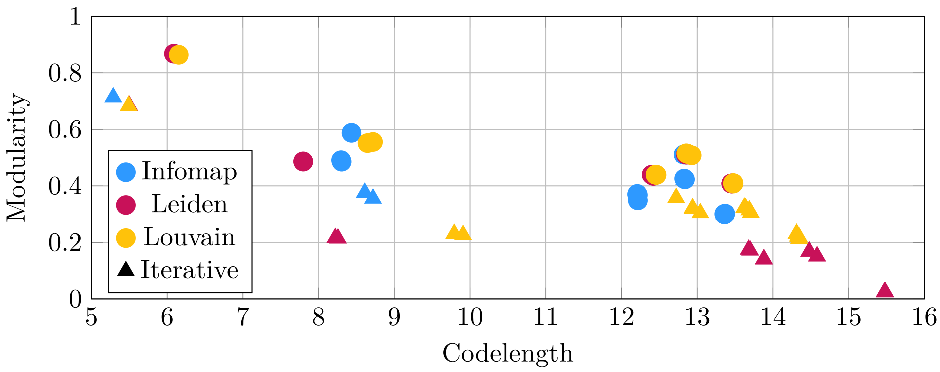

Figure 6 Codelength versus modularity

This phenomenon is even more apparent in Figure 7 where it's clear the modularity-based community detection methods have a much lower modularity score when run iteratively. Surprisingly, the `Infomap` algorithm seems to be largely unaffected. The iterative approaches do seem to be very effective at reducing community sizes. But again, the `Infomap` algorithm seems largely unaffected. It seems like the `Infomap` algorithm has trouble reducing the size of existing communities, presumably because these communities already have close to the optimal codelength. This would also explain why the modularity barely changes, the communities aren't changing either.

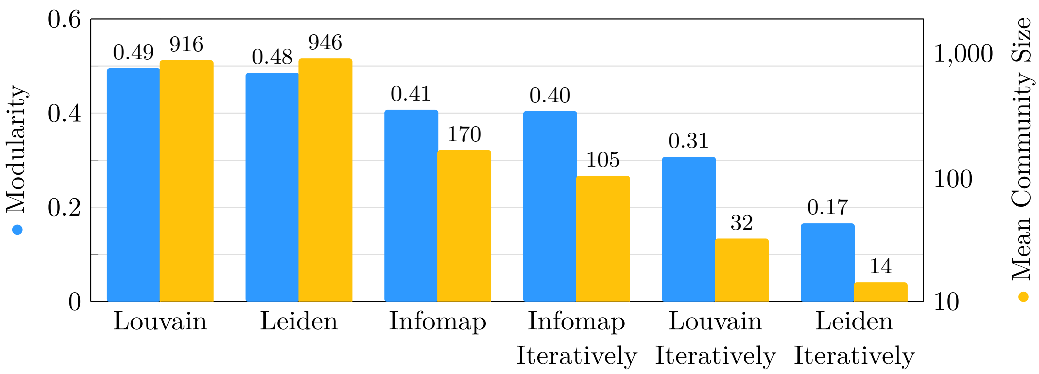

Figure 7 The average modularity and mean community size per algorithm

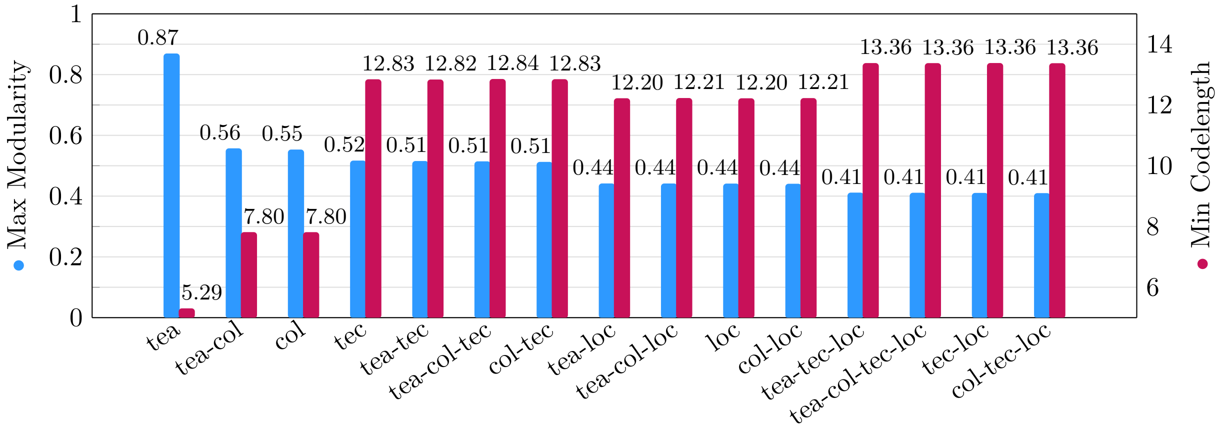

Figure 8 The maximum modularity and minimum codelength per dataset

The maximum modularity and minimum codelength for each dataset are plotted in Figure 8 ordered by maximum modularity from left to right. We observe the teacher-student dataset has both the maximum modularity and the minimum codelength out of all datasets. This means this dataset has an inherently strong community structure. This is to be expected; many of these teacher-student pairs are likely from the same region and time period and thus will create tightly-knit communities that are not well connected to other communities. On the other hand, we can see there are four worst-scoring datasets, each of which contains the technique-location pair of datasets. Looking back at Table 5 we can see that this combined dataset contains 84 604 602 edges, while the combination of all datasets contains 85 115 658 edges. The technique-location combination by far overshadows the other datasets in the fully combined dataset in terms of size and it therefore scores just as low as the technique-location dataset. Looking at the modularity scores of the other datasets, we observe the collaboration datasets also have an above-average community structure. Combining the collaboration dataset with the teacher-student dataset even increases its modularity score. Looking at the location modularity scores, we can see it performs the worst of all single dataset datasets. Combined with any other dataset, the location dataset lowers the community score.

6.2 Semi-supervised learning

Besides running the experiments, the results are also validated against a validation set as described in Section 5.3. In Figure 9 the mean community size is plotted against the Rand index. The figure shows an inverse correlation between the mean community size and the Rand index, indicating that having smaller more specific communities increases how well the results overlap with the validation set. Furthermore, we find that especially the iterative approaches result in a higher Rand index because these approaches lead to a smaller mean community size. We see that mostly the `Louvain` and `Leiden` algorithms perform very well whereas the `Infomap` algorithm generally scores lower in terms of the Rand index. We again see a clear difference in the performance of the iterative approach versus the non-iterative approach for the `Louvain` and `Leiden` algorithms, where this, just like in Figure 6, lacks for the `Infomap` algorithm.

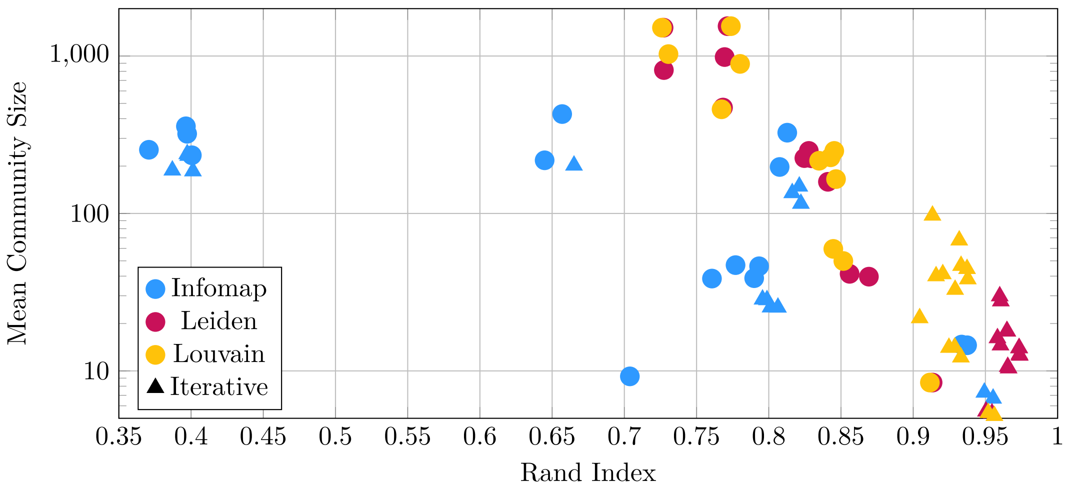

Figure 9 Rand index versus mean community size

The aggregate results of each algorithm are summarized, in order of Rand index from left to right, in Figure 10. The `Leiden` and `Louvain` algorithms outperform the `Infomap` algorithm whether run iteratively or not. Furthermore, the figure shows that especially the `Leiden` algorithm sees a notable decrease in mean community size when run iteratively. This is also supported by the large drop in modularity as seen in Figure 7 as opposed to the smaller drop in modularity for the `Louvain` algorithm. Similarly, we find that the `Leiden` algorithm outperforms the `Louvain` algorithm in terms of the Rand index when run iteratively. This is quite surprising as both techniques are quite similar. This could be an interesting subject of future research.

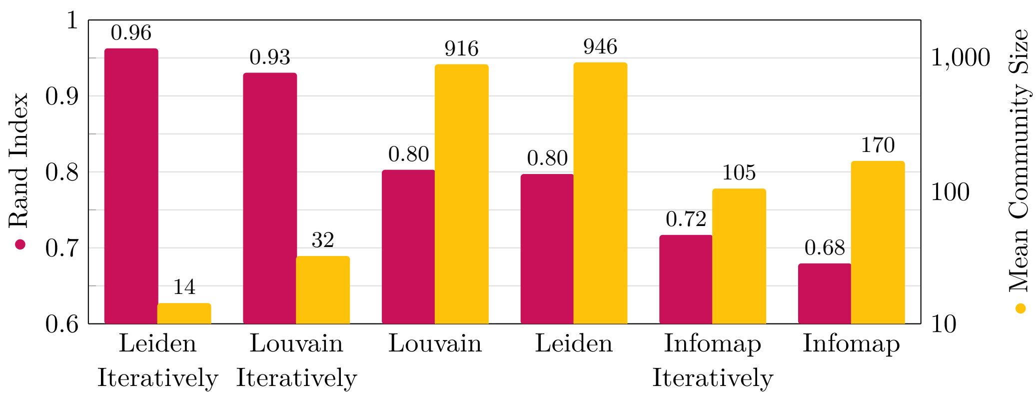

Figure 10 The average Rand index and mean community size per algorithm

Figure 11 shows the average Rand index and mean community size aggregated by each of the datasets. Similarly to Figure 8 we find that the teacher-student dataset, the collaboration dataset and the combination of the two appear as the top three best-performing datasets. Surprisingly, the location datasets now score better than the technique datasets. The Rand indices are slightly higher but especially the mean community sizes are much lower for any dataset with the location dataset in it. However, similarly to before, any dataset with the combination of the technique and location datasets performs quite badly, since this combination still overshadows any other datasets because of its sheer size.

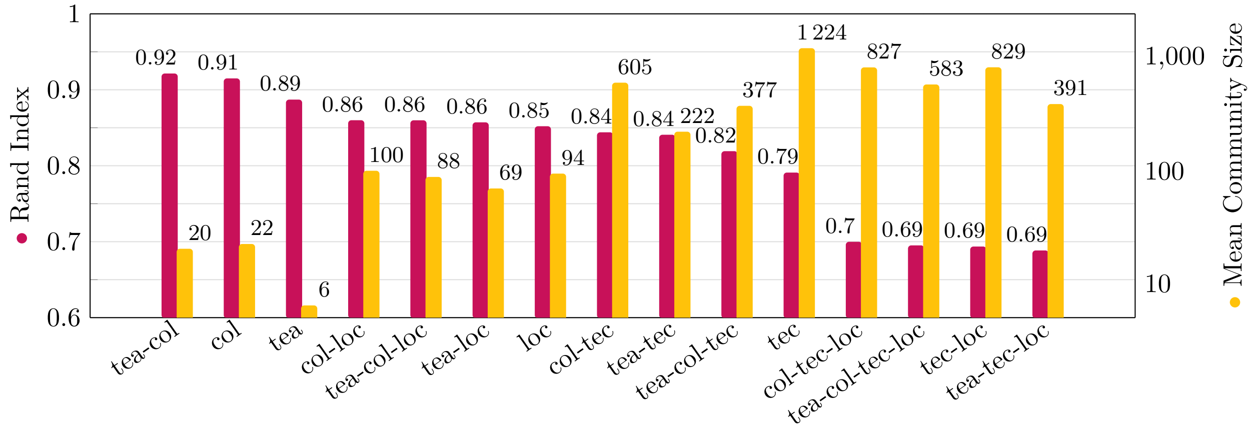

Figure 11 The average Rand index and mean community size per dataset

6.3 Validation survey results

A selection of six result sets has been made to be included in the validation survey described in Section 5.4, based on the metrics shown in this section and discussions with a domain expert. The teacher-student-collaboration and the teacher-student-collaboration-technique datasets were selected because these datasets have a high average modularity and Rand index across all algorithms. For the techniques, the non-iterative `Louvain` algorithm and the iterative `Louvain` and `Leiden` algorithms were selected to create a comparison against the iterative versus the non-iterative approach, as well as a comparison between the effectiveness of using the `Leiden` algorithm over the `Louvain` algorithm when applied iteratively. Table 8 shows the number of communities and questions in each result set. The questions are generated from the communities in the result set as described in Section 5.4, resulting in a total of 2 722 questions across the six result sets. Note the discrepancy in the number of communities versus the number of questions, denoting many communities contain less than four actors.

Technique	Dataset	# Communities	# Questions
Louvain	tea-col	175	27
Louvain Iteratively	tea-col	1 683	974
Leiden Iteratively	tea-col	1 621	816
Louvain	tea-col-tec	21	7
Louvain Iteratively	tea-col-tec	570	240
Leiden Iteratively	tea-col-tec	1 050	658

Table 8 The result sets presented in the validation survey

The survey received answers from 525 distinct users, answering 13 361 questions in total. Table 9 shows the average rating over all answered questions corresponding to each result set, with Figure 12 providing a visual representation of the average rating versus the Rand index. It's interesting to see the `Louvain` algorithm performs well on the smaller dataset, but does much worse on the larger dataset. Inversely, the two iterative approaches perform much better on the larger dataset than on the smaller dataset.

Technique	Dataset	# Answers	Avg. Rating	Rand Index
Louvain	tea-col	2 329	3.74	0.85
Louvain Iteratively	tea-col	2 231	3.53	0.96
Leiden Iteratively	tea-col	2 232	3.54	0.95
Louvain	tea-col-tec	2 083	3.08	0.78
Louvain Iteratively	tea-col-tec	2 277	3.67	0.93
Leiden Iteratively	tea-col-tec	2 209	3.80	0.97

Table 9 The results of the validation survey

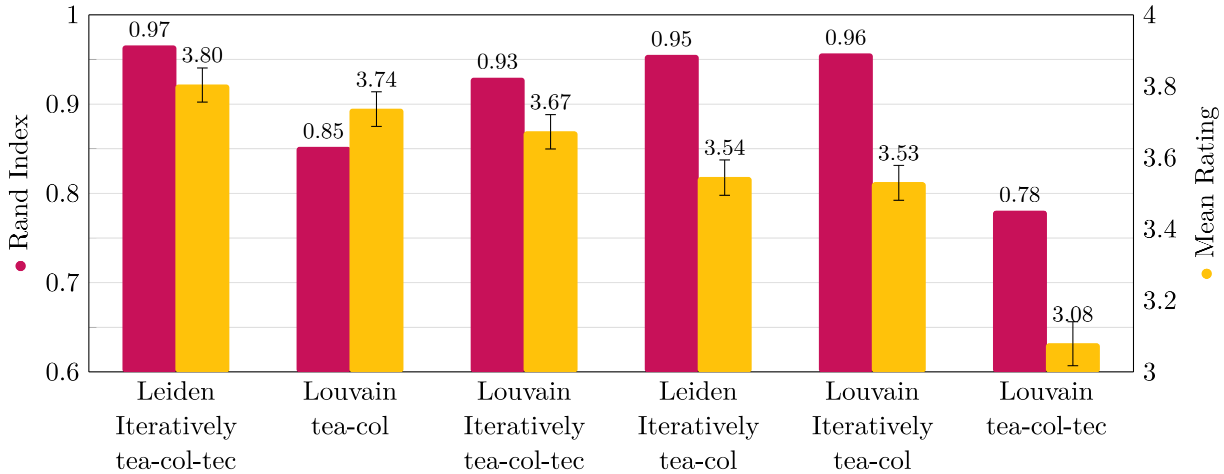

Figure 12 The Rand index and mean rating for every result set; the error bars show the 95% confidence interval

Result Set A	Result Set B	P-Value
Leiden Iteratively tea-col-tec	Louvain tea-col	0.052
Leiden Iteratively tea-col-tec	Louvain Iteratively tea-col-tec	0.000
Leiden Iteratively tea-col-tec	Leiden Iteratively tea-col	0.000
Leiden Iteratively tea-col-tec	Louvain Iteratively tea-col	0.000
Leiden Iteratively tea-col-tec	Louvain tea-col-tec	0.000
Louvain tea-col	Louvain Iteratively tea-col-tec	0.068
Louvain tea-col	Leiden Iteratively tea-col	0.000
Louvain tea-col	Louvain Iteratively tea-col	0.000
Louvain tea-col	Louvain tea-col-tec	0.000
Louvain Iteratively tea-col-tec	Leiden Iteratively tea-col	0.000
Louvain Iteratively tea-col-tec	Louvain Iteratively tea-col	0.000
Louvain Iteratively tea-col-tec	Louvain tea-col-tec	0.000
Leiden Iteratively tea-col	Louvain Iteratively tea-col	0.681
Leiden Iteratively tea-col	Louvain tea-col-tec	0.000
Louvain Iteratively tea-col	Louvain tea-col-tec	0.000

Table 10 The results of applying a Student's t-test to each pair of result sets

To give a statistical analysis of our findings, we applied the Student's t-test to the ratings of each pair of result sets, as detailed in Table 10. With a significance level of $\alpha = 0.01$, we find that almost all pairs of result sets have a statistically different mean rating from one another (p-value $< \alpha$). Interesting here is how statistically insignificant the two iterative approaches are when run on the small dataset, but when run on the larger dataset there is a clear statistical difference between the two, signifying that besides the technique used, the dataset strongly influences the results as well.

7 Conclusion

This study has demonstrated the effectiveness of community detection in extracting knowledge from the topology of linked data, supported by a series of experiments (Section 5) and a subsequent survey (Section 5.4) conducted on real-world users. The findings (Section 6) indicate that the quality of the results is influenced by the choice of community detection algorithm, whether the algorithm is run iteratively, and the aggregation method used.

The data aggregation methods presented in Section 4.2 transform heterogeneous linked data into a homogeneous graph, enabling the application of community detection algorithms on this data. This method allows for the effortless inclusion or exclusion of relevant or irrelevant data ensuring communities can be found that reflect the properties of the aggregation methods used. This allows for highly specific datasets to be created and used. Our findings indicate the choice of dataset drastically changes the performance of community algorithms used, with iterative approaches performing better on larger datasets and non-iterative approaches performing better on smaller more specific datasets.

Furthermore, a novel approach that iteratively runs community detection methods is proven to be an effective strategy for reducing community sizes without diminishing community quality, supported by the results of the survey on real-world users (Section 6.3). This effect is particularly evident in modularity-optimizing community detection algorithms such as the `Louvain` and `Leiden` algorithms (Section 6.1).

In conclusion, the iterative application of community detection algorithms on aggregated linked data offers a robust method for creating small, meaningful communities of actors within cultural heritage data, demonstrating significant potential for further research and application.

Future Research

This study has shown the effectiveness of applying graph theory to linked data to extract knowledge from its topology. Future research could explore several promising areas:

Weighted Data Adding weights to the data may prevent smaller datasets from being overshadowed in combined datasets. Additionally, by adjusting these weights to emphasize specific properties, researchers can more accurately discern which properties contribute to better results.

Additional Community Detection Algorithms Applying additional community detection algorithms could result in better results. The behaviour of the algorithms when run iteratively is quite erratic, therefore other community detection algorithms may respond in different ways than we have observed.

Cross-Disciplinary Applications This study has shown community detection algorithms are an effective way of extracting knowledge from linked cultural heritage data. Applying these techniques to linked data in other fields will validate their adaptability.

Iterative Algorithm Analysis The novel iterative approach used in this study showed that the `Leiden` algorithm reduces community sizes significantly more compared to the `Louvain` algorithm. Further research is needed to understand why these differences occur.

References

1 Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, Aug 2010. doi:10.1038/nature09182.

2 Sotirios Angelis and Konstantinos Kotis. Generating and exploiting semantically enriched, integrated, linked and open museum data. In Emmanouel Garoufallou and María-Antonia Ovalle-Perandones, editors, *Metadata and Semantic Research - MTSR 2020*, pages 367–379. Springer, 2021. doi:10.1007/978-3-030-71903-6_34.

3 Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, Oct 2008. doi:10.1088/1742-5468/2008/10/P10008.

4 Chris Dijkshoorn, Lora Aroyo, Guus Schreiber, Jan Wielemaker, and Lizzy Jongma. Using linked data to diversify search results a case study in cultural heritage. In Krzysztof Janowicz, Stefan Schlobach, Patrick Lambrix, and Eero Hyvönen, editors, *Knowledge Engineering and Knowledge Management - EKAW 2014*, pages 109–120. Springer, 2014. doi:10.1007/978-3-319-13704-9_9.

5 Chris Dijkshoorn, Lora Aroyo, Jacco van Ossenbruggen, and Guus Schreiber. Modeling cultural heritage data for online publication. *Applied Ontology*, 13(4):255–271, 2018. doi:10.3233/AO-180201.

6 Chris Dijkshoorn, Lizzy Jongma, Lora Aroyo, Jacco van Ossenbruggen, Guus Schreiber, Wesley ter Weele, and Jan Wielemaker. The Rijksmuseum collection as linked data. *Semantic Web*, 9(2):221–230, 2018. doi:10.3233/SW-170257.

7 C. Doulaverakis, Y. Kompatsiaris, and M.G. Strintzis. Ontology-based access to multimedia cultural heritage collections - the reach project. In *The International Conference on "Computer as a Tool" - EUROCON 2005*, volume 1, pages 151–154. IEEE, Nov 2005. doi:10.1109/EURCON.2005.1629881.

8 Daniel Edler, Anton Holmgren, and Martin Rosvall. The mapequation software package. https://mapequation.org, 2023.

9 Nicola Fanizzi and Claudia d'Amato. A hierarchical clustering method for semantic knowledge bases. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems - KES 2007*, pages 653–660. Springer, 2007. doi:10.1007/978-3-540-74829-8_80.

10 Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. doi:10.1016/j.physrep.2009.11.002.

11 Nuno Freire, Enno Meijers, Sjors de Valk, Julien A. Raemy, and Antoine Isaac. Metadata aggregation via linked data: Results of the Europeana common culture project. In Emmanouel Garoufallou and María-Antonia Ovalle-Perandones, editors, *Metadata and Semantic Research - MTSR 2020*, pages 383–394. Springer, 2021. doi:10.1007/978-3-030-71903-6_35.

12 Silvia Giannini. Rdf data clustering. In Witold Abramowicz, editor, *Business Information Systems Workshops - BIS 2013*, pages 220–231. Springer, 2013. doi:10.1007/978-3-642-41687-3_21.

13 Gunnar AAstrand Grimnes, Peter Edwards, and Alun Preece. Instance based clustering of semantic web resources. In Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *The Semantic Web: Research and Applications - ESWC 2008*, pages 303–317. Springer, 2008. doi:10.1007/978-3-540-68234-9_24.

14 Eero Hyvönen, Eetu Mäkelä, Mirva Salminen, Arttu Valo, Kim Viljanen, Samppa Saarela, Miikka Junnila, and Suvi Kettula. Museumfinland—Finnish museums on the semantic web. *Journal of Web Semantics*, 3(2):224–241, Oct 2005. doi:10.1016/j.websem.2005.05.008.

15 José-Lázaro Martínez-Rodríguez, Ivan Lopez-Arevalo, Ana B Rios-Alvarado, and Xiaoou Li. A brief comparison of community detection algorithms over semantic web data. In *International Semantic Web and Linked Open Data Workshop - ISW-LOD 2016*, pages 34–44, 2016. URL: `https://ceur-ws.org/Vol-1807/04_ISW-LOD2016_34_44.pdf`.

16 M. E. J. Newman. Analysis of weighted networks. *Physical Review E*, 70:056131, Nov 2004. `doi:10.1103/PhysRevE.70.056131`.

17 Mark Newman. *Networks*. Oxford University Press, Jul 2018. `doi:10.1093/oso/9780198805090.001.0001`.

18 William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. `doi:10.1080/01621459.1971.10482356`.

19 M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, Nov 2009. `doi:10.1140/epjst/e2010-01179-1`.

20 Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, Jan 2008. `doi:10.1073/pnas.0706851105`.

21 Guus Schreiber, Alia Amin, Mark van Assem, Victor de Boer, Lynda Hardman, Michiel Hildebrand, Laura Hollink, Zhisheng Huang, Janneke van Kersen, Marco de Niet, Borys Omelayenko, Jacco van Ossenbruggen, Ronny Siebes, Jos Taekema, Jan Wielemaker, and Bob Wielinga. Multimedian e-culture demonstrator. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora M. Aroyo, editors, *The Semantic Web - ISWC 2006*, pages 951–958. Springer, 2006. `doi:10.1007/11926078_70`.

22 Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(5233):1–12, 2019. `doi:10.1038/s41598-019-41695-z`.

23 Yiwen Wang, Natalia Stash, Lora Aroyo, Peter Gorgels, Lloyd Rutledge, and Guus Schreiber. Recommendations based on semantically enriched museum collections. *Journal of Web Semantics*, 6(4):283–290, Nov 2008. `doi:10.1016/j.websem.2008.09.002`.

A Linked Art Visualization

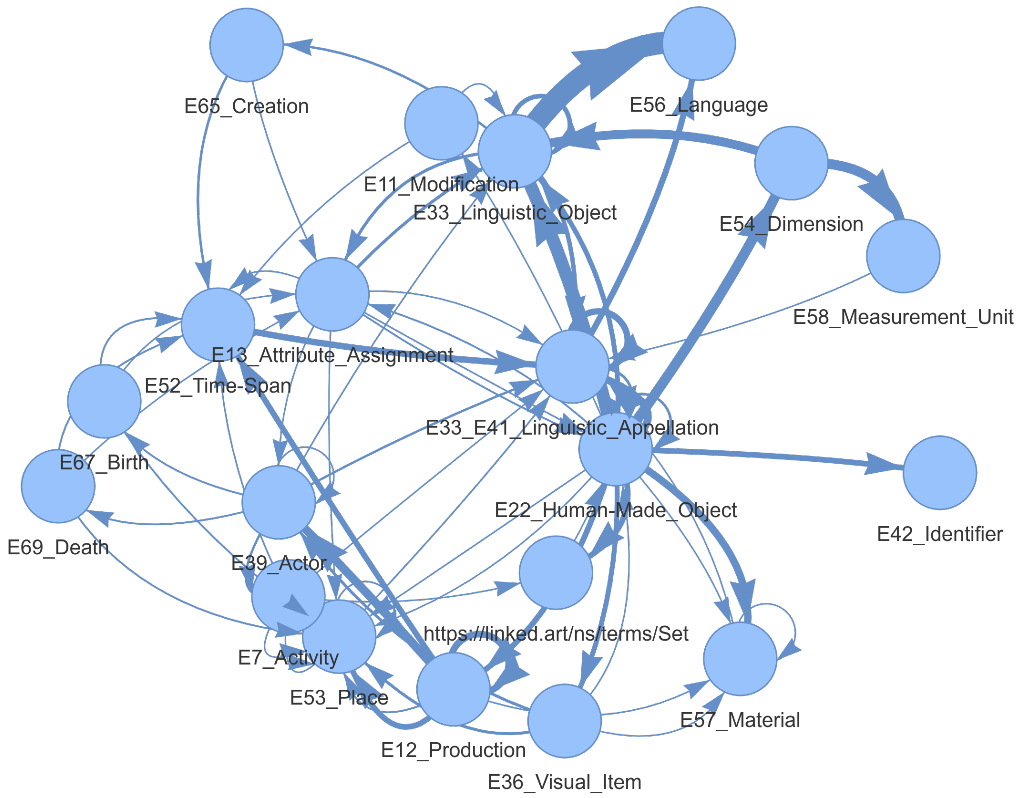

Figure 13 Graph showing how Linked Art entities are interconnected

B Datasets

Dataset	Diameter	APL	Transitivity	# CCs	# nodes	# edges
teacher-student	32	11.04	0.0231777	316	2 927	3 274
collaboration	9	2.91	0.0264787	117	7 982	50 7782
technique	4	1.84	0.7667573	1	17 972	63 786 788
location	5	1.94	0.7424659	58	15 691	20 817 814
teacher-student-collaboration	10	3.06	0.0271253	149	8 731	511 056
teacher-student-technique	9	1.88	0.7667262	15	18 356	63 790 062
teacher-student-location	8	2.02	0.7423577	86	16 365	20 821 088
collaboration-technique	5	1.87	0.7662950	6	18 574	64 294 570
collaboration-location	7	2.01	0.7413177	57	16 728	21 325 596
technique-location	4	1.82	0.6499929	5	19 323	84 604 602
teacher-student-collaboration-technique	6	1.88	0.7662660	11	18 715	64 297 844
teacher-student-collaboration-location	7	2.05	0.7412147	67	17 080	21 328 870
teacher-student-technique-location	6	1.84	0.6499798	12	19 544	84 607 876
collaboration-technique-location	5	1.83	0.6498781	5	19 568	85 112 384
teacher-student-collaboration-technique-location	6	1.84	0.6498653	8	19 677	85 115 658

Table 11 Graph metrics of each combined dataset
# CCs = the number of connected components in the graph
APL = the average path length across the graph

C Results

Table 12 The full results for every combination of technique and dataset.

Algorithm	Short Dataset	# Actors	# Comms	Mean Comm Size	Max Comm Size	5th %	95th %	Modularity	Codelength	Rand Index	Validation Set Overlap
Infomap	col	7 982	543	14.70	915	2	42	0.49	8.30	0.93	3 557
Infomap	col-loc	16 728	362	46.21	4 713	2	108	0.37	12.21	0.79	5 297
Infomap	col-tec	18 574	57	325.86	4 292	2	2 091	0.51	12.83	0.81	5 199
Infomap	col-tec-loc	19 568	61	320.79	12 616	2	652	0.30	13.36	0.40	5 567
Infomap	loc	15 691	334	46.98	4 720	2	88	0.37	12.21	0.78	4 956
Infomap	tea	2 927	317	9.23	1 365	2	5	0.59	8.43	0.70	1 615
Infomap	tea-col	8 731	600	14.55	820	2	36	0.49	8.30	0.94	3 798
Infomap	tea-col-loc	17 080	440	38.82	5 028	2	81	0.37	12.22	0.79	5 404
Infomap	tea-col-tec	18 715	86	217.62	8 991	2	754	0.42	12.84	0.64	5 287
Infomap	tea-col-tec-loc	19 677	84	234.25	12 771	2	304	0.30	13.37	0.40	5 633
Infomap	tea-loc	16 365	424	38.60	5 507	2	77	0.35	12.22	0.76	5 240
Infomap	tea-tec	18 356	93	197.38	4 350	2	910	0.51	12.82	0.81	5 094
Infomap	tea-tec-loc	19 544	77	253.82	12 862	2	467	0.30	13.36	0.37	5 560
Infomap	tec	17 972	42	427.90	8 169	3	1 560	0.43	12.83	0.66	4 846
Infomap	tec-loc	19 323	54	357.83	12 473	2	674	0.30	13.37	0.40	5 417
Infomap Iteratively	col	7 982	1 094	7.30	915	2	15	0.38	8.61	0.95	3 557
Infomap Iteratively	col-loc	16 728	593	28.21	4 630	2	60	0.37	12.21	0.80	5 297
Infomap Iteratively	col-tec	18 574	125	148.59	4 292	2	490	0.50	12.84	0.82	5 199
Infomap Iteratively	col-tec-loc	19 568	84	232.95	12 613	2	402	0.30	13.36	0.40	5 567
Infomap Iteratively	loc	15 691	551	28.48	4 346	2	64	0.37	12.20	0.80	4 956
Infomap Iteratively	tea	2 927	716	4.09	29	2	9	0.71	5.29	0.93	1 615

Continued on next page

Table 12 Continued from previous page

Algorithm	Short Dataset	# Actors	# Comms	Mean Comm Size	Max Comm Size	5th %	95th %	Modularity	Codelength	Rand Index	Validation Set Overlap
Infomap Iteratively	tea-col	8 731	1 306	6.69	818	2	13	0.36	8.72	0.96	3 798
Infomap Iteratively	tea-col-loc	17 080	675	25.30	4 722	2	55	0.37	12.21	0.81	5 404
Infomap Iteratively	tea-col-tec	18 715	162	115.52	4 342	2	417	0.50	12.84	0.82	5 287
Infomap Iteratively	tea-col-tec-loc	19 677	106	185.63	12 771	2	293	0.30	13.36	0.40	5 633
Infomap Iteratively	tea-loc	16 365	645	25.37	4 597	2	59	0.37	12.20	0.80	5 240
Infomap Iteratively	tea-tec	18 356	136	134.97	4 350	2	477	0.51	12.83	0.82	5 094
Infomap Iteratively	tea-tec-loc	19 544	104	187.92	12 777	2	290	0.30	13.36	0.39	5 560
Infomap Iteratively	tec	17 972	89	201.93	8 169	2	648	0.42	12.83	0.67	4 846
Infomap Iteratively	tec-loc	19 323	81	238.56	12 470	2	360	0.30	13.36	0.40	5 417
Leiden	col	7 982	193	41.36	2 431	2	238	0.49	7.80	0.86	3 557
Leiden	col-loc	16 728	67	249.67	4 489	2	2 158	0.44	12.44	0.83	5 297
Leiden	col-tec	18 574	12	1 547.83	5 452	2	4 785	0.51	12.86	0.77	5 199
Leiden	col-tec-loc	19 568	9	2 174.22	7 092	2	5 996	0.41	13.45	0.72	5 567
Leiden	loc	15 691	70	224.16	3 928	2	1 631	0.44	12.40	0.83	4 956
Leiden	tea	2 927	347	8.44	186	2	60	0.87	6.09	0.91	1 615
Leiden	tea-col	8 731	220	39.69	2 292	2	248	0.49	7.80	0.87	3 798
Leiden	tea-col-loc	17 080	76	224.74	4 034	2	1 886	0.44	12.43	0.82	5 404
Leiden	tea-col-tec	18 715	19	985.00	5 453	2	4 729	0.51	12.86	0.77	5 287
Leiden	tea-col-tec-loc	19 677	13	1 513.62	6 457	2	5 227	0.41	13.45	0.73	5 633
Leiden	tea-loc	16 365	103	158.88	3 182	2	1 185	0.44	12.42	0.84	5 240
Leiden	tea-tec	18 356	39	470.67	5 423	2	4 110	0.51	12.85	0.77	5 094
Leiden	tea-tec-loc	19 544	24	814.33	6 659	2	4 296	0.41	13.45	0.73	5 560
Leiden	tec	17 972	5	3 594.40	5 417	2 199	5 169	0.51	12.85	0.77	4 846
Leiden	tec-loc	19 323	9	2 147.00	6 307	2	5 525	0.41	13.45	0.73	5 417
Leiden Iteratively	col	7 982	1 442	5.54	1 048	2	10	0.22	8.22	0.95	3 557

Continued on next page

Table 12 Continued from previous page

Algorithm	Short Dataset	# Actors	# Comms	Mean Comm Size	Max Comm Size	5th %	95th %	Modularity	Codelength	Rand Index	Validation Set Overlap
Leiden Iteratively	col-loc	16 728	1 573	10.63	1 094	2	23	0.14	13.88	0.97	5 297
Leiden Iteratively	col-tec	18 574	1 034	17.96	2 550	2	36	0.15	14.58	0.96	5 199
Leiden Iteratively	col-tec-loc	19 568	1 540	12.71	906	2	35	0.02	15.47	0.97	5 567
Leiden Iteratively	loc	15 691	970	16.18	1 282	2	50	0.17	13.68	0.96	4 956
Leiden Iteratively	tea	2 927	741	3.95	13	2	9	0.68	5.50	0.93	1 615
Leiden Iteratively	tea-col	8 731	1 621	5.39	1 002	2	10	0.21	8.26	0.95	3 798
Leiden Iteratively	tea-col-loc	17 080	1 642	10.40	1 118	2	24	0.14	13.88	0.97	5 404
Leiden Iteratively	tea-col-tec	18 715	1 050	17.82	2 551	2	39	0.15	14.58	0.97	5 287
Leiden Iteratively	tea-col-tec-loc	19 677	1 571	12.53	906	2	34	0.02	15.48	0.97	5 633
Leiden Iteratively	tea-loc	16 365	1 124	14.56	1 275	2	37	0.17	13.70	0.96	5 240
Leiden Iteratively	tea-tec	18 356	662	27.73	2 661	2	62	0.17	14.48	0.96	5 094
Leiden Iteratively	tea-tec-loc	19 544	1 399	13.97	913	3	38	0.02	15.48	0.97	5 560
Leiden Iteratively	tec	17 972	602	29.85	2 660	3	71	0.17	14.48	0.96	4 846
Leiden Iteratively	tec-loc	19 323	1 382	13.98	887	3	38	0.02	15.48	0.97	5 417
Louvain	col	7 982	134	59.57	2 223	2	408	0.55	8.65	0.84	3 557
Louvain	col-loc	16 728	67	249.67	3 279	2	2 258	0.44	12.46	0.85	5 297
Louvain	col-tec	18 574	12	1 547.83	4 760	2	4 520	0.51	12.93	0.77	5 199
Louvain	col-tec-loc	19 568	9	2 174.22	4 639	2	4 539	0.41	13.47	0.77	5 567
Louvain	loc	15 691	69	227.41	2 586	2	2 060	0.44	12.45	0.84	4 956
Louvain	tea	2 927	347	8.44	228	2	56	0.86	6.15	0.91	1 615
Louvain	tea-col	8 731	175	49.89	2 225	2	226	0.56	8.72	0.85	3 798
Louvain	tea-col-loc	17 080	79	216.20	3 021	2	2 098	0.44	12.45	0.83	5 404
Louvain	tea-col-tec	18 715	21	891.19	4 733	2	4 200	0.51	12.88	0.78	5 287
Louvain	tea-col-tec-loc	19 677	13	1 513.62	6 900	2	5 462	0.41	13.47	0.73	5 633
Louvain	tea-loc	16 365	99	165.30	2 968	2	1 408	0.44	12.45	0.85	5 240

Continued on next page

Table 12 Continued from previous page

Algorithm	Short Dataset	# Actors	# Comms	Mean Comm Size	Max Comm Size	5th %	95th %	Modularity	Codelength	Rand Index	Validation Set Overlap
Louvain	tea-tec	18 356	40	458.90	4 938	2	3 650	0.51	12.87	0.77	5 094
Louvain	tea-tec-loc	19 544	19	1 028.63	6 261	2	4 629	0.41	13.48	0.73	5 560
Louvain	tec	17 972	6	2 995.33	4 621	1 683	4 477	0.52	12.86	0.78	4 846
Louvain	tec-loc	19 323	9	2 147.00	6 400	2	5 583	0.41	13.48	0.73	5 417
Louvain Iteratively	col	7 982	1 474	5.42	991	2	10	0.23	9.79	0.95	3 557
Louvain Iteratively	col-loc	16 728	1 190	14.06	2 101	2	28	0.32	12.94	0.92	5 297
Louvain Iteratively	col-tec	18 574	464	40.03	3 300	2	95	0.32	13.64	0.92	5 199
Louvain Iteratively	col-tec-loc	19 568	420	46.59	3 084	2	155	0.23	14.31	0.93	5 567
Louvain Iteratively	loc	15 691	724	21.67	2 268	2	53	0.36	12.72	0.90	4 956
Louvain Iteratively	tea	2 927	738	3.97	13	2	9	0.68	5.49	0.93	1 615
Louvain Iteratively	tea-col	8 731	1 683	5.19	921	2	9	0.23	9.91	0.96	3 798
Louvain Iteratively	tea-col-loc	17 080	1 399	12.21	2 007	2	22	0.30	13.04	0.93	5 404
Louvain Iteratively	tea-col-tec	18 715	570	32.83	3 367	2	81	0.31	13.69	0.93	5 287
Louvain Iteratively	tea-col-tec-loc	19 677	514	38.28	3 023	2	113	0.22	14.34	0.94	5 633
Louvain Iteratively	tea-loc	16 365	1 160	14.11	1 988	2	27	0.32	12.94	0.93	5 240
Louvain Iteratively	tea-tec	18 356	445	41.25	3 297	2	113	0.30	13.71	0.92	5 094
Louvain Iteratively	tea-tec-loc	19 544	439	44.52	3 046	2	166	0.22	14.33	0.94	5 560
Louvain Iteratively	tec	17 972	185	97.15	3 330	2	535	0.32	13.62	0.91	4 846
Louvain Iteratively	tec-loc	19 323	286	67.56	2 918	2	292	0.21	14.35	0.93	5 417

Concluded

D Survey

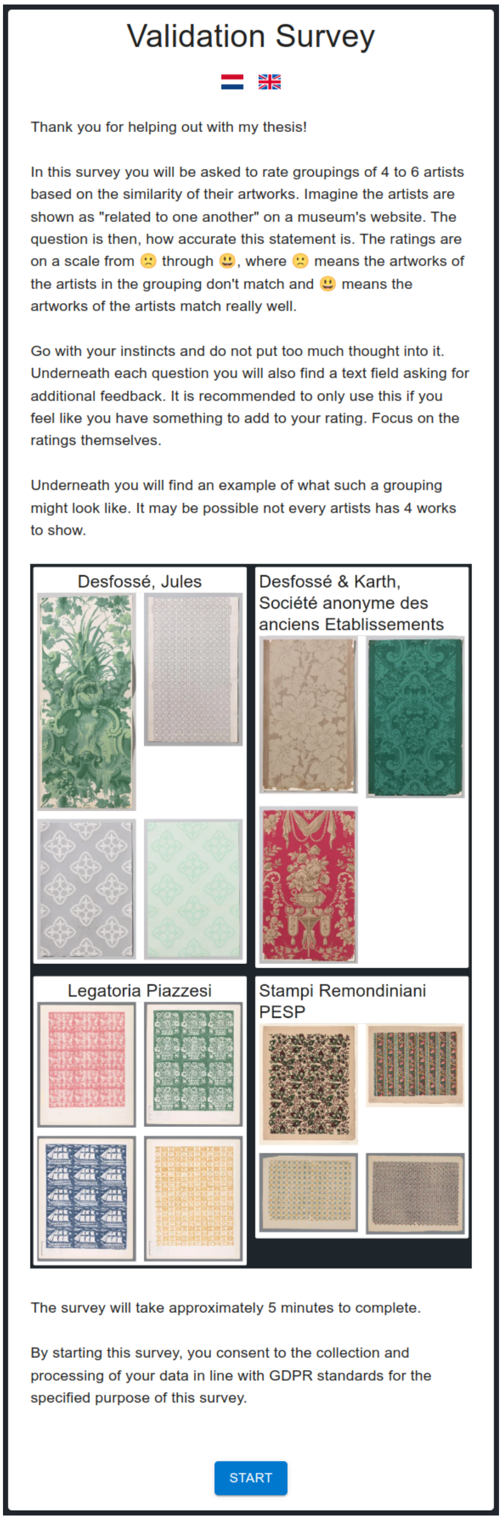

Figure 14 A screenshot of the survey website

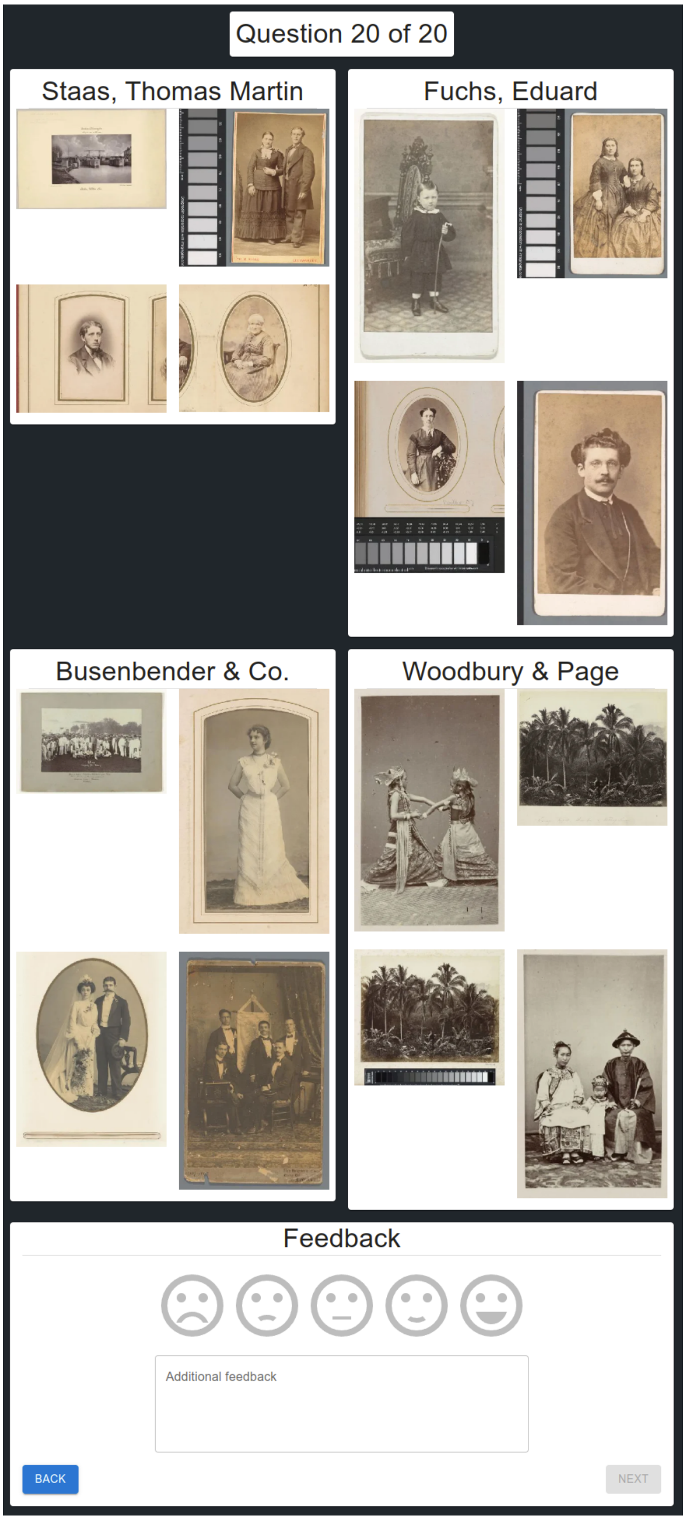

Figure 15 A screenshot of a question on the survey website