

Entity Matching in Digital Humanities Knowledge Graphs

Juriaan Baas, Mehdi M. Dastani and Ad J. Feelders

Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, the Netherlands

Abstract

We propose a method for entity matching that takes into account the characteristic complex properties of decentralized cultural heritage data sources, where multiple data sources may contain duplicates within and between sources. We apply the proposed method to historical data from the Amsterdam City Archives using several clustering algorithms and evaluate the results against a partial ground truth. We also evaluate our method on a semi-synthetic data set for which we have a complete ground truth. The results show that the proposed method for entity matching performs well and is able to handle the complex properties of historical data sources.

Keywords

entity matching, historical data, knowledge graphs

1. Introduction

The Golden Agents project¹ is developing a sustainable research infrastructure to study relations and interactions between producers and consumers of creative goods during the Dutch Golden Age. This requires integration of various heterogeneous datasets by using, among other things, semantic web solutions. Due to the independent nature of the institutions that govern these datasets, their respective knowledge graphs often use different URIs to refer to the same real-world objects. To access the information that is present for an object in a set of knowledge graphs, automated methods for identifying and linking duplicate entities within and among knowledge graphs are required. Linking duplicate entities in the semantic web literature is also known as entity matching.

In this work we focus on identifying duplicate entities in KGs that contain historical data. We discuss various characteristic and complex properties of such data and argue that existing entity matching approaches often use data that do not involve one or more of these complicating properties. In particular, we aim to identify duplicate entities in data sources that have (some of) the following properties:

1. The KGs contain internal duplicates, these can be either due to error (a record was mistakenly duplicated) or not (the same person married twice).
2. There exist duplicate entities that occur between different KGs.

CHR 2021: Computational Humanities Research Conference, November 17–19, 2021, Amsterdam, The Netherlands

✉ j.baas@uu.nl (J. Baas); m.m.dastani@uu.nl (M.M. Dastani); a.j.feelders@uu.nl (A.J. Feelders)

🌐 <https://github.com/jurian> (J. Baas)

🆔 0000-0001-8689-8824 (J. Baas); 0000-0002-4641-4087 (M.M. Dastani); 0000-0003-4525-1949 (A.J. Feelders)

© 2021 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://www.goldenagents.org>

3. Entities may have missing attributes.
4. Attributes, such as dates, can be approximate.
5. Attribute values can have errors, these can be in the original historical version or introduced over time when the data is copied by hand, digitized, moved, etc. This is also a potential source of duplicates within KGs.
6. There is no standard for names, e.g. there is no official way to spell a certain name. The use of shorthands for patronyms is common, e.g. the last names 'Jans', 'Jansz', 'Janz', 'Janssen' can all refer to the same person.
7. Different KGs use different attributes to describe the same type of entities, e.g. persons. These attributes can be highly correlated but not have identical meanings. For example, one KG can use birth dates and another one baptism dates. We call these high correlated attributes *proxy variables*.
8. There are many one-to-n and n-to-n relationships in the data, this makes the use of a tabular structure very difficult. Our method is capable of handling these kinds of relationships naturally.

The final output of our method is a set of clusters, each claimed to correspond to one and the same real life object. Furthermore, we assume that these clusters are themselves used in downstream tasks. The application of transitive closure can sometimes yield very poor performance if this is not taken into account during the clustering stage.

Our contribution is the proposal of a comprehensive method that starts with a set of KGs with the above mentioned characteristic properties. All relevant entities, such as all persons, in the merged KGs are then embedded based on their local context. This context can also be influenced by weights that are set in the configuration file to serve a specific task, such as entity matching (e.g. names are more important than dates). When all entities are embedded, we make use of an approximate nearest neighbor algorithm to efficiently find pairs of entities that are likely duplicates. We call the pairs that pass a similarity threshold the candidate pairs. Since ground truth is very limited, our method is unsupervised. Therefore, these candidate pairs are then further refined with the use of unsupervised clustering algorithms, resulting in a set of entity clusters, each predicted to represent a distinct object.

We experiment with a real life collection of KGs in the cultural heritage domain for which we have a partial ground truth (gold standard), and a semi-synthetic KG that acts as an analog that mirrors the graph structure and noise of the real-life data. We use the semi-synthetic KG because it is much larger, and moreover since we have introduced duplicates ourselves, we have a complete ground truth. For each case, we then embed all relevant nodes in a 50-dimensional Euclidean space. We chose reasonable values from Cochez et al. [11] for the number of dimensions and the BCA hyperparameters to reduce the number of parameters that need tuning. The settings used in the experiments can all be found in the repositories published in section 6.0.1. Finally we apply several clustering algorithms, both heuristic and exact, and compare their results to the transitive closure of the connected components formed by the candidate pairs. We show that a correctly tuned embedding can achieve good performance, even when no advanced clustering is used. However, in practice there is often no ground truth available and therefore it is difficult to choose the similarity threshold that yields these good results. To address this, we show that heuristic as well as exact clustering algorithms can be

used to repair clusters that are too large and achieve high performance for a much larger range of similarity thresholds.

The structure of this paper is as follows. First, we present other work related to our method in section 2. Then in section 3, we give an overview of our methodology. Afterwards we present the experimental setup in section 4, where we discuss the data and clustering algorithms. Next, section 5 provides an analysis of the results of the experiments. Finally, we conclude the paper with some future research directions.

2. Related Work

To position our work we make use of the taxonomy defined in the survey paper by Christophides et al. [10]. Our method can best be placed in both the *Multi-Source Entity Resolution* (ER) (there exist duplicates between data sources), and *Dirty ER* (data sources themselves contain duplicates) categories. We therefore categorize our method as *Multi-Dirty Source ER*.

Much work has been done on the problem of identifying entities in the data that refer to the same real-world object, often using related terms such as: entity linking, entity disambiguation, entity resolution, entity deduplication, entity alignment and record linkage. Many of these terms are used in different circumstances or for related problems, for example, record linkage is usually used in the context of matching records in one dataset with records in other datasets. These works can be rule-based [24, 31], make use of word embeddings of tokens that appear in the descriptions of entities [13, 34], or focus on scalability [23, 29, 30]. Some of these works make use of structured (tabular) data and exploit extra information such as duplicate free sources. Furthermore, it is often assumed that entities can be neatly described using a fixed number of attributes, and that all entities either use (different names for) the same attributes, or use different attributes that can be easily mapped.

Most relevant to our setting are methods that make use of representation learning techniques. In this case the term *entity alignment* is most often used. The key idea is to learn embeddings of KGs, such that entities with similar neighbor structures in the KG have a close representation in the embedding space. These techniques have their origins (for graphs in general) in Node2Vec [15] and Deepwalk [26], and (for KGs) in TransE [7] and its many extensions, such as [12, 32, 33]. Furthermore, several techniques exist for embedding nodes in the case of mapping two KGs that are in different languages, e.g. [8, 9]. In that case one can exploit the fact that there can be at most two duplicates entities per real world object.

In light of other work on cultural heritage data sources, Raad et al. [27] summarize their efforts to create a certificate linking method for Dutch civil certificates from the Zeeland region, based on efficient string similarity computations. Furthermore, they propose a contextual identity link [28], as they observe that the `owl:sameAs` link is often misused. They note that the notion of identity can change under different contexts. For example, two pharmaceuticals may be judged as equivalent when their names match under some conditions, while under other conditions their chemical structure needs to be identical as well. Their solution is an algorithm which detects the specific global contexts in which two entities are identical. Similarly, Idrissou et al. [20, 19] have proposed a contextual identity link based on the use of related entities to construct evidence for likely duplicate pairs. An example of evidence is that two entities may co-occur in multiple records under similar names. Their method requires the user to specify beforehand what is considered evidence and how entities should be matched. Koho et al. [21] reconcile military historical persons in three registers, and use similarity measures between

attributes in both a deterministic rule-based method, based on a pre-defined handcrafted formula, and a probabilistic method that makes use of supervised learning. In contrast to our work, only precision is reported, as an exact recall cannot be calculated with a small manually generated partial ground truth. Hendriks et al. [18] use data from the Amsterdam Notary Archives and Dutch East India Company (VOC) and perform both named entity recognition and record linkage with the help of supervised learning, where we use unsupervised learning.

To create the embedding, we build on the work of Cochez et al. [11] and Baas et al. [3], who both use the Bookmark Coloring Algorithm (BCA) [5] to create a co-occurrence matrix from the knowledge graph, and GloVe [25] to learn the embedding from the co-occurrence matrix. Additionally, Baas et al. [2] introduce a mechanism for merging multiple KGs based on user set similarity rules and focus on graph traversal strategies to create entity contexts. Afterwards they apply supervised learning to create a classifier which is able to identify duplicate pairs of entities. In more recent work [2], they introduce the cluster editing (also called correlation clustering) algorithm to create clusters using the aforementioned classifier to generate input weights.

Important differences with this work are that 1. We forego the supervised approach and instead use an unsupervised method, as labeled examples are often very rare (or non-existent) in historical research settings. 2. We apply a form of blocking and filtering to create clusters without having to compute all pairwise similarities. 3. We modified GloVe to be more efficient on large KGs, as explained in sections 3.2, and 3.3. 4. Since exact solution of the cluster editing problem is computationally infeasible for large instances, we also applied three heuristic clustering algorithms. 5. We use an additional dataset (DBLP) to show that our method works in multiple settings.

3. Methodology

We will first briefly formalize the entity matching objective in our setting. Given a starting set of knowledge graphs $\{KG_1, \dots, KG_n\}$ with $KG_i = (V_i, E_i)$, our objective is to find the subset of pairs of duplicate entities \mathcal{L} , where

$$\begin{aligned} V &= \bigcup_{i=1}^n V_i, \\ P &= V \times V, \text{ and} \\ \mathcal{L} &\subseteq P. \end{aligned}$$

It is understood in the field of entity matching that the quadratic growth of P presents a major problem. Therefore most methods employ some sort of blocking and/or filtering methods to reduce the number of pairs that have to be evaluated. We employ the embedding for this purpose, as detailed in section 3.4. The workflow of our method is illustrated in figure 1, and will be described in the rest of this section.

3.1. Merging KGs

We start with a set of KGs, as seen in panel *a* of Figure 1. In our setting, the only way the graphs can be compared and connected is through identical or similar values in their literal nodes. Therefore, we perform two steps:

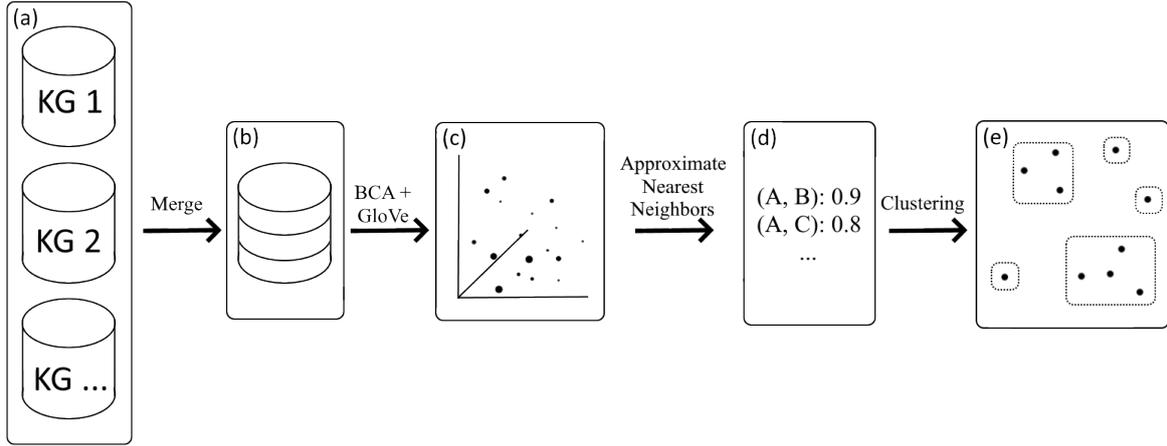


Figure 1: An overview of the entire process. First, a given set of KGs (a) is merged into a single graph (b), then the nodes in this merged graph are embedded (c) and a set of candidate pairs (d) is computed using approximate nearest neighbor search and a similarity threshold. Finally, the candidate pairs form connected components, which are then clustered (e).

1. All literal nodes with identical values are merged based on their predicate. Merging on solely the value of a literal node will cause ambiguity as to what the value of that literal node represents.
2. Although graphs can be partly connected after applying the first step, we still face the problem of noise present in the values of literal nodes, and, often predicates associated with literal nodes may only occur in one of the graphs. However, these predicates can still be related if one can act as a proxy variable for the other, e.g. one graph contains the *bornOn* predicate and another the *baptisedOn* predicate. We connect such nodes if they satisfy a similarity criterion, such as a preferred distance between certain dates. This technique can be used to deal with noise as well, e.g. to connect literal nodes that represent similar names by using, for instance, Jaro-Winkler string similarity.

The resulting merged graph, panel *b* in Figure 1, does not necessarily adhere to the RDF standards, as we may have added edges between literal nodes. From this point on, we do not treat the merged graph as a KG, but instead as an undirected weighted graph $G = (V, E)$, where the weights are either predefined (for predicates) or are created (for literal node similarities) in step 2 above.

3.2. Generating Entity Contexts

We use an adaptation of the Bookmark Coloring Algorithm [5] (BCA) to generate a context for each node in a graph. The general idea of BCA is to consider the context of a node in a graph as an approximation of the personalized PageRank for that node, i.e., a set of probabilities associated with nodes in the graph. A useful analogy is imagining dropping a unit amount of paint on a given starting node. A fraction α of the paint remains on the node and the fraction $1 - \alpha$ is distributed among neighboring nodes. When the amount of paint falls under ϵ the paint is no longer distributed. This means that even in the case of loops, the algorithm will

eventually terminate after running out of paint. The value for ϵ can be tuned to get a smaller or larger neighborhood. The fraction of paint which continues to flow can be adjusted with α .

Instead of uniformly distributing paint over neighboring nodes, as standard BCA does, the paint is distributed relative to the weight of edges. The weights on edges between literal nodes are computed from the similarity of their values. All remaining weights are set in the configuration file by the domain expert, or can be omitted, in which case they all equal 1.

As mentioned before, we treat the edges in the merged graph as undirected. This is because the directed structure of the graph can curb the ‘flow of paint’, especially when information about duplicate entities is dominated by literal nodes, which only have incoming edges by default. This will cause otherwise related entities to have no or very small overlapping context, something we wish to avoid.

Finally, we have increased the efficiency of calculating the entity contexts in two ways. First, we perform early stopping by not processing nodes for which we know there will not be enough paint to continue. This behaviour also introduces a limit on the minimum amount of paint present on any given node reached by BCA, which increases numerical stability in the embedding process. Secondly, we observe that we do not need to perform the BCA algorithm for every single node. Instead, only the nodes we wish to embed have their context calculated, where every node can potentially appear in a context, including those that are not embedded. We call the nodes that have their context calculated focus nodes. This modification greatly reduces the running time of the context generation stage, as usually only a fraction of the nodes in the graph need to be embedded, e.g. only nodes of type person. Furthermore, by still including each node in the potential context, we do not impair the context by removing important co-occurrence information. This second modification does require us to modify the GloVe algorithm slightly, detailed below.

3.3. Embedding nodes in the merged graph

GloVe [25] has been established as an effective method to use a text corpus for embedding words in a vector space. We adapt it to embed nodes in a graph instead. GloVe takes as input a co-occurrence matrix X , where in our case X_{ij} signifies the amount of paint of BCA for node j , when starting from node i . When BCA has not reached node j starting from node i , then $X_{ij} = 0$. Calculating node vectors can be achieved by minimizing the cost function

$$\text{cost} = \sum_i^N \sum_j^{|V|} f(X_{ij}) (b_i + \bar{b}_j + w_i^T \bar{w}_j - \log X_{ij})^2 \quad (1)$$

where N is the number of focus nodes we wish to embed, w_i is the vector for node i as a focus node, and \bar{w}_j is the vector for node j as a context node. Likewise, b_i and \bar{b}_j are the bias terms of node i as a focus node, and node j as a context node respectively.

In the original specification of GloVe, X is assumed to be a square matrix. However, as mentioned in section 3.2, we do not perform BCA for every node in the merged graph. Therefore X is no longer a square matrix, instead, there are N rows and $|V|$ columns. This also means that there are N vectors w , and $|V|$ vectors \bar{w} , the same goes for the bias terms. When the change between iterations in the cost function 1 falls below a user set threshold, we regard the algorithm to be converged. The final embedded vector for each focus node i is the average between vectors w_i and \bar{w}_i . We use the term embedding for the set of all embedded vectors, shown in panel *c* in Figure 1.

3.4. Efficiently selecting likely duplicate pairs

As stated in the entity matching objective, we wish to determine the subset $\mathcal{L} \subseteq P$ that contains the duplicate pairs. However, the vast majority of pairs in P do not link duplicate entities, so duplicate pairs are very rare [1]. We reduce the degree of rarity by creating two subsets $P_1 \subseteq P$ and $P_2 = P - P_1$, where P_1 is expected to be much smaller in size and to have a much higher proportion of duplicate pairs. We construct P_1 by, for every embedded entity, taking its approximate k nearest neighbors, where from experience we have learned that $k = 2$ is a good value for a range of cluster sizes. This effectively acts as a blocking mechanism, as most pairs in P are never considered. The problem of approximate nearest neighbor (ANN) search has been well researched [6, 16] and calculating the *approximate* k nearest neighbors for all entities can be achieved in much less than quadratic time, depending on the level of approximation one is willing to tolerate. We note that when a near neighbor j is omitted in the approximate result for node i , i.e. we miss the pair (i, j) , then the pair (j, i) can still be found when considering the approximate neighbors for node j . This reduces the impact of the errors returned in approximate search in our method. Next, all pairs in P_1 are treated as unordered, i.e. pairs (i, j) and (j, i) are merged. Finally, we calculate the cosine similarity s_{ij} for all pairs in P_1 , and name all pairs where the similarity exceeds a threshold θ the *candidate pairs*, shown in panel *d* in Figure 1.

3.5. Clustering

When the candidate pairs are treated as edges in an undirected graph, the resulting graph typically consists of a number of connected components. Since only pairs of entities in the same connected component are regarded as potential duplicates, further processing is performed separately for each connected component C . First, C is modified by adding a weighted edge for each possible pair of entities $(i, j) \in C$. We define the weight for the edge between entities i and j as $w_{ij} = s_{ij} - \theta$. Note that w_{ij} can be negative. We call the resulting weighted complete graph C' , which is then used as input for a clustering algorithm. Note that for high values of $\theta \approx 1$, there are fewer candidate pairs (and thus smaller connected components) than for lower values of θ . We elaborate on the specifics of each clustering algorithm in section 4.3.

4. Experimental Setup

For our experiments we have used two datasets, one is a set of KGs containing real historical data from the cultural heritage domain, the other is a semi-synthetic KG designed to have similar structural properties as the merge of the KGs from the cultural heritage domain. The main reason to design and use a semi-synthetic KG is to have a complete ground truth to validate the performance of the proposed approach. As mentioned in the introduction section, the KGs from the cultural heritage domain in our project have only a partial ground truth, obtained with manual validation by domain experts [22]. In this section, we first explain the used datasets, i.e. SAA and DBLP. We then explain how the combined KGs of the SAA, and the DBLP KG are then processed according to the methodology explained in section 3.5. Without getting into the technical details of the application of our methodology in this experimental setup, we only note that these knowledge graphs are embedded into a separate 50-dimensional space. Finally, we explain the clustering methods used are explained in the rest of this section.

4.1. City Archives of Amsterdam

The City Archives of Amsterdam² (in Dutch *Stadsarchief Amsterdam*, abbreviated SAA) is a collection of registers, acts and books from the 16th century up to modern times. The original data are in the form of handwritten books that have been scanned and digitised by hand. Often more information is stored in the original form than was transcribed. Fully digitising all information is an ongoing process, performed mostly by volunteers. For this project we made use of a subset collected from three different registers: Burial, Prenuptial Marriage and Baptism. The burial register does not describe who was buried where, but simply records the act of someone declaring a deceased person. To this end, it mentions the date and place of declaration and references two persons, one of whom is dead. Sadly, it does not tell us which one of the two has died. The prenuptial marriage records tell us the church, religion and names of those who are planning to get married. It also mentions names of persons involved in previous marriages if applicable. The baptism register mentions the place and date of where a child was baptised. It does not tell us the name of the child, only the names of the father and mother. Lastly the above records were combined with a subset from the Ecartico³ data set, which is a comprehensive collection of biographical data from more well-known people from the Dutch golden age. Figure 2 shows the graph representations of a record in each register. Note that these records can be linked to each other by sharing a literal node, in this case a name or a date field, where the name of an individual is often written in different ways and many dates are approximate. For our experiments we use a subset containing 12,517 (non-unique) persons, and a partial ground truth of 1073 clusters.

4.2. Semi-synthetic DBLP-based KG

Since we have only a partial ground truth available for the SAA dataset, and to validate the performance of our approach, we have created a KG based on a 2017 RDF release⁴ of DBLP,⁵ a well-known dataset among computer scientists, containing publication and author information. This version of DBLP has already been disambiguated, that is, different persons with the same name have unique URIs. We have reversed this by first taking a random sample of persons and then creating new anonymous URIs for each author listed per publication. In total there are 76,397 new URIs created for disambiguation into 51,515 clusters. These anonymous author nodes are then assigned their original name, with the further introduction of noise. Every character in each name instance has a 1% chance of deletions, random character swaps and replacing a character with a random character. Certain characters (é becomes e) have a 25% chance of alteration, and middle names are shortened to their initials with a 50% probability. There are never more than 3 alterations per name instance. We chose these particular percentages to create enough noise such that names can have multiple different, but not unrecognisable, versions, as is the case in the SAA dataset. This type of attribute error, if large enough, can decrease precision as identical entities will have very dissimilar attributes. The final result is a KG with publication nodes, each with a title attribute and one or more authors, each of which has a name attribute. The clustering objective is to group together all author URIs that represent the same real life person, where we only use noisy

²<https://archieff.amsterdam/indexen>

³<http://www.vondel.humanities.uva.nl/ecartico>

⁴<http://data.linkeddatafragments.org/dblp>

⁵<https://dblp.uni-trier.de>

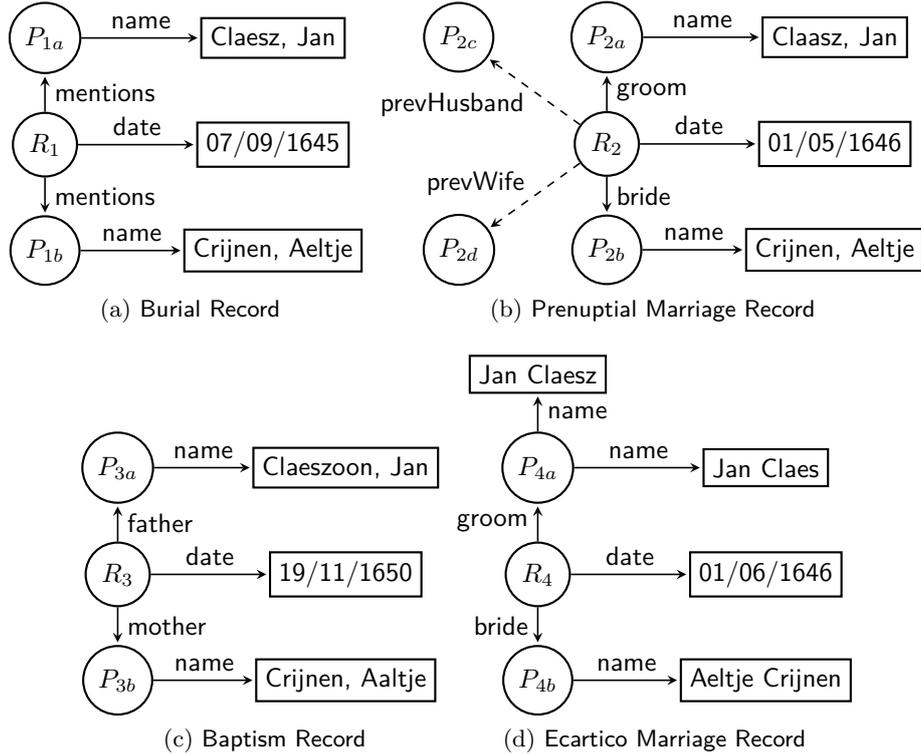


Figure 2: Each registry in the City Archives of Amsterdam has a number of records (R_1, \dots, R_n), all of which are associated with two or more persons (P_{1a}, \dots, P_{1z}). These persons can have attributes for themselves, in this simplified example we only use their full name. Dotted lines are optional relationships. Dates are often approximate.

name information, the co-occurrence of other authors and their co-authors (who again only have a name), and publication titles. This makes the DBLP KG share properties that are analogous to the merged KGs of SAA, such as n-to-n relationships, noise in literal values, the way entities can be related (through literal nodes) and similar cluster size distribution. Note that the clustering objective, where we try to cluster together person nodes in records, remains similar to that of SAA. Finally, we note that since we have a complete ground truth for DBLP, we can compute reliable precision and recall values in our experiments.

4.3. Clustering Algorithms

We perform our experiments by first constructing the weighted complete graphs C' , explained in Section 3.5, for both SAA and DBLP knowledge graphs. Then, four different clustering algorithms are applied and compared, with transitive closure method as the baseline.

4.3.1. Transitive Closure:

This method, which is used as the baseline, generates the transitive closure of the set of entities involved in a connected component. In this case, each connected component C' is treated as a single cluster, regardless of weights. For small similarity thresholds of $\theta \approx 0.5$, this results

in fewer but larger components, and thus clusters, while a large $\theta \approx 1$ results in many small (singleton) clusters.

4.3.2. Center Clustering [17]:

This method first sorts all entity pairs in C' in descending order by their weights. Then, for each entity pair, both nodes are clustered according to the following rules: 1) If neither node is assigned to a cluster, then assign one of the nodes as the center of a new cluster and add the other node to that cluster. 2) If one of the nodes is the center of a cluster, and the other node has no cluster, then add the other node to that cluster. 3) Otherwise, do nothing. The center clustering algorithm has a tendency to create many small clusters based on strong links, as the pairs with highest similarity are treated first, yielding a high precision but often at the expense of recall. For each (sub)component, all possible entity pairs are calculated to supply the algorithm with maximum information.

4.3.3. Merge-Center Clustering [17]:

This method adds another step to the center clustering algorithm where if one of the nodes is a center node, and the other node is already assigned to a different cluster, then the two clusters are merged. This tends to result in fewer and larger clusters than center clustering.

4.3.4. Vote Clustering [14]:

This algorithm processes the nodes in C' in arbitrary order. Each node i is assigned to the cluster with the largest positive sum of weights with regard to i . If there are no clusters yet, or no cluster has a positive sum, then a new cluster is created for i . For high values of $\theta \approx 1$, this will cause most sums to be negative, thereby creating more clusters. Vote clustering has been suggested as a heuristic alternative to the exact correlation clustering, explained next.

4.3.5. Correlation Clustering [4]:

This method partitions the nodes of $C' = (V, E)$ into a number of disjoint subsets (clusters) such that the sum of the inter-cluster positive weights minus the sum of the intra-cluster negative weights is minimized. More formally, let $\Gamma = \{V_1, V_2, \dots\}$ denote a partitioning of V into disjoint subsets (clusters). Furthermore, let $E^+ = \{(i, j) \in E : w_{ij} > 0\}$ denote the edges with positive weight, and let $E^- = E \setminus E^+$ denote the edges with non-positive weight. Finally, let $\text{intra}(\Gamma)$ denote the collection of edges with both endpoints in the same partition (cluster), and $\text{inter}(\Gamma)$ the collection of edges with both endpoints in a different partition (cluster). The objective is to find the clustering Γ that minimizes the cost:

$$\text{cost}(C', \Gamma) = \sum_{(i,j) \in \text{inter}(\Gamma) \cap E^+} w_{ij} - \sum_{(i,j) \in \text{intra}(\Gamma) \cap E^-} w_{ij}$$

The intuition behind minimizing the cost function is to discourage the situation where nodes with positive similarity are assigned to different clusters and nodes with negative similarity are assigned to the same cluster. Note that high values of θ will cause most weights to be negative and to have an associated cost of putting the corresponding nodes in the same cluster. This will yield an optimal solution with smaller clusters. Low values of θ will result in fewer but larger clusters. Due to the NP-hardness of the correlation clustering problem, we only

Table 1

The maximum $F-\frac{1}{2}$ and F-1 Scores and associated similarity threshold θ for the SAA KGs and DBLP KG

	SAA				DBLP			
	F- $\frac{1}{2}$ score		F-1 score		F- $\frac{1}{2}$ score		F-1 score	
	θ	max	θ	max	θ	max	θ	max
closure	0.80	0.81	0.72	0.73	0.83	0.82	0.79	0.72
center	0.63	0.72	0.60	0.54	0.74	0.61	0.65	0.43
merge-center	0.78	0.73	0.66	0.58	0.80	0.68	0.76	0.53
vote	0.79	0.81	0.69	0.73	0.80	0.82	0.72	0.73
corr. clust. & vote	0.75	0.81	0.63	0.73	0.79	0.82	0.71	0.73

perform correlation clustering on components no larger than 50 nodes. Larger components are processed with the vote algorithm.

5. Results

Table 1 together with Figures 3 and 4 show the results of the experiments for both the SAA and DBLP data sources. For a range of values for θ (in steps of 0.01) we generate precision, recall and associated F-scores. Table 1 reports the similarity threshold θ that yields the maximum F-scores for each clustering method. It is our experience that precision is more important for minimizing errors than recall, when the clusters are used in downstream tasks, such as to be used by a SPARQL reasoner. This reasoner will conclude the transitive closure of all sameAs links it is given and may therefore return unwanted results when precision is low, i.e. there are many false positive links. Furthermore, domain experts tend to prefer fewer correct results over many unreliable ones. Therefore we also report the $F-\frac{1}{2}$ score, which weighs precision twice as heavily as recall.

We first observe that the performance of the proposed clustering methods applied to SAA is similar to those applied to DBLP. Furthermore, the maximum F-scores of the baseline are as good as the maxima of the correlation clustering and vote algorithms. However, in practice the optimal similarity threshold cannot be known in advance, as there is only partial ground truth available (or none at all). It is therefore important to use a method that does not depend strongly on the choice of threshold, and preferably has a high overall precision (and F-score). Figure 3 presents the overall scores of the clustering methods for various similarity thresholds applied to SAA, while figure 4 does the same, but with respect to the DBLP dataset. The center clustering algorithm has good precision, but suffers from very low recall, as seen in Figures 3c, 3d, 4c and 4d. Note that both center and merge-center have a much lower maximum F-score than the other methods. From Figures 3b and 4b we can see that the correlation clustering and vote combination performs best by a slight margin, with good precision while not suffering too much in recall. The vote algorithm by itself is nearly as good and much more efficient. Both are a good choice when it comes to overall performance over a range of thresholds.

6. Conclusion

The domain of cultural heritage often makes use of KGs that have characteristics which make them difficult to process, such as noise (misspelling of names), proxy variables and n-to-n

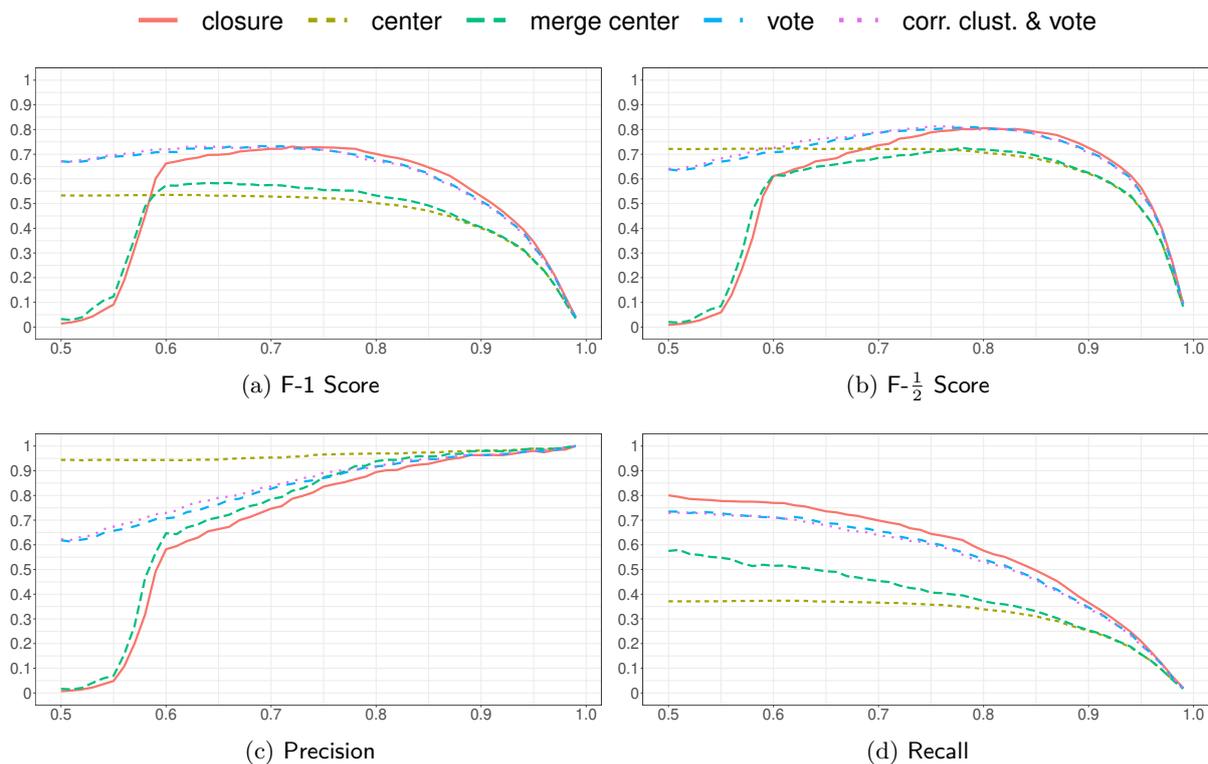


Figure 3: Results for a range of similarity thresholds (horizontal axis) for the SAA KGs.

relationships. Furthermore, most of the time there is either no or a very limited ground truth available.

We have shown that our method can handle the complexities of real KGs from the cultural heritage domain and is able to yield a result with high precision while still having good recall. The experiment with the DBLP KG shows that the results can be replicated on a different KG with similar structural properties. Furthermore, we have shown that several different clustering algorithms can be applied in our method.

The use of an unsupervised method can be most appropriate when there is no or very limited ground truth available. Moreover, in case of limited ground truth a supervised learner can work if only a few features are used. Therefore, in the future we will experiment with introducing supervised learning in combination with additional symbolic features, such as logical rules about how certain properties relate, to create a system that combines both symbolic and sub-symbolic information.

6.0.1. Reproducibility

We have published the source code⁶ for creating the embeddings and the scripts for running the experiments online⁷, including the instructions on how to build the source code into a standalone executable, the configuration files that were used to create the embeddings, and the KGs used in the experiments.

⁶<http://www.github.com/jurian/graph-embeddings>

⁷<http://www.github.com/jurian/chr2021>

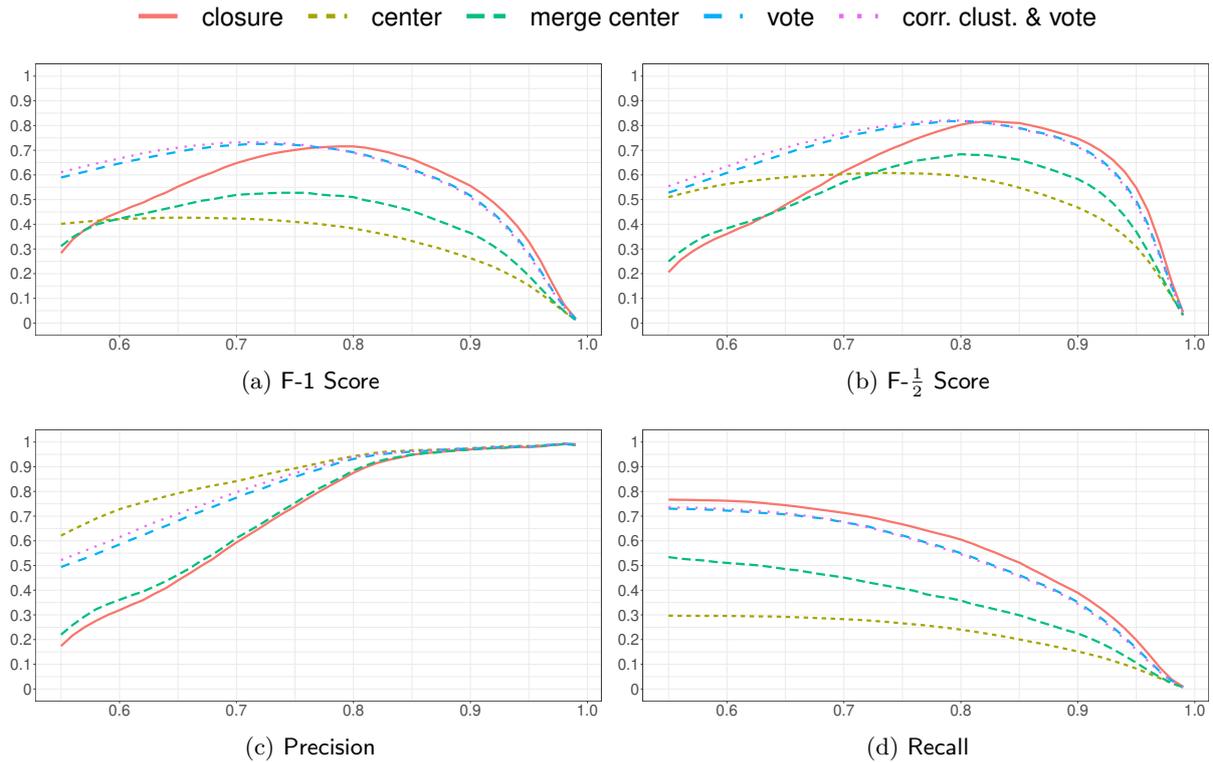


Figure 4: Results for a range of similarity thresholds (horizontal axis) for the DBLP KG.

References

- [1] M. Al Hasan and M. J. Zaki. “A survey of link prediction in social networks”. In: *Social network data analytics*. Springer, 2011, pp. 243–275.
- [2] J. Baas, M. Dastani, and A. Feelders. “Exploiting Transitivity for Entity Matching”. In: *ESWC2021 Poster and Demo Track*. 2021.
- [3] J. Baas, M. Dastani, and A. Feelders. “Tailored graph embeddings for entity alignment on historical data”. In: *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services*. 2020, pp. 125–133.
- [4] N. Bansal, A. Blum, and S. Chawla. “Correlation Clustering”. In: *Mach. Learn.* 56.1-3 (2004), pp. 89–113.
- [5] P. Berkhin. “Bookmark-coloring algorithm for personalized pagerank computing”. In: *Internet Mathematics* 3.1 (2006), pp. 41–62.
- [6] E. Bernhardsson. *Annoy at GitHub*. <https://github.com/spotify/annoy>. 2015.
- [7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. “Translating embeddings for modeling multi-relational data”. In: *Neural Information Processing Systems (NIPS)*. 2013, pp. 1–9.
- [8] M. Chen, Y. Tian, K.-W. Chang, S. Skiena, and C. Zaniolo. “Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment”. In: *arXiv preprint arXiv:1806.06478* (2018).

- [9] M. Chen, Y. Tian, M. Yang, and C. Zaniolo. “Multilingual knowledge graph embeddings for cross-lingual knowledge alignment”. In: *arXiv preprint arXiv:1611.03954* (2016).
- [10] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis. “An Overview of End-to-End Entity Resolution for Big Data”. In: *ACM Comput. Surv.* 53.6 (2020). DOI: 10.1145/3418896.
- [11] M. Cochez, P. Ristoski, S. P. Ponzetto, and H. Paulheim. “Global RDF vector space embeddings”. In: *International Semantic Web Conference*. Springer. 2017, pp. 190–207.
- [12] K. Do, T. Tran, and S. Venkatesh. “Knowledge graph embedding with multiple relation projections”. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. Ieee. 2018, pp. 332–337.
- [13] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. “Distributed representations of tuples for entity resolution”. In: *Proceedings of the VLDB Endowment* 11.11 (2018), pp. 1454–1467.
- [14] M. Elsner and W. Schudy. “Bounding and comparing methods for correlation clustering beyond ILP”. In: *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*. 2009, pp. 19–27.
- [15] A. Grover and J. Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.
- [16] R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar. “Accelerating large-scale inference with anisotropic vector quantization”. In: *International Conference on Machine Learning*. Pmlr. 2020, pp. 3887–3896.
- [17] O. Hassanzadeh and R. J. Miller. “Creating probabilistic databases from duplicated data”. In: *The VLDB Journal* 18.5 (2009), pp. 1141–1166.
- [18] B. Hendriks, P. Groth, and M. van Erp. “Recognising and Linking Entities in Old Dutch Text: A Case Study on VOC Notary Records”. In: *Proceedings of the International Conference Collect and Connect: Archives and Collections in a Digital Age*. 2021, pp. 25–36.
- [19] A. Idrissou, V. Zamborlini, F. Van Harmelen, and C. Latronico. “Contextual entity disambiguation in domains with weak identity criteria: Disambiguating golden age amsterdamers”. In: *Proceedings of the 10th International Conference on Knowledge Capture*. 2019, pp. 259–262.
- [20] A. K. Idrissou, R. Hoekstra, F. Van Harmelen, A. Khalili, and P. Van Den Besselaar. “Is my: sameas the same as your: sameas? lenticular lenses for context-specific identity”. In: *Proceedings of the Knowledge Capture Conference*. 2017, pp. 1–8.
- [21] M. Koho, P. Leskinen, and E. Hyvönen. “Integrating historical person registers as linked open data in the warsampo knowledge graph”. In: *Semantic Systems. In the Era of Knowledge Graphs. SEMANTiCS* (2020), pp. 118–126.
- [22] C. Latronico, V. Zamborlini, and A. Idrissou. “AMSTERDAMERS: from the Golden Age to the Information Age via Lenticular Lenses”. In: *Digital Humanities Benelux*. 2018.
- [23] M. Nentwig, A. Groß, and E. Rahm. “Holistic entity clustering for linked data”. In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. Ieee. 2016, pp. 194–201.

- [24] A.-C. N. Ngomo and S. Auer. “Limes-a time-efficient approach for large-scale link discovery on the web of data”. In: *integration* 15.3 (2011).
- [25] J. Pennington, R. Socher, and C. D. Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.
- [26] B. Perozzi, R. Al-Rfou, and S. Skiena. “Deepwalk: Online learning of social representations”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.
- [27] J. Raad, R. Mourits, A. Rijpma, R. Zijdemans, K. Mandemakers, and A. Merono-Penuela. “Linking Dutch civil certificates”. In: (2020).
- [28] J. Raad, N. Pernelle, and F. Saïs. “Detection of contextual identity links in a knowledge base”. In: *Proceedings of the knowledge capture conference*. 2017, pp. 1–8.
- [29] A. Saeedi, M. Nentwig, E. Peukert, and E. Rahm. “Scalable matching and clustering of entities with FAMER”. In: *Complex Systems Informatics and Modeling Quarterly* 16.16 (2018), pp. 61–83.
- [30] A. Saeedi, E. Peukert, and E. Rahm. “Using link features for entity clustering in knowledge graphs”. In: *European Semantic Web Conference*. Springer. 2018, pp. 576–592.
- [31] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. “Silk-a link discovery framework for the web of data”. In: *Ldow*. 2009.
- [32] Z. Wang, J. Zhang, J. Feng, and Z. Chen. “Knowledge graph embedding by translating on hyperplanes”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 2014.
- [33] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma. “Collaborative knowledge base embedding for recommender systems”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 353–362.
- [34] C. Zhao and Y. He. “Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning”. In: *The World Wide Web Conference*. 2019, pp. 2413–2424.