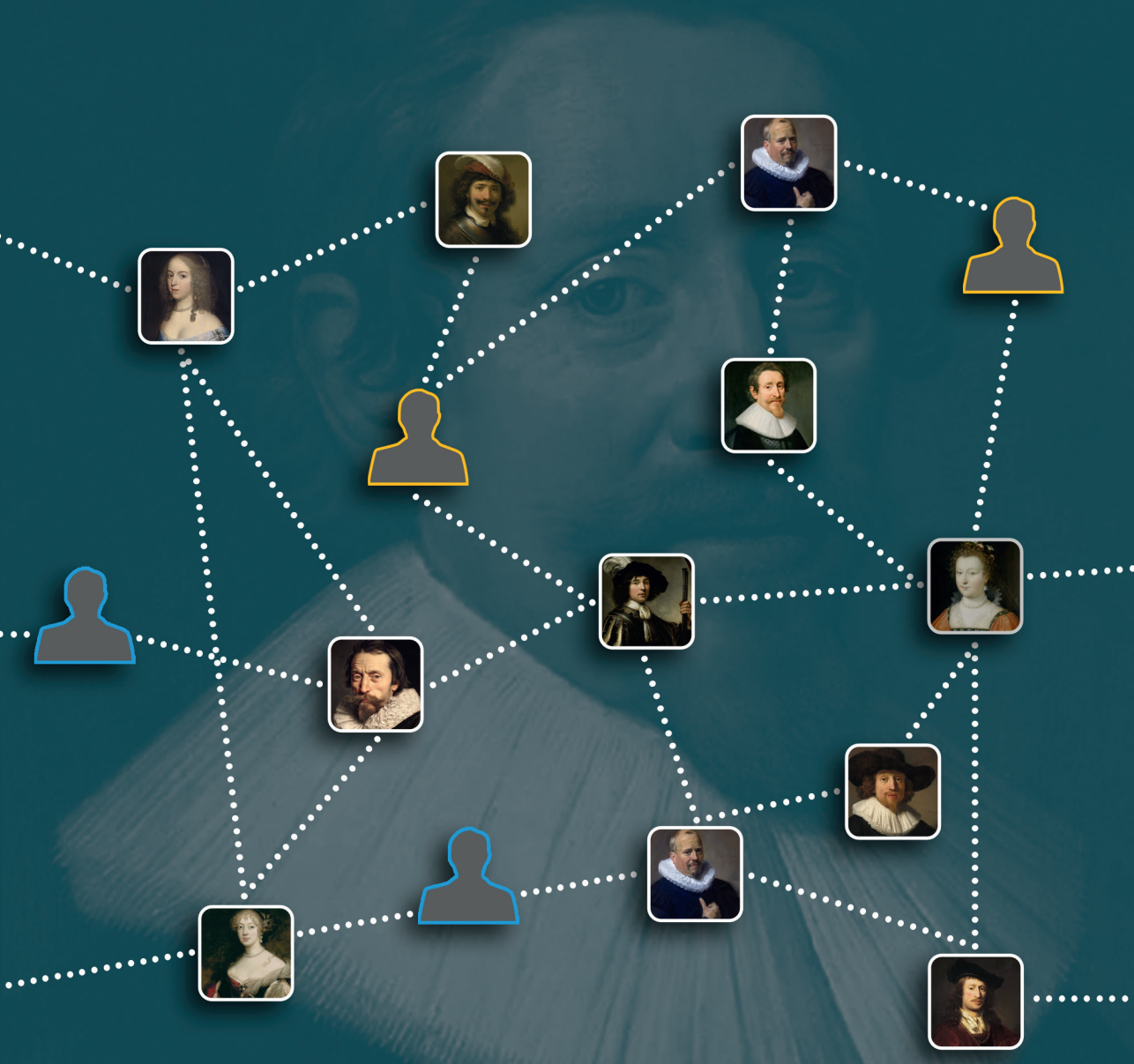
Entity Resolution on Historical Knowledge Graphs



Jurian Baas

Entity Resolution on Historical Knowledge Graphs

JURIAN BAAS

Utrecht University

SIKS Dissertation Series No. 2023-25
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Cover by Ruud Baas
Printed by Ridderprint | www.ridderprint.nl

ISBN 978-90-393-7596-9

Entity Resolution on Historical Knowledge Graphs

Entiteitsresolutie in Historische Kennisgrafen

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit Utrecht
op gezag van de
rector magnificus, prof.dr. H.R.B.M. Kummeling,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen op
maandag 16 oktober 2023 des middags te 14.15 uur

door

JURIAN BAAS

geboren op 18 mei 1989
te Gouda, Nederland

Promotoren:	Prof. dr. M.M. Dastani
	Prof. dr. E. Stronks
Copromotor:	Dr. A. J. Feelders
Beoordelingscommissie:	Prof. dr. A. Betti, Universiteit van Amsterdam
	Dr. E. A. M. Caron, Tilburg University
	Prof. dr. J. van Eijnatten, Universiteit Utrecht
	Prof. dr. C.M.J.M. van den Heuvel, Universiteit van Amsterdam
	Prof. dr. Y. Velegrakis, Universiteit Utrecht

Contents

1 Introduction

There exists a large body of work on the topic of creative goods and their associated producers in the Dutch Golden Age. Examples of creative goods are plays, books, pamphlets and paintings. For such types of goods, humanities researchers have either studied the creative goods, e.e. paintings, themselves, or they have created registers containing information about the individuals that produced these goods. Registers on producers are very valuable for in-depth research into a few selective individuals, but are insufficient for uncovering the broader patterns between these producers and the consumers of their goods. In fact, the consumer side of things was never accounted for in depth, as the available data was too impenetrable for such an analysis. For example, one could examine probate inventories to get an idea of who owned (consumed) what products. However, these inventories contain many duplicates and have no links with other related data sets. Furthermore, there may not be a complete ontology which describes all the different roles for individuals, types of goods, etc. It is therefore very difficult to extract meaningful global patterns from these data sets.

In parallel with this ongoing research by historians, semantic web technology is increasingly being used by humanities researchers. This technology makes it easier to access large-scale data sets from the cultural heritage world, such as the indexes on persons and locations of the Amsterdam City Archives. Semantic web technology also facilitates the integration of different data sources into knowledge graphs, which in turn enables cross-data set analyses that were previously infeasible. This makes it possible, for example, to reconstruct someone's life on the basis of primary archival sources. However, the integration of different historical data sets entails a number of complications. The majority of these archival data sets have been digitized with the aim of providing quick and easy access for scholars. This practice causes the issue that when the same person is included multiple times within a single data set, or appears multiple times across different data sets, they will have a new entry and unique identifier each time.

The project *Golden Agents: Creative Industries and the Making of the Dutch Golden Age*[1] was created in an effort to, among other things, combine semantic web technology with ongoing research into consumers and producers of creative goods during the Dutch Golden Age. The Golden Agents project has developed a research infrastructure to study relations

[1] `https://www.goldenagents.org`

and interactions between producers and consumers of creative goods during the 17th and 18th century in Amsterdam. It brings together heterogeneous data sets from several content providers as linked open data, i.e. using knowledge graphs. However, the independent nature of the institutions that govern these data sets causes them to use different identifiers to refer to the same real-world object, if they provide an identifier for these resources at all. Especially when dealing with archival data the situation is even more complicated, as it is rarely the case that entities are identified as more than a textual reference to e.g. a person or a location. Due to the size and type of this archival data, internal disambiguation or external reconciliation is often not available, which makes every reference an unresolved ambiguous one. In order to use these data sets for prosopographical (common characteristics of a group of people) and biographical research, and ask new quantitative questions that shed more light on the production and consumption of cultural goods, we first need a way to efficiently disambiguate these textual references to entities. This is a very important step, since individuals and the networks they form are at the heart of the creative industry. Disambiguating these textual references makes it possible to better understand the impact of the Dutch Golden Age on the creative industries and its actors. This dissertation offers a solution to this problem and describes a method to reduce the number of duplicates in a set of knowledge graphs by clustering these unique entries, where each cluster represents a single real life object.

Before we go into detail on the method, we start with a short explanation of some terminology which is important for understanding the general method. A 'vector' is a list of numbers of fixed length. Vectors are very useful for many purposes and are widely used across all fields of science. For example, vectors of length 3 can be used to represent the coordinates of entities in a 3d space. Vectors are also very useful as representations of entities which are otherwise difficult to handle by a computer. Entities such as words in a document or, as in our case, nodes in a knowledge graph. An additional benefit is that it is easy to compute similarity scores between vectors, given they are the same number of components (length). This way we can, for instance, compute that the two words 'apple' and 'orange' are similar, as they both belong to the category 'fruit', even when they have few letters in common. When we create a vector representation of an entity, we say that we have 'embedded' that entity. The embedding itself consists of many embedded entities, each with its own associated vector representation. In other words, an embedding is simply a set of vectors. With these definitions in mind, we next describe in general terms the disambiguation method explained in this dissertation.

The method described in this dissertation is called *Disambed*, which is short for *disambiguation with embeddings*. Disambed is designed to work well with the specific difficulties of historical data sets that are represented as knowledge graphs (i.e. networks). It constructs embeddings such that the vector representations of two nodes in a knowledge graph have high similarity if these nodes have a strongly overlapping context in the knowledge graphs. In this case, the context of a node consists of the other nodes in the knowledge graph that are close by in the graph. An intuitive way of understanding how such a context is created is to imagine dropping some paint on a node in the graph and then watching how this paint spreads to other nodes along the edges of the graph. After a while, the paint dries out and stops flowing. All nodes which have at least some paint on them are then part of the context of the node we originally dropped paint on. We work under the assumption that when two

nodes have a strongly overlapping context, and therefore have similar vector representations, this is indicative of them being duplicates. In fact, we modify the underlying knowledge graphs such that this assumption holds in such cases. After we have embedded all nodes that we wish to disambiguate, we can then find potential duplicates of a node by searching for the nearest neighbors of the vector representation of that node. This yields a number of pairs, each with an associated similarity score. A naive approach would be to first set a threshold, and then accept all node pairs with a similarity larger than this threshold as being duplicates. However, relying on these pairwise matches is not without risks. This is because we are using an equivalence relation, i.e. we are saying that two nodes are duplicates, which means that the transitivity relation holds as well. For example, when we say that the entities in the pairs (A,B) and (B,C) are duplicates, this must mean that that entities A and C are duplicates as well. The application of this naive approach with a threshold value can lead to transitivity violations. That is, the entity pairs (A,B) and (B,C) can both have a high similarity above a threshold, but this need not be the case for pair (A,C). To cluster together entities A, B and C means that somewhere a mistake was made in the values of similarities. This mistake has to be rectified. This issue can occur due to, for instance, insufficient information in the original data. To address this, we employ algorithms that make use of the pairwise similarities to find clusters that conform as best as possible to the computed similarities. Nevertheless, it happens that these clustering algorithms produce false positives and negatives. To counteract this and improve the clustering results, this work describes the use of domain-specific knowledge and constraints to detect and correct clustering errors. An example of such a restriction is that one cannot marry oneself, i.e. two references to persons within a single marriage record are never duplicates. Another example is the natural flow of ones life. That is, a person living in 17^{th} century Amsterdam is first baptized and then buried.

Lastly, the output that is generated by Disambed is intended to be used by a multi-agent framework that was developed in parallel [19, 20]. To this end, Disambed has been developed to act as a standalone application. The source code of Disambed is also freely available[2]. Moreover, Disambed can be applied to the disambiguation task of any type of entity across any number of knowledge graphs.

1.1 Road Map

This dissertation was written in such a way as to best explain the entire disambiguation method, which was developed over a period of several years. During this period, many parts of the method were iteratively improved and refined. As such, we have not included the published papers in chronological order, but instead written this thesis as a narrative with a step by step description of the entire process from the knowledge graphs that act as input all the way to the final clusters. What follows next is a short introduction to each of the chapters.

1.1.1 Relation to Existing Work

In this chapter we position our work and highlight and compare several related techniques and tools. The subject of this dissertation, namely the problem of entity resolution in historical

[2] `https://github.com/Jurian/disambed`

data sets, which are represented as knowledge graphs, is inherently very interdisciplinary. Important related fields are semantic web, data mining, machine learning and cultural heritage. Of course, much work on entity resolution has already been done in each of those fields, with the occasional overlap between two fields, such as machine learning and cultural heritage. This process of independent research across multiple domains has caused some confusion in terminology, as works often use terms such as entity identification, entity linking, entity disambiguation, entity resolution, entity deduplication, entity alignment and record linkage. These terms are typically used for related, but slightly different problems, and the meaning may vary somewhat between different publications. The novel contribution of this work is a high performance entity resolution method that has been developed in the interdisciplinary environment mentioned above.

1.1.2 Historical Data

This chapter discusses the data sources and how they are represented as knowledge graphs. As mentioned before, semantic web technology is increasingly used by researchers in the humanities to answer important - yet with traditional means often hard to answer - questions in their respective fields. Not only does this technology make it possible for the data to be more easily accessed and large scaled, it also facilitates the integration of other relevant data sources, allowing for the answering of questions that are only answerable when different data sets are combined. The conversion of existing data sets to knowledge graphs remains a work in progress and is performed by experts of both semantic web technology and the cultural domain of the data set. Much effort is required to first clean up and then properly structure the data with a custom ontology. This was also the case in the Golden Agents project. As such, for the duration of the project, these data sets were constantly improved upon by adding new registries and cleaning up attributes and values.

1.1.3 Reconciliation

Our method relies on the fact that highly related, and therefore potentially duplicate, entities are closely interconnected in the knowledge graphs. However, in the setting of the Golden Agents project, but in general as well, each individual knowledge graph uses unique identifiers (URIs) for resources, such as persons, with no overlap between the different knowledge graphs, i.e. no URI is shared between knowledge graphs. This means that shared literals, such as names, are the only means of creating an interconnection between entities. However, in many cases there is no exact match between literals to make such connections, e.g. due to spelling variations of names. This would cause potentially duplicate entities that are very similar but not identical in their properties to be either disconnected or are far apart in the knowledge graph, i.e. have a long minimum traversal path between each other. As mentioned before, this causes problems with the assumption of interconnectedness of highly related entities. Therefore we need to create a shorter path between these similar entities by adding new edges to the graph which are weighted by the computed similarity between their respective properties such as names. If we do this, it allows for graph traversal algorithms to reach all nodes in the merged graph which may be relevant for some particular starting entity. These entity neighborhoods can then provide the basis on which the entities can be disambiguated.

1.1.4 Embedding RDF Nodes

As mentioned in the introduction, embeddings are a well known technique for creating numerical representations for entities that are otherwise difficult to process, such as words in text or, as in our case, nodes in a graph. When we assign a numerical vector representation to an entity, we say that we embed that entity. In this chapter we discuss how relevant nodes in an knowledge graph (such as all nodes of `type:Person`) are embedded based on their graph neighborhood structure. A useful feature of embeddings is that, when constructed appropriately, the similarity between entities can be calculated (in parallel) using Euclidean distance or cosine similarity, enabling further processing such as clustering entities based on their similarity.

1.1.5 Clustering with Pairwise Similarities

The chapter on embedding RDF nodes discusses how we create embeddings such that we can calculate pairwise similarities between entities. These similarities can then be used to classify entity pairs as duplicates. In the context of semantic web, we establish sameAs relationships between entities. As mentioned before, this method of classifying pairs of entities with the use of pairwise similarities creates a problem, as it may violate transitivity. Whereas the sameAs relationship, which denotes that two entities are equivalent, is obviously transitive. For example, the pairs (A,B) and (B,C) may have high similarity, but when using embeddings, there is no guarantee that the entities in pair (A,C) are also as similar. This creates an issue when we want to create clusters for entities A, B and C. Should we group A and B together and keep C separate? Maybe the information of high similarities of B with both A and C constitutes as evidence that the low similarity between A and C was in error. The intuition here is that the decision of how to group these entities into clusters depends on the specific magnitudes of the similarities between each entity. A solution is to group entities into clusters such that we minimize the sum of weights between entities in different clusters and maximize the sum of weights between entities inside a cluster. Furthermore, there is an algorithm called correlation clustering that provides the optimal solution for this problem. However, correlation clustering becomes increasingly computationally infeasible as the number of entities increases. Therefore we have experimented with several alternative heuristic clustering algorithms that are able to be computed much faster, while retaining much of the performance compared to the exact solution.

1.1.6 Incorporating Domain Knowledge

The application of Disambed with the techniques described above yields good results. Nevertheless, there may still be errors in the entity resolution outcome. That is, a false positive error can occur (type 1 error), where pairs of entities may be identified as duplicate while in reality they are not. Alternatively a false negative error may happen (type 2 error), where two duplicate entities are assigned to different clusters. These errors may come about due to inherent weaknesses in the embedding technique, or incomplete and unreliable information in the original data. Due to the lack of ground truth, a very common occurrence in the field of Digital Humanities, it is not viable to switch to a supervised method in an attempt to improve performance. Therefore, it is necessary to use other types of information, such as rational axioms and domain knowledge, to improve the entity resolution method. In this chapter we

focus on reducing the number of false positive errors.

We argue that domain-specific knowledge can be used to detect and correct matching errors, and propose an approach to incorporate domain knowledge in an existing entity resolution algorithm. Examples of such domain-specific knowledge are

1. Two entities cannot be identical if they occur in the same civil registration record.
2. Two entities, one with birth date x, the other with marriage date y cannot be identical if $x > y$.

These rules were created with the help of domain- and data experts. The purpose of these rules is to test the conclusion of a sub-symbolic method that two entities are the same. Furthermore, a complication to the use of such rules is that domain-specific knowledge can involve uncertainty due to the ambiguity in the data. For example, if we want to exploit the fact that a person cannot be born after they have died, one must be able to identify the born and dead persons unambiguously. The difficulty is that in this type of archival sources it is not a given that the person is actively involved in the registration event when they are mentioned. It could even be that the person is already deceased and is solely used as a disambiguating description for someone else (e.g. *Claartje Jans, widow of Cornelis Pieters*). If this happens in a burial registration or the registration of a testament, we need a rule that is aware of this knowledge and we do not treat the fact of a person's death as 100% certain, but see this as relative to the number of persons involved in the event. This uncertainty then propagates to the conclusion of the rule.

1.1.7 Conclusions

In the final chapter, we discuss what we learned during the development process of Disambed per major topic and make a number of observations on data management in the field of Digital Humanities. Additionally, we discuss future work on how entity resolution may be performed in such as way that the output, and by extension the learned patterns, are easier to understand.

2 Relation to Existing Work

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 identification, entity linking, entity disambiguation, entity resolution, entity deduplication, entity alignment and record linkage. These terms are typically used for related, but slightly different problems, and the meaning may vary somewhat between different publications. For example, record linkage is usually used in the context of matching records in one data set with records in other data sets. These works can be rule-based [46, 66], make use of word embeddings of tokens that appear in the descriptions of entities [23, 71], or focus on scalability [45, 61, 62]. Some of these works make use of structured (tabular) data and exploit extra information such as that the sources to be combined are themselves duplicate free. 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. In our case we cannot make this assumption. To position our work we make use of the taxonomies defined in the survey papers of Christophides et al. [17] and Morris et al. [44]. The first survey divides the entity resolution problems into three categories:

1. *Clean-Clean entity resolution*: In this situation each data set has no internal duplicates and the objective is to find one-to-one matches between them. An example of such a case is finding matches between two well curated knowledge graphs that are in different languages. For example, finding the corresponding URI for the city of Napels in an English knowledge graph, in an Italian knowledge graph where the city is mentioned as Napoli.

2. *Dirty entity resolution*: The goal here is to find duplicate entities in a single data set. An example is the disambiguation of authors of computer science papers in the DBLP [1] data set. A single person can be mentioned multiple times in this data set, one for each of their publications that was recorded. When multiple people share the same name, they can be distinguished by their co-authors and publication patterns such as venues. Similarly, one person can be mentioned under different names due to the different practices of venues in how they represent these names, e.g. only initials and last name or the full name. It is still possible to detect these entities as duplicates using, for example, the already mentioned co-authorship information. Of note is that the increased

[1] `https://dblp.org/`

use of digital identifiers such as ORCID [2] has substantially mitigated this issue in this particular case.

3. *Multi-source entity resolution*: Finding duplicates in more than two clean data sets. It is easy to mistake this situation for our historical setting. However, this situation usually does not account for internal duplicates in each data set, which is very common in digital humanities data sets. Examples of works that deal with multi-source entity resolution and assume clean data sets are Saeedi et al. with CLIP [61] and Nentwig et al. [45].

Our method can best be placed in both the *Multi-source entity resolution* (there exist duplicates between data sources), and *Dirty entity resolution* (data sources themselves contain duplicates) categories. There is a subtle difference between our situation and the Dirty entity resolution case, as one could say that when if we combine all our data sets we could simply apply the methods that are developed for Dirty entity resolution. This misses the fact that our data sets can be heterogeneous in their ontology, values and structure. We therefore categorize our method as *Multi-Dirty Source entity resolution*. Many other methods in the literature make (implicit) assumptions on being in only one of these categories and can therefore not be applied without modification to our problem setting.

The second survey of Morris et al. distinguishes some generalizations of sources of information and different techniques of handling the entity resolution problem in the Semantic Web field. We will briefly mention those relevant to our case and then again position ourselves with regards to these different techniques.

1. *Machine Learning*: The most commonly proposed and developed methods in the entity resolution literature make use of machine learning. The main idea is to use supervised learning to identify duplicates. For example, for disambiguation of authors in the DBLP data set, one can train a support vector machine on features containing author information and other parameters. Of course, these supervised methods need training data, which can be hard or impossible to obtain. An alternative is the use of unsupervised machine learning methods, such as clustering, to group entities into clusters where all entities in a single cluster are regarded as duplicates of each other. This is the main idea behind our method, even though we have also experimented with supervised approaches, and shown that they can work with very little training data if one is careful about choosing the right (number of) features.

2. *Lexicon & Taxonomy*: A popular approach to discovering the semantics of words is to use a taxonomy structure, ontology or lexicon, i.e. an alphabetical list of words and their definitions. For example, one can assign the right meaning to words by computing their relation to other words by looking at shared terms in their definitions. To decide whether two words are synonyms or not, the similarities between them and other possibly unrelated words need to be computed. This idea can be extended to the semantic web field by substituting words with URIs and looking at shared terms in their respective ontologies or properties. The decision is then made using a predefined cut-off value. A drawback is that the choice of the cut-off value is not obvious beforehand. We

[2] https://orcid.org/

also use a cut-off value and have experimented with ways of making this choice less impactful on the end result.

3. *Similarity Functions*: A very popular method of entity resolution is the use of similarity functions. A similarity function computes a score based on how many components two entities have in common or not. Advantages of this method are that these functions are easily formalized and it is not difficult to tune the functions to work well under specific circumstances. One way of using similarity functions for entity resolution is to compare entities based on simple characteristics such as name, address, etc. A major drawback that is not often mentioned is the fact that many of these functions do not necessarily obey the transitivity relation. That is, a function could yield a high similarity score for entity pairs (i, j) and (j,k), but not for the pair (i,k). One then has to make a decision on which computed similarity was correct. Dealing with this issue in a principled manner is a major part of our work and is discussed in chapter 6.

4. *Structure Exploitation*: Another way of performing entity resolution in knowledge graphs is to exploit the information that is stored in the structure of the graphs. To compute this structural similarity between two entities i and j, one can take into consideration matching names and other such features, and also their respective neighborhoods. A simple approach to neighborhood matching could be to compute the Jaccard similarity between the neighborhoods of i and j. This similarity can then be used as a feature and combined with other features to come to a conclusion as to whether i and j are duplicates. Our method also makes use of structures in the knowledge graphs to generate features for entities, on which similarities can then be computed. However our method builds on the work of Cochez et al. [18], which uses random walks to generate neighborhoods and then uses a popular natural language processing technique called GloVe [52] to assign numerical features to each node in the knowledge graph. This approach, discussed in chapter 5, has more in common with the next item in this list and is therefore explained more thoroughly there.

5. *Word-sense disambiguation*: An important aspect of more general entity disambiguation is word sense disambiguation, as words can have different meanings in different contexts. This has lead to studies that attempt to make sense of words based on the context in which they appear. Such concepts can be translated to the semantic web field, where nodes in a knowledge graph that are identified by a URI can be distinguished and related based on the context in which they appear. Such a context is usually defined as the local neighborhood in the graph, obtained in various ways such as random walks and fast approximations thereof. An advantage of this approach is that it is less dependent on a common ontology, though in practice some effort is still required in this respect. We discuss these partial ontology alignments in chapter 4.

We have already mentioned several aspects in which our approach is integrated in the above listed techniques, as we make use of them all in different ways. For instance, machine learning is used to both create representations of entities and to cluster them. We have experimented with both supervised and unsupervised methods. In the end, we have settled on an unsupervised approach, as this matches the requirements of the digital humanities setting best. That is, there is little to no training data available, and generating it is relatively expensive as it requires the use of domain experts. This topic also covers aspects of lexicon- and

structure exploitation, and word-sense disambiguation. Furthermore, similarity functions are used by us in both a preliminary alignment of the data as well as in the clustering process. Finally, the clustering process takes inspiration from the lexicon approach by comparing multiple interrelated entities before making a decision on whether any of them are duplicates. In the rest of this chapter we will describe other related work, categorized by their respective overlap to different aspects of this dissertation.

2.1 Entity Alignment in Knowledge Graphs

Much work has been done in the field of entity alignment in the context of knowledge graphs. This is because entity disambiguation is a problem that has been around since databases were starting to be used extensively and, for the field of Digital Humanities, since the first digitisation initiatives of traditional (archival) indices and other entry points. Several tools have been created for this purpose, with the use of different techniques. These methods rely on either a strictly defined data model, or on the availability of sufficient data to disambiguate persons easily (e.g. a birth date). Furthermore, these methods often assume that there are only two data sets, often called source and target, and that these source and target data sets do not have internal duplicates.

For instance, Jentzsch et al. [39] have developed the Silk Link Discovery Framework, which can be used by data publishers to generate links between data sets. One of their use cases is identifying duplicate person descriptions in a data stream. They did this by crawling the FOAF (Friend Of A Friend) web and adding owl:sameAs relations between persons judged as duplicates. Additionally, Ngomo et al. [46] created LIMES, a time-efficient approach for the large-scale matching of instances on knowledge bases. They achieve this time-efficiency by using a pessimistic metric which enables them to exclude many unlikely matching candidates. This method of performing (expensive) computations only on likely matches is called *blocking* and is an important aspect of entity resolution. During blocking, the data is divided into groups (blocks) and an exhaustive search is only performed within blocks. Most, if not all, methods use some form of blocking. Otherwise they would be unusable for any realistically sized data set. Niu et al. [49] have proposed Rule Miner, a semi-supervised approach which iteratively constructs and refines rules by which high confidence matches can be made. Similar to Rule Miner is the work of Paganelli et al. [51] with TuneR, who propose a software library for tuning sets of matching rules. An example of one such rule is:

$$(name,\ edit,\ =,\ 1) \wedge (address,\ edit,\ >,\ 0.8) \wedge (city,\ edit,\ >,\ 0.8)$$

Which can be interpreted as *mark two entities as duplicate if the names of both entities match exactly, and the similarity between the addresses and city names is greater than 80 percent*. In this case the similarity will be calculated with a (normalized) edit distance. A rule set is a disjunction of a set of rules. Since there are many possible rules and rule combinations, it is important to efficiently find sets of rules that are close to optimal. They achieve this by first progressively improving the rules with a beam search that attempts to further investigate promising rules that result in a higher performance. In the next phase, the complex rules are iteratively trimmed as long as the performance does not decrease. Since

they need to measure performance, this is a supervised method and a ground truth is required.

Achichi et al. [1] note that many existing linking tools require extensive knowledge of the data. For this reason they have developed Legato, with which they attempt to minimize the difficulty of manual configuration when it comes to data-related parameters, such as which properties to compare. They take as input two knowledge graphs and are then able to automatically generate for each relevant entity a one-to-one matching between them. Briefly, the Legato method works as follows. Entities that are to be disambiguated are first represented as text documents and then projected into a vector space. The text representation of an entity consists of a set of literals that are relevant to that particular entity. These could, for instance, be all the properties, e.g. name, of that entity. The text representations are then vectorized using the TF-IDF (Term Frequency-Inverse Document Frequency) method. These vectors can then be compared with the cosine similarity measure to find what they call candidate links. These links are then refined using a blocking method, where clusters of vectors from the first data set are compared to clusters of vectors from the second data set. In our case, it is important to note that it is co-occurrence with other entities that helps in the disambiguation effort, one cannot rely solely on one-to-one comparisons, such as matching names. Another relevant method is called Lenticular Lens [35]. Lenticular Lens is exceptional in the sense that it can also deal with internal duplicates. That is, they note that it can work when the source and target are the same data set. What distinguishes our method from those explained above is that in their case only literals are used for disambiguation. Our method is more generic in the sense that all nodes related to an entity can act as context and it, in contrast to Lenticular Lens, does not solely rely on pre-defined rules to perform entity resolution. The more heterogeneous a data set is, the less feasible it is to work with a rule-based approach. These are major problems for applying other techniques to our data. Another difference is that our method takes into account the semantics of properties when comparing entities, something that is not done in, for instance, the bag-of-words model of Legato. This is important as we do not want to treat, for instance, death dates and birth dates equally in the context of an entity. That is, the fact that the birth date of an entity is the same as the death date of another entity does not mean they are similar.

Lastly, there have been recent efforts to use entity alignment techniques for the purposes of matching entities occurring in multiple knowledge graphs which are written in different languages, such as English and Chinese. For example, Wu et al. [69] make use of node neighborhood matching while taking into account the fact that, in many cases, entities and their counterparts might not have similar neighborhood structures. Zhu et al. [72] make use of an attention mechanism to learn a neighbor-level representation and integrate this with a relation-level feature representation. Meanwhile, Wong et al. [48] use both the local semantics (i.e. the predicates) of an entity together with the global knowledge graph structure. Finally, Trisedya et al. [64] adapt TransE [12] to make use of the similarities between the attributes of entities. All these translation oriented methods assume one-to-one duplicate relationships and are therefore not usable in our case. Furthermore, they usually present their results in terms of hits@k and leave it at that. One can take the top result in the ranking and then mark it as used. This would be sufficient for their use case. However, this is insufficient in our case, as this does not solve the problem of clustering multiple entities.

2.2 Learned Entity Representations

Most relevant to our setting are methods that make use of representation learning techniques. The key idea is to learn vector representations, called embeddings, of nodes in KGs, such that nodes with similar neighbor structures in the KG have a close representation in the embedding space.

The most relevant method in this section is the work of Cochez et al. [18] as we partly build upon it in this dissertation. They generate a co-occurrence matrix from the input knowledge graph with the Bookmark Coloring Algorithm [9], which is then fed into the GloVe [52] algorithm. The node features are derived using graph walks, in this case generating an approximation of the personalized PageRank for each node, which is then fed directly into GloVe. One of the differences with our work is that the weights they use are calculated by a choice of different network statistics such as (inverse) predicate frequencies, instead of based on domain expert input and semantic similarities between literal values. Chapter 5 explains in detail this method and all the adaptations we made to it.

More generally, these techniques have their origins for undirected homogeneous graphs with Deepwalk [53] and Node2Vec [29]. Deepwalk, designed by Perrozi et al., aims at computing entity representations which work well for modeling communities in undirected homogeneous graphs. According to them, these representations should have the following characteristics:

- Adaptability: Modifying the graph should not necessitate repeating the learning process from scratch.
- Community aware: The similarity between vectors should reflect the interconnectedness between the corresponding nodes in the graph.
- Low dimensional: The dimensionality of the vector space should be low to generalize better when community labeling is scarce.
- Continuous: The vector representations should be in a continuous space. This allows communities to have smooth decision boundaries, which allows for more robust classification.

They learn these representations by use of a number of short random walks. The use of random walks gives two desirable properties: First, it is easy to parallelize. Second, by only examining the graph locally, the graph can be modified slightly without the need to recompute the vectors for all other nodes. We shall briefly explain the algorithm used in Deepwalk. In essence, they construct pseudo sentences out of nodes in a graph using random walks and then apply an existing language model to embed these nodes. They do this by first performing a fixed number of γ random walks of length t starting at each node in the graph. Second, they use the SkipGram [43] language model to assign vectors to each node in the graph, based on the information gathered during the random walks. The SkipGram model was originally designed for embedding words in a corpus. It works by maximizing the co-occurrence probability among words that appear in a window in a sentence. We refer the reader to the work

of Mikolov et al. [43] for more details on SkipGram.

Similar to Deepwalk, Grover et al. build with Node2Vec [29] on the intuition behind Deepwalk but instead they make use of a more flexible biased neighborhood sampling strategy that allows an interpolation between breath- and depth first searches. In short, the neighborhood sampling is parameterized with parameters p and q, which together determine the probability of jumping to a neighboring node in a random walk. Parameter p determines the probability of returning to the previous node, while parameter q influences the tendency of the random walker to visit nodes that are far away from the starting node. They argue that there several benefits of this approach over a pure breath- or depth first search. The first benefit is the fact that the space complexity of storing the neighbors of a node depends only on the total number of nodes and the average node degree in the graph, which is usually small. The second benefit is that random samples can be reused across different source nodes, which allows for a faster sampling rate. Finally, Node2Vec also uses SkipGram to compute vectors for the nodes in the graph.

Furthermore, the algorithm of Bordes et al. [12], called TransE, is relevant as it provides an alternative way of embedding nodes in a knowledge graph. We shall briefly explain the main ideas behind TransE in the context of knowledge graphs. Given a set of triples of the form (s,p,o) (subject, predicate, object), they aim to create vector representations of both the subject, object nodes and predicates such that the distance $d(s+p,\ o) \approx 0$ when (s,p,o) holds (i.e. the triple exists in the graph) and, when (s,p,o) does not occur in the graph, the distance between $s+p$ and o should be large, i.e. $d(s+p,\ o) \gg 0$. As a measure of distance they use either the L_1 (Manhattan distance) or L_2 (Euclidean distance) norm. The intuition behind this method is that translations between subject and object vectors are learned via the predicate vectors. It is then relatively simple to predict new triples by adding a predicate vector to a subject vector and searching for an object vector that is nearby. To learn such an embedding they minimize the following cost function:

$$J = \sum_{(s,\ p,\ o) \in S} \ \sum_{(s`,\ p,\ o`) \in S`_{(s,p,o)}} \Big[\gamma + d(s+p,\ o) - d(s`+p,\ o`)\Big]_+$$

Here $S`$ is a set of corrupted triples, where either the subject s or object o (but not both) where replaced with a random entity, $\gamma > 0$ is a margin hyperparameter and $[x]_+$ is the positive part of x. The intuition behind the margin γ is that when $\gamma \gg 0$, this forces the quantity $d(s`+p,\ o`)$ to be large as well for J to be minimized. An advantage of TransE is that it is relatively easy to understand and has few parameters. A downside of the TransE method is it fails at modeling relations of type 1-to-N, N-to-1 or N-to-N. Many extensions, such as [21, 68, 70], of TransE have been developed, partly to improve upon this drawback.

An alternative approach to computing compact entity representations is with the use of tensor factorization. An example of this is RESCAL, designed by Nickel et al. [47]. With RESCAL, latent component representations of the entities and predicates in a knowledge graph are learned. These representations can then be used for link prediction and entity resolution. We shall briefly explain how RESCAL works. First, for a knowledge graph G with n entities and m predicates, they construct a binary tensor X of size $n \times n \times m$, where

each k^{th} slice is the adjacency matrix of size $n \times n$ for the k^{th} predicate. They employ the following factorization:

$$X_k \approx AR_kA^T, \text{ for } k = 1, \ldots, m$$

The learning objective here is to find an $n \times r$ matrix A and m matrices R_k of size $r \times r$, where $r < n$. The value of r is learned via QR decomposition. In doing so, computing R_k is not dependent on the number of entities but instead on the complexity of the model. Approximate solutions are found with a least squares method with an additional regularization term to prevent overfitting. To use this method in practice for link prediction for the k^{th} predicate, is to compute $\hat{X}_k = AR_kA^T$. Then we can, for example, predict a link between entities i and j by checking if $\hat{X}_{ijk} > \theta$, for some threshold θ. Similarly, one can perform entity resolution by, instead of predicting *sameAs* relations as some other methods do, computing similarities between entities with the use of the heat kernel $k(i, j) = e^{-||\mathbf{a}_i - \mathbf{a}_j||^2/\delta}$. Here $\mathbf{a}_i$ and $\mathbf{a}_j$ are the i^{th} and j^{th} row in matrix A respectively, and δ is a user given constant. It is unclear which values for δ work best.

Lastly, several techniques exist for embedding nodes in the case of mapping two KGs that are in different languages, e.g. [15, 16]. In that case one can exploit the fact that there can be at most two duplicates entities per real world object.

2.3 Other Cultural Heritage Works

Others have worked on similar cultural heritage data sets but without the use of learned entity representations such as embeddings. For instance, Raad et al. [57] create a certificate linking method for Dutch civil certificates from the Zeeland region, based on string similarity computations. Furthermore, they propose a contextual identity link [56], as they observe that the `owl:sameAs` link is often misused. They note that the notion of identity can change from one context to another. For example, two pharmaceuticals may be judged as equivalent when under some conditions their names match, 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. [36, 38] 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. [41] 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. [34] 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. Finally, Efremova et al. [24, 25] perform entity resolution on civil certificates by making use of name similarity (corrected for name popularity), proximity of locations, and limited co-occurrence information, as features in regression trees and logistic regression models.

2.4 Including Domain Knowledge

An obvious source of information that can be leveraged in the entity resolution process is specific knowledge about (entities in) the data. With this information one can, for example, exclude certain matches because one can conclude they are very unlikely or impossible using the domain knowledge. Since there are many applications of domain knowledge, we restrict ourselves to those methods that apply them to the creation of embeddings.

For instance, Guo et al. [31] have developed KALE, where an embedding is learned by jointly using facts from a knowledge graph and t-norm fuzzy logic. Similarly, Rocktäschel et al. [60] use first-order logic background knowledge to aid the matrix factorisation algorithm in learning dependencies between relations. Domain knowledge can be used with pre-trained embeddings as well, Wang et al. [67] predict new facts (also known as link prediction) by combining the output of an existing embedding with physical and logical rules into an integer linear programming problem. Others have worked on the very similar problem of author name disambiguation, where authors are linked in scholarly data sets such as dblp [3]. For instance, Cen et al.[14] learn stopping criteria to use with hierarchical agglomerative clustering. This method, however, requires a training set, which is often not or very limitedly available when working with cultural heritage data. Furthermore, the time complexity of hierarchical agglomerative clustering is high compared with some other clustering methods and it needs to take the number of clusters as input while determining the number of clusters is usually an intractable problem or completely unknown in many digital humanities applications. One can not simply iterate over a set of possible numbers of clusters and pick the optimal one using a metric like AIC or BIC scores. This is because there can be many thousands of clusters with a large range between the lowest and highest estimates.

2.5 Conclusion

In this chapter we have discussed related work in the fields of entity resolution methods, learned entity representations, cultural heritage works and domain knowledge. The domain of our research is an intersection of each. The novelty of our work therefore lies in the combination of the following properties:

- We are dealing with the complex situation of having both multiple internal duplicates inside data sets and also having multiple duplicates between more than two heterogeneous data sets. We therefore categorize our method as *Multi-Dirty Source entity resolution*.
- The data sets can be heterogeneous on the value, structural and logical levels. Achichi et al. [1] from Legato have created a classification of data set heterogeneity types that provides a good covering of all the ways data sets can differ. We give a few examples when they relate to our case:
  - Terminological heterogeneity: This means that identical properties use different lexical labels. For example, names can be denoted with the `rdf:name` or `foaf:name` predicate across different data sets.

[3] `https://dblp.org/`

  - Data type heterogeneity: Properties of entities can have different types. For example one property might be modelled with a literal, while another is modelled as an URI which points to some controlled vocabulary.
  - Structural heterogeneity: A challenge for property based entity resolution methods is that properties can be modelled differently across data sets. For instance, one data set can denote both first and last name with a single literal value, while another data set makes use of a blank node to split the name into multiple parts.

- We make use of many different techniques and combine them in novel ways. Techniques such as:

  1. *Machine Learning*: We have applied both supervised and unsupervised machine learning methods. Supervised methods could be used when we had access to a limited ground truth. However, this was not a realistic assumption. Therefore we continued with an unsupervised approach. This unsupervised method was then validated by domain experts.
  2. *Lexicon & Taxonomy*: We too look for shared items between entities, such as names and other properties. However, we extend this idea to include other entities as well. For example, other entities can be persons, places or events.
  3. *Similarity Functions*: We make use of similarity functions on two occasions. First they are used on properties such as names and dates to create new links between knowledge graphs. This is explained in detail in chapter 4. Second, we make use of similarity functions on vector representations of entities to calculate clusters. How clusters are created is explained in chapter 6.
  4. *Structure Exploitation*: The existing structures in the knowledge graphs are used for creating neighborhoods for entities, these neighborhoods are then treated as a context and used for calculating vector representations for the entities. This process is explained in detail in chapter 5.
  5. *Word-sense disambiguation*: Similarly to structure exploitation, we make use of the intuition that words can be defined by the context they appear in. Words with very similar context then have the same meaning. In our case context can be substituted with neighborhood in the knowledge graph. There are problems with this approach, however, as in our case very similar entities can actually be different entities. An example would be husband and wife, who share many other entities in their respective neighborhoods. We deal with this issue by adding domain knowledge in the form of logical rules that exclude or discourage certain matches. This is explained in chapter 7.

3 Historical Data

Semantic web technology is increasingly used by researchers in the humanities such as (cultural) historians, literary scholars and art historians, to answer important - yet that are hard to answer with traditional means - questions in their respective fields. Not only does this technology make it possible for the data to be more easily accessed and large scaled, it also facilitates the integration of other different data sources, enabling for the answering of questions that can only be answered when different data sets are combined. The conversion of existing data sets to RDF remains a work in progress and is done by experts of both semantic web technology and the cultural domain of the data set. Much effort is required to first clean up and then properly structure the data with a (custom) ontology. This was also the case in the Golden Agents project. Data sets were constantly improved upon by adding new registries and cleaning up attributes for the duration of the project. All experiments described in this work were performed with combinations of the data sets described in this chapter, depending on availability and suitability at the time.

As described in the previous chapter on related methods, the properties of the historical data sets place them in the (most difficult) *Multi-Dirty Source entity resolution* category of entity resolution problems. Many other methods in the literature make (implicit) assumptions on and can therefore not be applied without modification to this problem setting. There are combinations of complications with these data sets that are quite unique to historical data sets. For example, there are errors and noise present in the data, data sets can be heterogeneous in their structure (especially in between data sets), and an entity can occur more than once under different identifiers, both inside and between data sets. This dissertation is dedicated to alleviating one of these complications, namely the issue of duplicates. With duplicates we mean the existence of multiple entities that all refer to the same real world object. For example, we can have multiple references to the same real life person. There are, however, other problems that either need to be solved by data experts or worked around by software designed to take historical data as input. An example of such a problem is the fact that the same person can have their name written in multiple variants. This makes comparing entities based on their properties problematic. Chapter 4, Reconciliation, is devoted to alleviating the problem of comparing the many different types of proprieties of entities in such a way that the property similarity information can be used later in the embedding process. This chapter discusses the data sources themselves and how they are represented as RDF graphs.

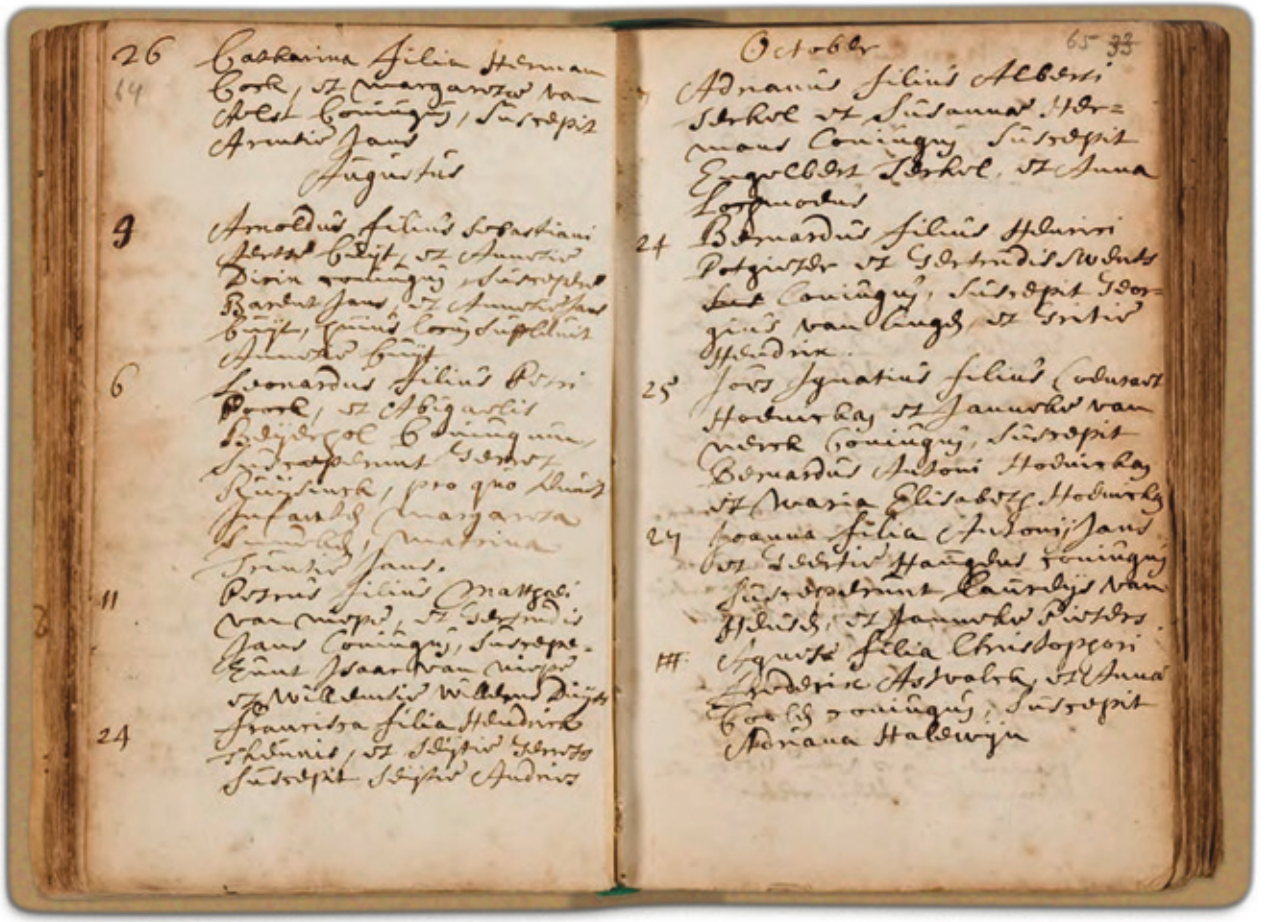


Figure 3.1: *An example of a scan of a Roman Catholic Baptism registry in the period of 1668-1698*

3.1 Amsterdam City Archives

The Amsterdam City Archives[1] play an important role throughout this work. Traditionally, the Amsterdam City Archives served the interests of its users by unlocking its data with regards to the inhabitants of the city partially via paper indices (cards) that always contain a person's name and the date of the registration in the source. With the scanning of the original sources and the digitisation of the indices starting around the year 2000, it became easier to find the individuals mentioned in each index. The Golden Agents project has taken these digitised sources and converted and modelled them into RDF (briefly explained in section 3.6) so that they can be connected to other data sets. We make use of two versions, depending on which was available at the time experiments were performed. Earlier experiments were done with the 2019 version (figure 3.2), while the later experiments where we include domain knowledge (chapter 7) use the 2021 version (figure 3.3).

3.1.1 Notice of Marriage registrations

One of the most important indices for early modernists, those who study the period of ca 1500 - ca 1800, is the index on Notice of Marriage (in Dutch: Ondertrouw) registrations. At the Council of Trent (1545 – 1563) Roman-Catholic churches were, next to baptisms, required to also register marriages. One reason was to counter bastardy and problems with inheritance, but another reason was to get a better picture of how many people belonged to which religion. A marriage had to legally be preceded by a notice of marriage, which encoded the moment when the groom and bride-to-be notified the government that they were going to get married. When the notice of marriage was registered, a period of three Sundays followed, in which any family members or outsiders had the opportunity to object to the marriage. If no objections occurred during that period, a marriage could follow.

Whereas regular Marriage registrations do not contain that much information on the occupations, ages, residency, etc. of the groom and bride-to-be, the Notice of Marriage does. This is important information for the disambiguation of persons, as this extra information can help to create better contexts for them. For the groom and bride-to-be, the following information was recorded:

- The name of the person to be married.
- The place of birth.
- Age in years, but only at the first marriage.
- The marital status of the person to be married. This would take the form of young man or young woman if not previously married, and widower, widow or divorced otherwise.
- The name of the street in Amsterdam where they lived, or the place name if they lived someplace else.
- One or more witnesses, for a first marriage a witness had to present who was able to give permission, i.e. a parent or guardian, and another witness had to confirm the

[1] `https://archief.amsterdam/indexen`

identity of the bride or groom. The first role was usually filled by the father, while the mother took on the second role. If a witness could not be present due to death or sickness, someone else could assist. Where possible, the relation of this person to the family would be written down.

- The occupation of the groom, but only until October 1714.
- The religion, but only after February 1755.

In total, the index counts 906.695 references to persons and distinguishes roles (groom, bride) in the events being registered. However, not all the information listed above was eventually digitized. For instance, the witness information is not present. Recent efforts have attempted to alleviate this issue. This extra data is coming from the volunteers of the crowd sourcing project *Ja, ik wil!* (Yes I do!) [65], who have digitized the records for every fifth year. This means that for 20% of this data, we also have information on among others the witnesses participating in the event. Both the original Notice of Marriage index as well as the enriched data have been integrated in the Golden Agents project. More information on the Notice of Marriage index can be found in Dutch at the website of the Amsterdam City Archives [2].

3.1.2 Baptism registrations

Baptismal registers are registers of a particular religious denomination for which all baptisms were registered. Regulations for keeping baptismal records were established at the Council of Trent (1545-1563). The oldest baptismal register in Amsterdam dates from one year after the Council. It is a baptismal book of the 'Oude Kerk' (Dutch for 'old church'), starting in 1564. When the Netherlands merged into the French Empire in 1810, the civil registry was introduced, a population registration after the French model. In 1811, the churches were obliged to submit their baptismal registers, well as their marriage and burial registers, to the civil registry. In this project we only use data from before 1811.

This index contains data from the baptismal registers or from similar registers with birth registration of the following denominations:

- Dutch Reformed Church
- Walloon Church
- English Presbyterian Church
- English Episcopalian Church
- Evangelical Lutheran Church
- Restored Evangelical Lutheran Church
- Mennonite Churches

[2] https://archief.amsterdam/uitleg/indexen/45-ondertrouwregisters-1565-1811

- Remonstrant Church
- Roman Catholic Church
- Old Catholic Church
- Greek Catholic Church
- Portuguese Israelite Church

Also included are the entries from the Register of the Orange-Nassau Regiment, running from 1785 to 1795. Data on births of High German Jews are not included because the registers are in Hebrew.

The entry in the register was made by the clergyman who performed the baptism. He wrote down the names as he heard them, and there was no check for completeness or spelling. As a result, people's names can be written in very different ways. The registers usually contain the first names of the child and the full name of the parents and witnesses, as well as the date of baptism. The date of birth was not initially mentioned. In 1792, the States of Holland made it mandatory to also mention date of birth and place of birth. In almost all baptismal books the name of the clergyman who performed the baptism can also be found. In the margins of the baptismal books, comments are sometimes written about, for example, name changes, the provision of baptismal certificates or important functions of parents or witnesses in the city council. An important issue with this data is that only the first name of the child is mentioned in the registers. This can make disambiguating these persons problematic. Especially in the case when a child died young, and another child was born with the same first name a few years later. It is not always obvious what the last name of the child should be. For instance, at a time when the use of patronymics was common, a child did not have the same last name as the father, because the father's first name became the patronymic of the child. For example, when the first name of the father was 'Jan', the last name of the child could become 'Janszoon' (Dutch for 'Son of Jan').

A notable issue with this data set is that, depending on the period, not all information that is in the source material has been digitized. For the period 1701-1811 all data are included, even comments written in the margins have been added. However, for the period 1564-1701 this is not the case. The index for this period only includes the names of the baptism and the parents and not the other information such as the names of the witnesses and of the pastor.

Containing 4.880.206 references to persons, the index on the Baptism registrations is the largest early modern data set of the City Archives of Amsterdam. As does the Notice of Marriage index, the index on the Baptisms includes the roles in which persons are mentioned: child, father, mother, and sometimes witness, if available in the source. Also, the churches where the event took place, are mentioned. This index is highly interesting for those looking for networks of family, friends and of religious congregations. More information on the Baptism index can be found in Dutch at the website of the Amsterdam City Archives [3].

[3] https://archief.amsterdam/uitleg/indexen/57-doopregisters-voor-1811

3.1.3 Burial registrations

One of the oldest indices of the Amsterdam City Archives is the one made on the burial registrations. Before 1811, the Amsterdam population was buried in a Protestant church or in an ecclesiastical or urban cemetery. An exception to this were the Jews who had their own cemeteries outside the city. The churches and cemeteries had registers indicating who was buried. The oldest burial register in Amsterdam is the burial book of the 'Oude Kerk' (Dutch for 'old church') from 1553. This index on the burial registers before 1811 includes all burial entries from that year until the introduction of civil status in 1811. In 1810, the Netherlands had merged into the French Empire. At that moment the civil status was introduced, a population registration on the French model. In 1811, to loud protest, the churches had to hand in their burial registers, as well as their baptism and marriage registers, to the civil registry.

Regulations for keeping baptismal and marriage records were established at the Council of Trent (1545-1563). The Council did not rule on the keeping of burial records. This is the primary reason that, as far as registration is concerned, few regulations are known. In general, the grave diggers, who were appointed by the city government, kept the registers. In this they accounted to the church masters for the funds they were owed. As a result, these registers are mainly financial in nature. In addition, there are grave books, in which the owners of the graves were recorded. Often the grave digger wrote down the name of the deceased as it had been passed on to him when the deceased was buried. However, there was no check at all for completeness or spelling. As a result, people's names can be written in very different ways. A burial registration contains at least the following information: the name of the deceased and the address where they died. On the left side of the margin is usually the day of the funeral. Instead of a number, the abbreviation 'do' (for 'ditto') can also be mentioned. This is the Latin word for 'on the same day'. On the right in the margin is the amount (guilders and pennies) that had to be paid for the funeral. However, there are many variations on this standard burial registration. Of note is that not all this information was eventually digitized.

Although this index is quite a large one, as it contains 1.509.132 person references, it cannot be considered trustworthy. For instance, it is unclear which of the mentioned persons is buried, and which person is mentioned as partner, the one that declared the burial event. For this reason, the only role persons have in the burial event is the one of 'being registered'. Next to this, not all the registration books of burials in churches and on graveyards in Amsterdam survived over time. Although useful as an entrance to the source, one has to be careful in drawing final conclusions from (not) matching records from the burial records with other sources. Chapter 7, Domain Knowledge, is about dealing with these sorts of uncertainties. More information on the Baptism index can be found in Dutch at the website of the Amsterdam City Archives [4].

3.1.4 Notarial archives

Contrary to the previous three indices, up to 2016 no digitised index was available of the Amsterdam notarial archive. Together with the start of the Golden Agents project and for a large part co-financed by it, the City Archives started in 2016 with the indexing, and thereby

[4] https://archief.amsterdam/uitleg/indexen/72-begraafregisters-voor-1811

creating a structured version of the most important meta data of its largest early modern archival collection. From this 3,5 kilometres shelf length counting archive almost all early modern deeds have been scanned, amounting to nearly ten million scans. Approximately two million of them have been indexed on deed type, person names, date, and locations in the crowdsourcing project 'Alle Amsterdamse Akten' (Dutch for 'All Amsterdam Deeds'),[5] resulting in almost 600.000 unique deeds with 3.102.264 person references. Information that is stored in these records is:

- The type of deed.
- The name of the deed holder.
- The names of others who are mentioned in the deed. Patronymics are attached to the first name if there is a family name
- The date or year.
- A short description.
- The name of the notary.

In this dissertation, three deed types have our special attention: Last Wills (in Dutch: Testamenten), Prenuptial Agreements (in Dutch: Huwelijkse Voorwaarden) and Probate Inventories (in Dutch: Boedelinventarissen). These deeds are highly interesting because, contrary to the other mentioned indices, each deed gives insight into a complete network of family and friends. This means that, when combined with other data sets and disambiguated, a notarial archive deed can form a hub, linking together persons from these other data sets. Unfortunately, the only role of people in the registration event that is included in the index is the one of 'being registered'. The records in this index resemble the burial registration records in this respect. More information on the Notarial index can be found in Dutch at the website of the Amsterdam City Archives [6].

3.2 ECARTICO

ECARTICO[7] is a comprehensive collection of biographical data from cultural entrepreneurs, such as painters, writers, book printers, illustrators, goldsmiths and related figures from ca. 1475-1725 in the Low Countries and Dutch Republic. It aggregates information on a total of 61.011 persons (at the time of writing) from this time period and includes references to both primary (e.g. a deed from the city archives) and secondary sources (e.g. a biography), including the Amsterdam City Archives. The persons mentioned are divided among the following occupations, where the remaining number of persons consists of family members:

- Painters: 9574
- Engravers: 1426

[5] https://alleamsterdamseakten.nl/
[6] https://archief.amsterdam/uitleg/indexen/49-notariele-archieven-1578-1915
[7] https://www.vondel.humanities.uva.nl/ecartico/

- Booksellers, printers and publishers: 3383
- Gold- and silversmiths: 7342
- Sculptors: 423

Because ECARTICO has been curated by experts, it does not contain any duplicate entities. This means that this data set can provide a basis for the creation of a ground truth for person disambiguation. For instance, by purposefully creating duplicates and ambiguating the data we can create a data set with duplicate entities for which there is a known ground truth. ECARTICO is hosted by the University of Amsterdam and published as Linked Open Data in the SCHEMA.ORG vocabulary. It is included in the Golden Agents project as one of the main data sets.

3.3 Short-Title Catalogue Netherlands

The Short-Title Catalogue Netherlands (STCN) is the national bibliography of the Netherlands until 1801. It contains descriptions of more than 220,000 works that were printed in the Netherlands or in the Dutch language between 1460 and 1801. These titles are still preserved in one or more copies in a Dutch or European heritage library (a total of 638,000 copies). The STCN is compiled on the basis of collections in the Netherlands and abroad, and all titles are described by experts with the book in hand, sometimes called in autopsy. For example, in addition to the title of the book and the author(s), also information about the printer, the language, the format, the structure (collation) and typographic characteristics. Each title also receives a unique 'fingerprint', this allows users to tell different editions apart or identify works without a title page. Since everything is checked by hand, the trustworthiness of the data is very high, although books that existed for which no physical copy remains today are not included. The complete data set is open access and the metadata is available in various formats for research and re-use [8]. Of main interest to the Golden Agents project is the production side of creative goods, such as printers of books, which can be found in this data.

3.4 Occasional Poetry

This data set with Occasional Poetry published in the Dutch Republic between ca. 1600-1800 built by the Royal Library of the Netherlands (KB) contains 6.906 printed poems or collections of poems for a particular type of event, such as marriage (2,433), death (1,049), or various types of anniversaries (474). Work on the data set has been conducted by the KB,[9] and the data set was converted to linked data in the Golden Agents project. It contains bibliographical information on poems written for an event, together with information on the event's date and participants. Textual references to authors and printers/publishers were disambiguated and connected to existing resources for metadata: the Dutch Thesaurus of Author names (NTA)[10] and the STCN printer thesaurus[11] respectively. The same was done, if possi-

[8] https://data.cerl.org/stcn
[9] https://www.kb.nl/zoeken/jsru/gelegenheidsgedichten-tot-1800-nederland
[10] http://data.bibliotheken.nl/id/dataset/persons
[11] http://data.bibliotheken.nl/id/dataset/stcn/printers

ble, for persons that are mentioned in relation to the event that the poem is written for, such as the bride and groom in a marriage. In the case that these author and person references could not be found in an existing thesaurus (e.g. the NTA or Wikidata[12]), attempts were made to find mentions of these actors in archival sources in the data sets of the Amsterdam City Archives (section 3.1) and connect them accordingly by making use of a rule-based linking approach implemented in the Lenticular Lens tool [35].

3.5 Data Quality

In computer science, the concept of "garbage in, garbage out" is well known. It means that bad input will lead to bad output. Therefore data quality is an important aspect because it can significantly alter the outcomes (for the worse) of, for instance, entity resolution algorithms. The quality of the data can vary greatly between different data sets. Many were improved over time. We give an overview by categorizing the issues with data quality into three different sources:

1. *Errors are present in the data.* For instance the year an event took place can be off by many years if the last two characters in the date were mistakenly switched, e.g. 1642 to 1624. This issue can have multiple causes:

   (a) There was an error in the original handwritten document, i.e. the scribe made a mistake in writing it down. Since this was done by many different people over a large span of time, some were better at not making mistakes than others.

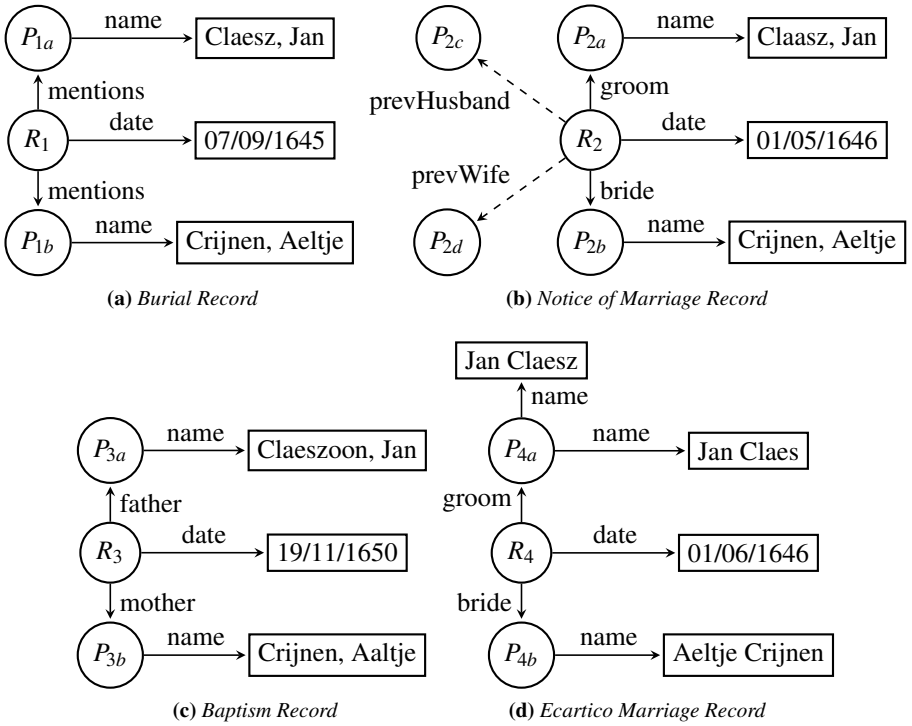
   (b) The document was incorrectly digitized. Since the documents are handwritten (in old variants of a language), HTR, i.e. handwritten text recognition, is often not very effective, especially for numbers. For this reason it is common for documents to be manually transcribed to digital form. This is done by a group of volunteers where, although this group is trained and supervised, errors can creep in.

2. *There is missing and/or ambiguous data.* This can have two causes:

   (a) The handwritten documents do not specify all relevant information. An example would be a record describing a burial registration which does not specify who has been buried. In this case, a domain- and data-expert has to infer from (sometimes cryptic) additional information who has been buried and release a new version of the data set that is less ambiguous.

   (b) Not all information was digitized. This is usually done to expedite the process, often performed by volunteers. In this case it may be beneficial to re-examine the original documents and enrich the data set with extra information. However this can be a massive undertaking, necessitating the review of millions of records. For this reason it can be that only part of the data is enriched. For example, the organizers may review all records from a certain year, then skip four years, etc. This process will effectively enrich one-fifth of the data.

[12] https://www.wikidata.org/



(a) *Burial Record* **(b)** *Notice of Marriage Record*

(c) *Baptism Record* **(d)** *Ecartico Marriage Record*

Figure 3.2: *The 2019 version of the City Archives of Amsterdam. Each registry has a number of records* $(R_1,\ldots,R_n)$*, all of which are associated with two or more persons* $(P_{1a},\ldots,P_{1z})$*. 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.*

3. *Noise is present in the data*. We distinguish this from errors. This issue can be further divided into 'valid' and introduced noise:

   (a) An example of 'valid' noise is that there is more than one valid way of spelling a name. Another example is that some dates may simply not be known exactly.

   (b) Some noise is introduced and not valid. For example, researchers might append the (correct) birth year of a person in the name attribute. This is mostly due to researchers not being familiar with (proper) database design and maintenance. This is slowly changing over time, however, and as more data sets are being shared in a Linked Open Data fashion the pressure to get it right is increasing.

Most types of noise and some types of errors (e.g. uncertainty in dates and name spelling variations) can be mitigated by performing similarity checks on attributes. This process is explained in detail in chapter 4. Issues such as contamination of attributes (e.g. appending the birth date after the name) need to be fixed at the source by data experts.

3.6 Data Structure

All data in this project are in the form of knowledge graphs (KGs) specified in the Resource Description Framework (RDF). We shall briefly explain the RDF data model. RDF is based on making statements in the form of ⟨ `subject`, `hasRelationTo`, `object` ⟩. These statements are called *triples*, and together they form a graph structure. An example of a triple is ⟨ `actor:12b43ac`, `rdfs:label`, `"Keanu Reeves"` ⟩. The subject of a triple denotes a *resource*, and can either be specified with a Uniform Resource Identifier (URI) or it can be anonymous. When no URI is given, the anonymous resource is called a *blank node*. Blank nodes are useful for purposes such as reification (e.g. provenance information) and complex attributes (e.g splitting names into surname and given name). The object of a triple can either be another resource or otherwise a literal, in which case the object models an attribute of the subject resource. There are many types of literals, for instance they can be strings or numbers, but also dates.

2019 Version

The 2019 version of the RDF data is relatively simple compared to the version that would come later. A schematic of the structure is shown in figure 3.2. There is only one type of record, although one can deduce from which registry a record originated using the associated URI. Record nodes have a date attribute, indicating when the event the record represents took place. Furthermore, record nodes have multiple links to person nodes. These person nodes each have a name attribute. The role a person had in a record is encoded in the predicate. For example, to denote a person who was the male participant in a wedding, the 'groom' relation is used between that person and the record. This can be seen in subfigure 3.2b. Some record resources have optional links to persons when applicable, such as when in a notice of marriage record a person from a previous marriage is mentioned. These optional relationships are denoted as dashed lines in figure 3.2. Overall, the data set is relatively small compared to later versions, and some persons with very common names have been removed.

2021 Version

The 2021 version of the RDF data is more complicated and makes use of a custom event based ontology. An ontology is what defines the concepts and relationships in the RDF graph. It is very important to have an ontology that suits the particular data well (e.g. bibliographical data), but it also needs to take into account how the data is used later on, such as when it is used as input in Disambed. There can be additional needs such as modeling how the data came to be and what source it originally came from. An event based ontology has been developed in the project by others to model all the data. In this ontology records describe events which each have persons participating in them in certain roles. We shall give an example below after some further elaboration. The primary reasons for using an event based ontology are that it is an accepted way of modeling (CDOC and SEM are examples of other event based ontologies), it is more generic in terms of reification, and finally it enables the easy construction of timelines for persons. Events can have additional properties such as associated religion, location and (approximate) date of occurrence. Roles have an associated

```
@prefix pnv: <https://w3id.org/pnv#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix roar: <https://data.goldenagents.org/ontology/roar/> .
@prefix sem: <http://semanticweb.cs.vu.nl/2009/11/sem/> .
@prefix thes: <https://data.goldenagents.org/thesaurus/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix deed:
    <https://archief.amsterdam/indexen/deeds/a74a2bbf-20be-4c27-9d14-d477f5c09ea3> .
@prefix personName: <https://data.goldenagents.org/datasets/personname/> .


deed:event=Event1 a thes:Begraven ;
    sem:hasTimeStamp "1789-10-24"^^xsd:date .

deed:person=99e87e23-d295-2bb2-e053-b784100a6a2e a roar:Person ;
    rdfs:label "Lucretia Wilhelmina van Merken" ;
    roar:participatesIn deed:event=Event1 ;
    pnv:hasName personName:5289fa40-1a30-5ac9-8583-6a4d01c6443a .


personName:5289fa40-1a30-5ac9-8583-6a4d01c6443a a pnv:PersonName ;
    pnv:baseSurname "Merken" ;
    pnv:givenName "Lucretia Wilhelmina" ;
    pnv:literalName "Lucretia Wilhelmina van Merken" ;
    pnv:surnamePrefix "van" .


deed:person=99e87e23-d294-2bb2-e053-b784100a6a2e a roar:Person ;
    rdfs:label "Nicolaas Simon van Winter" ;
    roar:participatesIn deed:event=Event1 ;
    pnv:hasName personName:b3f66e9f-9f64-5d45-bbe0-24343cd3a90f .


personName:b3f66e9f-9f64-5d45-bbe0-24343cd3a90f a pnv:PersonName ;
    pnv:baseSurname "Winter" ;
    pnv:givenName "Nicolaas Simon" ;
    roar:carriedBy deed:person=99e87e23-d295-2bb2-e053-b784100a6a2e ;
    pnv:literalName "Nicolaas Simon van Winter" ;
    pnv:surnamePrefix "van" .


_:role1 a thes:Geregistreerde ;
    roar:carriedIn deed:event=Event1 .


_:role2 a thes:Geregistreerde ;
    roar:carriedBy deed:person=99e87e23-d294-2bb2-e053-b784100a6a2e ;
    roar:carriedIn deed:event=Event1 .
```

Listing 1: *Example RDF Turtle syntax for a single event (Burial registration) in the subset in which two persons participate in the role of 'being registered'. Their name is described as a separate resource using the Person Name Vocabulary (*PNV*) [54].*

type such as *Mother*, *Father*, *Bride*, *Groom* and *Witness*. Persons, besides a URI, only have name information, modeled as a blank node with first and last names split into different fields.

Listing 1 gives an example of how a burial event has been modeled using the turtle syntax. In its raw form, each person URI for each event is unique. When persons are disambiguated it allows for events to be linked together. Figure 3.3 (shown at the end of this chapter) shows the

general structure of how multiple events can be connected via the disambiguated persons who participated in these events. When many events (especially events from different registries) are connected in this way, previously hidden structures should hopefully emerge, such as underlying religious groups that interact more among themselves than with other groups.

3.7 Conclusion

Semantic web technology is increasingly used by researchers in the humanities such as (cultural) historians, literary scholars, art historians to answer important - yet with traditional means often hard to answer - questions in their respective (sub) fields. Not only does this technology make it possible for the data to be more easily accessed and large scaled, it also facilitates the integration of other relevant data sources, allowing for the answering of questions that are only answerable when different data sets are combined. The conversion of existing data sets (to RDF) remains a work in progress and is done by experts of both semantic web technology and the cultural domain of the data set. Much effort is required to first clean up and then properly structure the data with a (custom) ontology. This was also the case in the Golden Agents project. Data sets were constantly improved upon by adding new registries and cleaning up attributes for the duration of the project.

In this chapter we have seen how the data from historical sources can have a multitude of issues that will need to be handled first before it can be properly processed. These issues can be errors, missing or ambiguous data, and the fact that different sources use different data structures, as seen in figure 3.2. In the next chapter, we discuss how the different knowledge graphs can be combined in preparation for the disambiguation process.

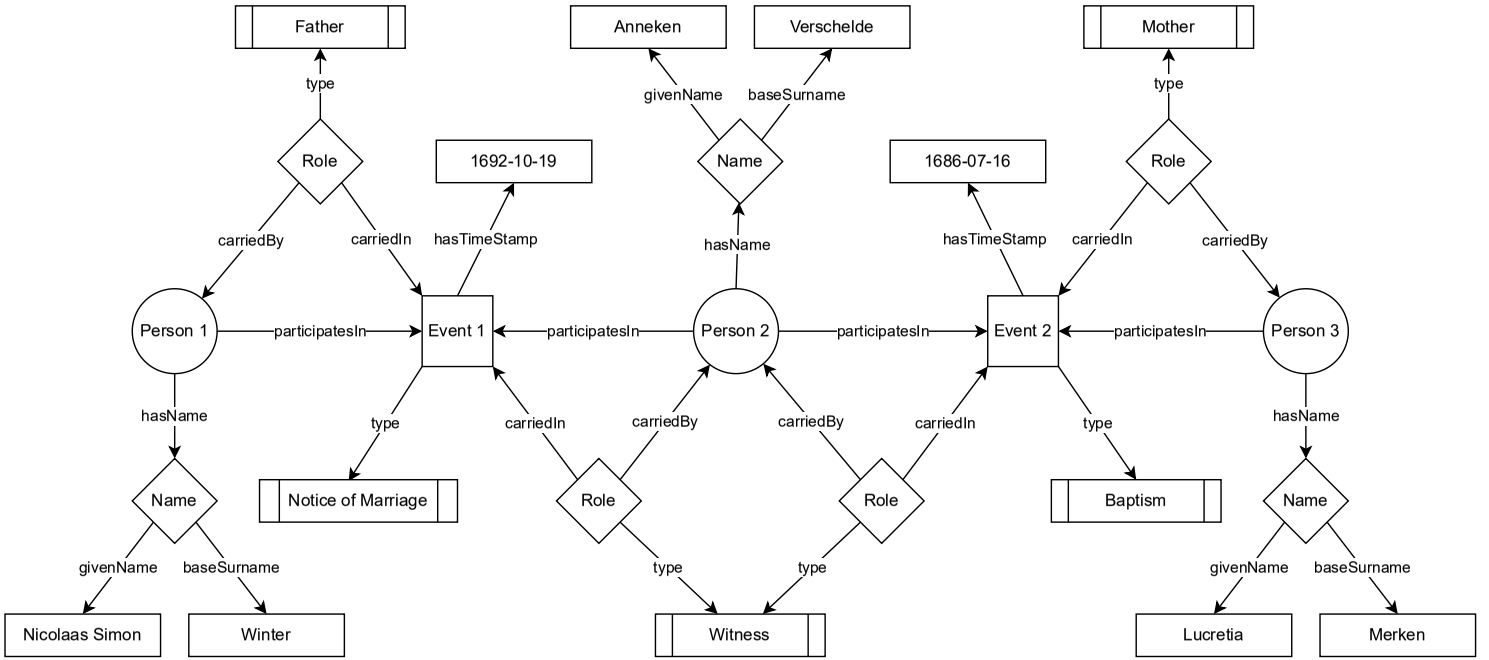


Figure 3.3: *The structure of the 2021 version of the Amsterdam City Archives in RDF. In this example there are two events and three persons participating. One person participates in both events in the same role. Roles are modeled using blank nodes. One could interpret this figure as a mother being present as a witness at both her son's wedding and the baptism of her daughter's child. Note that, in the actual data, there are usually more persons related to events than shown here, e.g. a baptism event always at least denotes the mother, father and child. Some type information has been omitted and instead been encoded in the shapes of the elements.*

4 Reconciliation

This chapter is based on the following publications: [3, 4]

- Graph Embeddings for Enrichment of Historical Data, *Workshop on Graph Embedding and Data Mining (GEM)*, 2019
- Tailored Graph Embeddings for Entity Alignment on Historical Data, *International Conference on Information Integration and Web-based Applications and Services*, 2020

In the setting of the Golden Agents project, but in general as well, each individual KG uses unique URIs for resources, such as persons, with no overlap between the different knowledge graphs, i.e. no URI is shared between knowledge graphs. This means that shared literals, such as names, are the only means of creating some sort of interconnection between entities in these knowledge graphs. However, in many cases there is no exact match between literals, such as due to spelling variations of names, to make such connections. This would cause potentially duplicate entities that are very similar but not identical in their properties to be either disconnected or have a long minimum traversal path between each other in the RDF graph. Therefore we need to create a shorter path between similar entities too. If we do this, it allows for graph traversal algorithms to reach all nodes in the merged graph which may be relevant for some particular starting entity. These entity neighborhoods can then provide the basis on which the entities can be disambiguated.

Each KG can have more than one internal duplicate for each entity, and an entity can have more than one duplicate across other knowledge graphs. For this reason, all the knowledge graphs are first combined into a single graph after which properties of entities (literals) are connected based on their similarity. We call this process reconciliation, and this chapter gives examples on how this is achieved. Some user input is necessary to specify which properties to compare to what other properties and in what way, more on this in section 4.1.5. Knowledge graphs can have many different types of properties, some of which can be merged and others compared. Literal nodes are merged when they have identical values and predicates, as shown in figure 4.1. More specifically, for every triple pair $\langle s, \mathtt{p}, \ell \rangle$, $\langle s', \mathtt{p}, \ell \rangle$, ℓ refers to the same literal node in the graph. However, for the pair $\langle s, \mathtt{p}, \ell \rangle$, $\langle s', \mathtt{p}', \ell \rangle$ it does not. The process of merging literal nodes by predicate allows the literal nodes to be compared by their

semantic content and not just by their value. That is, we only merge two literal values if both mean (approximately) the same thing. For example, we could merge birth dates with baptism dates, but not with death dates. This can be shown in figure 4.1, p could stand for the predicate `bornOn` and p' for `diedOn`, where the unfortunate person s died on the same day as they were born. Here we do not merge these two date literals, even though they have the same value.

4.1 Connecting Similar Literal Nodes

Sometimes two literals are not identical, but they are still very similar to each other. In this case we cannot merge them, but we still want to connect them to each other with a new edge. An example would be to connect similar names to each other, such that persons with similar names are in each others neighborhood. We explain this concept further in chapter 5. To support the wide range of values that literal nodes can have, and how they can be compared, we provide an array of methods to compare them. The domain expert selects a number of similarity functions $\sigma(\ell,\ell')_{p,p'} \in (0,1]$, each associated with a predicate pair (p,p'). For example, p could stand for the predicate `bornOn` and p' for `baptisedOn`. Note that p and p' are not necessarily different predicates, they could both refer to `hasName`, in which case names are compared to all other names. Then, all combinations of the form $(\langle s,p,\ell\rangle,\langle s',p',\ell'\rangle)$ are compared. Each similarity function is given a threshold value t, and when the computed similarity score exceeds t, a new bi-directional edge is added to the graph between ℓ and ℓ', with the similarity score as weight, as shown in figure 4.2. We describe per literal data-type how they are compared.

4.1.1 A Generic Similarity Function

In many cases, the similarity between two literals can be seen as a function of some numerical distance between them. This captures many types of literal comparisons, such as numerical (e.g. ages) and dates (e.g. birth dates). The function that describes the similarity between two literals ℓ and ℓ' is defined as:

$$\sigma(\ell,\ell') = \frac{1}{(||\ell-\ell'|-\delta|+1)^{\alpha}} \tag{4.1}$$

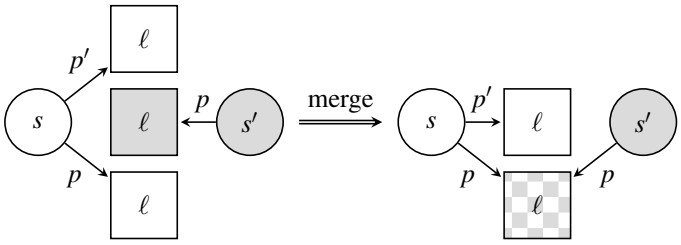


Figure 4.1: *Two disjoint RDF graphs (gray and white) have their literal (square) nodes merged based on predicate information. In this example, the three literal nodes have identical values, but only two are merged because they share the predicate p.*

Note that $\sigma(\ell,\ell') = 1$ if $||\ell-\ell'|-\delta| = 0$ and that $\lim_{|\ell-\ell'|\to+\infty}\sigma(\ell,\ell') = 0$. That is, if there is no difference between two literals their similarity is 1, and as the difference between two literal values grows to infinity, their similarity approaches zero. The parameter δ can be set to a non-zero value if the two literals ℓ and ℓ' have to be δ distance apart for $\sigma(\ell,\ell')$ to be close to 1. This is useful for when one of the literal values acts as a proxy for another predicate. For example, we may be comparing birth dates with baptism dates. In this case, we treat the baptism date as a proxy for a birth date. Since baptisms occur several days after a birth, we can use $\delta = 3$ to make sure that a birth and baptism date are identical (i.e. $\sigma(\ell,\ell') = 1$) when the latter occurs 3 days after the former. Furthermore, α can be used as a smoothing factor by setting it to some positive value below 1. The intuition is that for a given value of $||\ell-\ell'|-\delta|$, as α gets closer to 0, the similarity increases. In other words, the similarity between to literals does not change as fast as their distance increases with when α is close to 0. In practice, it can be difficult to determine the correct value for α. Therefore the user can define a distance d at which the similarity $\sigma(\ell,\ell')$ is equal to the aforementioned threshold value t. The correct value for α can be calculated using equation 4.2, assuming t and d are positive.

$$\alpha = -\frac{\log(t)}{\log(1+d)} \tag{4.2}$$

For example, when we configure that for two numerical literals to have a similarity equal to a threshold of 0.9 if the difference $||\ell-\ell'|-\delta|$ is equal to 3, then the associated value for α is $-\log(0.9)/\log(1+3) \approx 0.076$.

4.1.2 Date Literal Nodes

The similarity between date literal nodes is computed with equation 4.1. In the case of dates we include an additional parameter. That is, $|\ell-\ell'|$ can be set to mean the number of years, months or days between the two dates. As explained before, the value for δ comes in useful here, for example when a birth-date is unavailable but we do know the date of baptism. Using the knowledge that, on average, it takes about three days for a baby to be baptised, we set $\delta = 3$ and the difference between dates to be counted in days. We can extend this comparison with even more domain knowledge, for we know that non-stillborn babies are always baptised after they are born. Therefore we set the comparison to only give a non-zero value if the date associated with the *birth-date* predicate is before the one for the *baptism-date* predicate.

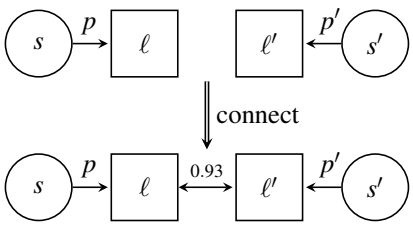


Figure 4.2: *Two literal nodes are connected by comparing their values. If the similarity between ℓ and ℓ' exceeds a threshold, a new bi-directional edge is added and the calculated similarity value is used as a weight.*

4.1.3 String Literal Nodes

Since strings can be used for many different purposes ranging from names to titles and even pieces of text, we support a range of similarity functions in our framework:

- Levenshtein: The minimum number of single-character edits required to change literal ℓ into literal ℓ'.
- Jaro-Winkler: Designed specifically for duplicate detection, most useful for purposes such as matching person names.
- N-gram Jaccard: $\sigma(\ell,\ell') = \frac{|A \cap B|}{|A \cup B|}$, where A and B are the sets of n-grams that occur in literal nodes ℓ and ℓ' respectively. An n-gram is a sequence of n characters. For example, the string 'Jans' contains two n-grams 'Jan' and 'ans' for $n = 3$. Useful when repetitions should not affect the similarity. The size of the n-grams can be set by the user.
- N-gram Cosine: $\sigma(\ell,\ell') = \frac{A \cdot B}{||A|| \cdot ||B||}$, where A and B are the n-gram frequency vectors of literal nodes ℓ and ℓ' respectively.
- Token Jaccard: similar to n-gram Jaccard, but sets A and B are now tokens (separated by spaces). This can be useful for cases where spelling errors are unlikely.
- Token Cosine: the cosine similarity mentioned above, but the components of vectors A and B are the frequencies of tokens rather than n-grams.

4.1.4 Preliminary Alignments

Our method does not necessitate that the ontologies be aligned first, however a preliminary alignment will prevent making superfluous comparisons later on. We now show an example where a preliminary alignment is beneficial. Let's say we have two KGs which both contain name information, where the first KG uses the predicate `hasName`, while the second uses `name`. In this case we would have to compare the set A of all literals that have the `hasName` predicate to the set B of all literals that have the `name` predicate, with a total number of $|A| \cdot |B|$ comparisons for $|A|$ distinct names in the first KG and $|B|$ distinct names in the second. In addition we would still have to internally compare both the `hasName` and `name` literals, each with $\frac{1}{2}(k-1)k$ comparisons, where k is the number of literals. In total we require $\frac{1}{2}(|A|+|B|-1)(|A|+|B|)$ comparisons. However, changing the predicate `name` to `hasName` in the second KG will yield $\frac{1}{2}(|A \cup B|-1)|A \cup B|$ comparisons, where $|A \cup B|$ is the number of distinct literals after merging, which in general is much smaller than $|A|+|B|$.

4.1.5 User Input

The user can specify which literals need to be compared and in what way. In listing 2 we give an example of how this is done. In the example we first compare the full names of all resources of type saa:Person using the LSH algorithm (explained in the next section) on ngrams of length 3. Therefore we will create a new edge between two literals if their similarity is greater than the threshold of 0.9. Additionally we also compare the first- and family

-

```
similarity:
  - sourceType: saa:Person
    targetType: saa:Person
    sourcePredicate: saa:full_name
    targetPredicate: saa:full_name
    method: LSH_NGRAM_COSINE
    ngram: 3
    threshold: 0.9
    buckets: 100
    bands: 1
  - sourceType: saa:Person
    targetType: saa:Person
    sourcePredicate: saa:first_name
    targetPredicate: saa:first_name
    method: LEVENSHTEIN
    threshold: 0.9
  - sourceType: saa:IndexOpOndertrouwregister
    targetType: saa:Marriage
    sourcePredicate: saa:prenuptial_marriage_date_approx
    targetPredicate: saa:marriage_date_approx
    method: DATE_DAYS
    pattern: yyyy-MM-dd
    time: FORWARDS
    threshold: 0.5
    thresholdDistance: 4
    offset: 18
  - sourceType: saa:Person
    targetType: saa:IndexOpBegraafregistersVoor1811
    sourcePredicate: saa:death_date_approx
    targetPredicate: saa:burial_date_approx
    method: DATE_DAYS
    pattern: yyyy-MM-dd
    time: FORWARDS
    threshold: 0.5
    thresholdDistance: 2
    offset: 2
```

Listing 2: *In this example configuration the names of all entities that have the type* `saa:Person` *will have their names compared in two different ways. First, Locality Sensitive Hashing is used to efficiently make selections of full names with the use of ngrams of size 3. The comparisons themselves are done with a fast bit-wise Levenshtein algorithm.*

names of all resources of type saa:Person, this time using the Levenshtein string comparison algorithm. Next we compare the dates of notice of marriage events with the dates of marriage events. Note that we can specify the pattern and set the time to `FORWARDS`, this means that to be similar marriage events have to occur after notice of marriage events. If the marriage event precedes the notice of marriage event the similarity will be 0 by default. When there is a difference of 18 days between the two events the similarity will be 1 (i.e. at its maximum). Finally we compare death dates to burial dates. This time the dates have to match more closely, with a threshold distance of only two days. Alternative possible values for time direction are `BACKWARDS` and `BIDIRECTIONAL`.

4.2 Efficient Comparison

A major bottleneck time-wise is the quadratic growth in the number of comparisons necessary when reconciling multiple large KGs. That is, when the number of literals to compare grows

by a factor of n, the number of necessary comparisons grows by n^2. The baptism archive (section 3.1.2), for instance, contains millions of records. In practice this requires many millions of comparisons, for which only very few will yield new links. It is therefore beneficial to somehow reduce the number of comparisons done by, for instance, only comparing literals that are likely to be similar. For this purpose we apply a technique called Locality Sensitive Hashing (LSH).

The intuition behind LSH [42] is best described by contrasting it to the general concept of hashing. A hashing function is typically described as a one-way function, i.e. one cannot predict the input when given only the output. The input can be of any size and the output is of some fixed size. Additionally, when the input changes slightly this will usually yield a completely different output (hash). This means knowing the output hash value will tell you nothing about what the input was, a property that is highly valued in the application of hashing in cryptography, such as storing passwords securely. Since the input space is much larger than the output space, it can happen that multiple inputs all map to the same output. This is called a collision. When two different inputs collide, we say that they fall into the same bucket. Depending on the application, collisions are generally something to be avoided or reduced. In contrast, the idea behind LSH is that we would like inputs that are similar to map to the same output, i.e. fall into the same bucket. It is then easy to reduce the number of comparisons by first checking in which bucket(s) a hash falls, and only comparing it to other values that fall in the same bucket. We can influence the probability of two inputs that are similar to fall into the same bucket by changing the total number of available buckets b and an additional parameter called the number of stages (sometimes called bands). There is a set of b buckets for each stage, and a hash falls into a single bucket for every stage. This means that if there are s stages, a hash falls into s buckets.

We shall briefly explain how the LSH analysis is performed, also shown as pseudo code in algorithm 1. First, in the `construct` procedure (line 1), literals are represented with binary vectors, where each component of the vector is an indicator for the presence of an n-gram (line 10). Since we are dealing with millions of literals, the indicator vectors are stored in a dynamic binary compressed sparse row (CSR) matrix I. Each row represents a distinct literal value (such as a name) and the non-zero columns are the n-grams that occur in that value. The compressed sparse row representation of a sparse matrix allows for very fast lookup of rows but is less efficient when retrieving columns. We allow the matrix to grow dynamically as previously unseen n-grams are discovered when adding new rows. After the n-gram indicator matrix is built we can create a new binary CSR matrix B of size $k \times sb$ where the non-zero columns are, for each hash, in which buckets it has fallen and k is the number of distinct literals. It is now easy to, for a given literal, find which other literals fell into the same bucket(s). This is shown in the `FIND` procedure (line 14). We only have to find the non-zero columns in the corresponding row, which represent the relevant buckets, then all other rows that also have a non-zero value in any of those columns are candidates to be compared. It is important to be very efficient in the lookup process, otherwise we are simply exchanging time spent on making a large number of comparisons to time spent on finding candidates for comparison. Therefore, due to the inefficiency of column lookup in CSR, we also store the transpose of the bucket matrix B', where the non-zero columns in B are queried as rows in B', as shown in figure 4.3.

Algorithm 1 Perform LSH

procedure CONSTRUCT(L, n, b, s)
    Argument L is set of distinct literals, such as names
    Argument n is size of n-grams
    Argument b is the number of buckets
    Argument s is the number of stages
    $k \leftarrow |L|$
    $P \leftarrow$ *all possible n-grams in L*
    $I \leftarrow$ *n-gram indicator matrix*
    for each literal $\ell \in L$ **do**
        $T \leftarrow$ *set of n-grams present in* ℓ
        $v_\ell \leftarrow (v_p)_{p \in P}$ such that $v_p = 1$ if $p \in T$ and $v_p = 0$ if $p \notin T$
        $I.appendRow(v_\ell)$
    $B(k, sb) \leftarrow$ *bucket allocation matrix*
    for each indicator vector $v_\ell \in I$ **do**
        $h_\ell \leftarrow LSH(v_\ell)$, *bucket allocation for indicator vector* v_ℓ
        $B.append(h_\ell)$
    return B
procedure FIND(B, i)
    Argument B is the bucket allocation matrix
    Argument i is the index of the literal to compare
    $B' \leftarrow$ *transpose of B*
    $r_i \leftarrow B.row(i)$, *set of non-zero column indexes*
    $j \leftarrow \emptyset$, *set of indexes of literals to compare to*
    for each non-zero column index $c \in r_i$ **do**
        $r_c \leftarrow B'.row(c)$
        $j \leftarrow j \cup r_c$
    return j

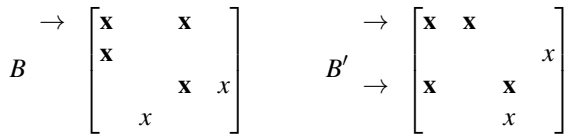


Figure 4.3: *Efficiently finding other literal values that are likely to be similar. In this example we query the first row in B and see that the first and third columns are non-zero, i.e. it fell into these two buckets. Now we query the first and third rows in B' to find the indexes of other candidate literals that fell in at least one of these buckets, namely those mapped to columns two and three. Note that in practical applications, these matrices can become very large.*

A final challenge is tuning the parameters b and s to keep the size of each bucket as small as possible (to reduce the number of candidates) while still finding (almost) all values that have high similarity. From experiments where we compare the LSH method to normal exhaustive comparison, we are able to find more than 90% of relevant matches with 10 buckets and two to three bands, while reducing the time taken three- to fourfold.

4.3 Conclusion

In the setting of the Golden Agents project, but in general as well, each individual KG uses unique URIs for resources, such as persons, with no overlap between the different knowledge graphs, i.e. no URI is shared between knowledge graphs. This means that shared literals, such as names, are the only means of creating some sort of interconnection between resources in these knowledge graphs. However, in many cases there is no exact match between literals, such as due to spelling variations of names, to make such connections. This would cause potentially duplicate entities that are very similar but not identical in their properties to be either disconnected or have a long minimum traversal path between each other in the RDF graph. Therefore we need to create a shorter path between similar entities too. If we do this, it allows for graph traversal algorithms to reach all nodes in the merged graph which may be relevant for some particular starting entity. These entity neighborhoods can then provide the basis on which the entities can be disambiguated.

In this chapter we have shown how the different knowledge graphs can be combined efficiently with a wide support of options for all the different data types and also for the varied semantics of predicates. In the next chapter we will discuss how the nodes in the combined knowledge graph are each embedded into a space, based on overlap between their respective neighborhoods. This space, called an embedding, will then be used further as a source of information for calculating the similarity between pairs of entities.

5 Embedding RDF Nodes

This chapter is based on the following publications: [2–4]

- Graph Embeddings for Enrichment of Historical Data, *Workshop on Graph Embedding and Data Mining (GEM)*, 2019
- Tailored Graph Embeddings for Entity Alignment on Historical Data, *International Conference on Information Integration and Web-based Applications and Services*, 2020
- Entity Matching in Digital Humanities Knowledge Graphs, *Computational Humanities Research*, 2021

Embeddings are a well known technique for creating numerical representations for entities that are otherwise difficult to process, such as words in text or, as in our case, nodes in a graph. When we assign a numerical vector representation to an entity, we say that we embed that entity. In this chapter we discuss how relevant nodes in an RDF graph (such as all nodes of `type:Person`) are embedded based on their graph neighborhood structure. A useful feature of embeddings is that, when constructed appropriately, the similarity between entities can be calculated (in parallel) using Euclidean distance or cosine similarity, enabling further processing such as clustering entities based on their similarity.

5.1 Context Generation

In order to generate the features on which the embedding algorithm will work, we devise a strategy in which a neighborhood is created for each relevant node in the graph, which consists of a set of weighted nodes that can be reached from that node. For a given (RDF) knowledge graph $G = (V, E)$, the edges E are directed and associated with a predicate. As in Node2Vec, we define $N_S(i) \subset V$ as the neighborhood of node $i \in V$ generated through a neighborhood sampling strategy S. The weight given to each node in the neighborhood depends on the distance from the starting node, the network structure, and predicate weights given by a domain expert. Having multiple paths to the same node will increase this weight. The output of the neighborhood sampling strategy is an (ideally sparse) co-occurrence matrix X of size

$|V| \times |V|$ where X_{ij} is a non-zero positive real number iff node j occurs in the neighborhood of node i. The co-occurrence matrix X is then used as input for the GloVe algorithm.

Like Cochez et al. [18] we use an adaptation of the Bookmark Coloring Algorithm [9] (BCA) to generate features for each node in an RDF graph. The general idea of BCA is to consider the neighborhood of a node as an approximation of the personalized PageRank [13] 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 remains on the node and the fraction $1-\alpha$ is distributed among neighboring nodes using one of the strategies mentioned in section 5.1.1. When after a certain point 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 user can adjust α to influence how rapidly paint is diminished at each step, and ε can be adjusted in order to get a smaller or larger neighborhood. Values in the range $[10^{-1}, 10^{-4}]$ for α and $[10^{-4}, 10^{-8}]$ for ε seem to work best for various applications. Having a very small ε can cause the neighborhoods to be too large and contain many unimportant nodes, which causes memory issues later on. Algorithm 2 contains the pseudo-code of the BCA taken from [9], with some adjustments. The algorithm maintains a queue Q with pairs (i, r_i), where i is a node, and r_i is the amount of paint that node has received since it was last visited. The elements in Q are ordered by their respective paint values r_i. Initially Q only contains the start node s, with a unit amount of paint (line 6). Now the following steps are repeated:

1. An element (i, r_i) is polled (selected and removed) from Q (line 8).
2. A fraction α of the paint r_i it has received since the last visit is added to the paint p_i collected by node i (line 9).
3. If $r_i > \varepsilon$, the remaining $(1-\alpha)$ fraction of r_i is distributed among its neighbors $N_S(i)$ according to traversal strategy S (explained in the next section). Each neighbor j receives an amount proportional to the weight w_{ij} between i and j. This paint is added to the paint the neighbour had already received since it was last visited (line 15 or 17).

When the queue is empty, the algorithm terminates and returns the amount of paint that has been collected by each node.

5.1.1 Traversal Strategies

The RDF graph can be traversed in different ways to generate different neighborhoods depending on the requirements of the user and underlying structure of the graph. For some graphs, such as the Amsterdam City Archives graph, literal nodes, such as those representing names, act as hubs linking together concepts from multiple sub-graphs that represent, for example, different data sets. In other cases the edges can be distributed much more equally among nodes. We consider three different types of neighborhood $N(t,i)$ with $t \in \{d, a, b\}$:

- *Directed*, $N(d,i) = \{j : (i,j) \in E\}$: Distribute paint among all outgoing edges according to weights given by a domain expert.
- *Anti-directed*, $N(a,i) = \{j : (j,i) \in E\}$: Distribute paint among all incoming edges according to weights given by a domain expert. Sometimes the directed strategy is

combined with the anti-directed strategy. This creates a node neighborhood that contains all nodes leading up to and out of a starting node up to a certain distance. The reasoning behind this is that it is analogous to having a symmetric window when creating word embeddings [18].

- *Bi-directed*, $N(b,i) = \{j : (i,j) \in E \vee (j,i) \in E\}$: Ignore the edge directions and distribute paint based only on the weights given by the domain expert. This means that all incoming and outgoing edges are considered. This can be useful if the direction of edges in the network prevents certain important nodes from being visited from some starting nodes. This strategy is the default strategy used in our experiments. This strategy can be extended by adding the condition that edges that return to the previous node are not considered. Although this extension breaks with the default BCA behaviour, it can lead to much faster running times and very similar performance when the RDF data consists of many small connected components.

We use these different neighborhoods to define three different context generation strategies:

1. *D+A*: In this case the algorithm is run twice, once with directed neighborhood and once with anti-directed neighborhood. The two co-occurrence matrices are then summed together.

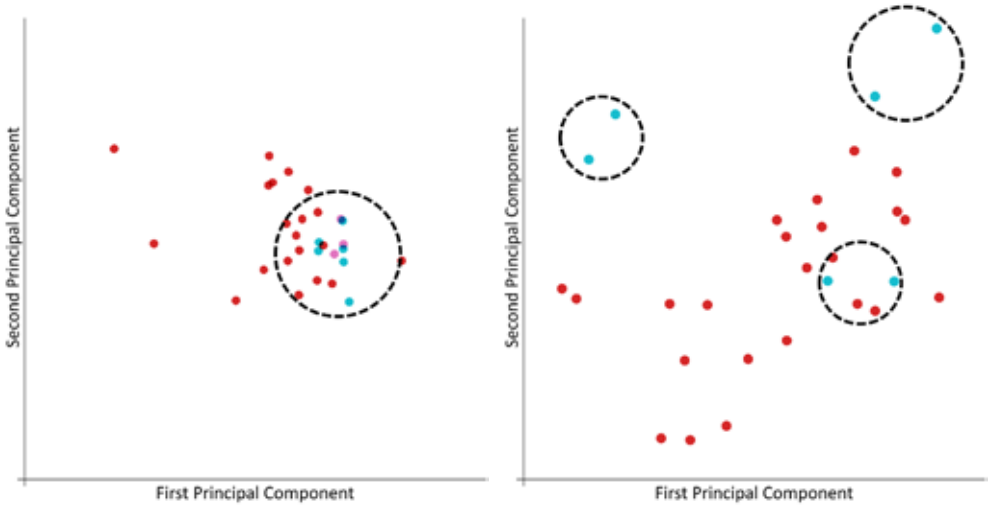
2. *Bi-directed*: The algorithm is run with bi-directed neighborhood.

Algorithm 2 BCA(b, α, ε, S)

b: *starting node (bookmark)* ▷ Arguments
α: *fraction of paint to distribute*
ε: *minimum paint on node*
S: *neighborhood sampling strategy*

$p \leftarrow 0$: *initialize p to zero vector* ▷ Variables
$Q \leftarrow \{(b,1)\}$: *drop unit amount of paint on the starting node*

while $Q \neq \emptyset$ **do**
  $(i, r_i) \leftarrow Q.poll()$ ▷ poll element (i, r_i) with largest r_i
  $p_i \leftarrow p_i + \alpha \cdot r_i$
  if $r_i < \varepsilon$ **then**
    next
  for all $j \in N_S(i)$ **do**
    $\Delta r_j \leftarrow (1-\alpha) \cdot r_i \cdot \frac{w_{ij}}{\sum_{k \in N_S(i)} w_{ik}}$
    if $(j, r_j) \in Q$ **then**
      $r_j \leftarrow r_j + \Delta r_j$
    else
      $Q \leftarrow Q \cup \{(j, \Delta r_j)\}$
return p



(a) *Directed* **(b)** *Bi-directed*

Figure 5.1: *PCA analysis of a subset of a particularly frequent name in the Amsterdam City Archives: Jan Jansen. Each data point represents the mention of a person named Jan Jansen in some record of the archives. Red points are from the baptism archive, cyan points are from the burial archive, while red points are from the notice of marriage archive. The three pairs of cyan points on the right are known duplicates in the burial archives. Note how they are much better distinguished using the bi-directed graph traversal strategy. Both directed strategies proved insufficient for generating overlapping contexts. In this example all weights for relations have been set to 1.*

3. *Hybrid*: As in the case of *D+A*, the algorithm is run twice. The difference is that, for literal nodes, we use the *bi-directed* neighborhood.

With the help of an embedding visualization tool[1] we can show how the choice of neighborhood traversal strategy influences the final result. Figure 5.1 shows how we can improve the clustering of persons in the Amsterdam City Archives by using the right graph traversal strategy for the problem. Since the Amsterdam City Archives is an agglomeration of multiple data sets which are interconnected via literal nodes, the *bi-directed* strategy generates more representative neighborhoods by traversing more of the graph while at the same time not stopping when it encounters a node with solely outgoing or incoming edges.

5.2 GloVe

Just like Word2Vec, GloVe [52] has been established as an effective method of embedding words in a corpus in such a way that the word vectors capture meaning in vector space. Unlike Word2Vec, it first aggregates all local local patterns into global statistics and afterwards continues to work with the aggregate. GloVe does this by first scanning through the entire

[1] `https://projector.tensorflow.org`

corpus and, for each word, count how many times other words occur within a given distance. An efficiency boost is obtained as GloVe only needs to scan the corpus once. The result is a so called co-occurrence matrix X, where X_{ij} signifies the number of times word j occurs in the context of word i. Since most pairs of words do not co-occur at all, X is usually a sparse matrix.

$$J = \frac{1}{2}\sum_{i,j=1}^{D} f(X_{ij})\left(b_i + \bar{b}_j + w_i^T \bar{w}_j - \log X_{ij}\right)^2 \tag{5.1}$$

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{5.2}$$

As Pennington et al. shows [52], calculating word vectors can be achieved by minimizing the loss function 5.1, where D is the size of the vocabulary, b_i and $\bar{b}_j$ are the bias terms of word i and j respectively, and w_i and $\bar{w}_j$ are the focus- and context vectors for words i and j. The weighting function f, shown as equation 5.2, is used to curb both the influence of very common words and unreliable rare co-occurrences. Both x_{max} and α are hyper parameters but the authors conclude from experiments that values of $\frac{3}{4}$ for α and 100 for x_{max} work well in practice.

Optimizing GloVe for Knowledge Graphs

Although GloVe has been established as an effective method, it is mainly used for embedding words from a text corpus in a vector space. Cochez et al. [18] were the first to make use of GloVe, without altering it, to embed nodes from a knowledge graph. However, they have not taken into account that knowledge graphs have different types of nodes. We have adapted GloVe to only embed certain types of (focus) nodes, such as URI nodes of type person, while still allowing all (context) nodes, such as literals, to act as context. Originally 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

$$J = \frac{1}{2}\sum_{i}^{N}\sum_{j}^{|V|} f(X_{ij})\left(b_i + \bar{b}_j + w_i^T \bar{w}_j - \log X_{ij}\right)^2 \tag{5.3}$$

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, the co-occurrence matrix X is assumed to be square. However, as mentioned in section 5.1, 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 $\bar{w}$. Similarly there are N elements b_i and $|V|$ elements $\bar{b}_j$. When the change between iterations in the cost function 5.1 falls below a user set threshold, we regard the algorithm to have 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.

5.2.1 Learning Embeddings

In this section we shall briefly explain how the embedding is learned in our setting. Algorithm 3 gives an overview of the process. As inputs we have the co-occurrence matrix X from the output of the bookmark coloring algorithm, the number of dimension d we wish the embedding to have, the maximum change δ_{max} between iterations before we regard the algorithm as converged, and finally the maximum number of iterations t_{max}. We recommend to set the number of dimensions to a value between 50 and 300 for best performance. When dealing with a large number of entities that need to be embedded, using a lower value helps reduce memory footprint at the cost of some potential performance loss. The maximum change between iterations δ_{max} is usually set at 10^{-6}, and the maximum number of iterations t_{max} defaults to 1000. The algorithm is supposed to converge and therefore terminate before t_{max} is reached, this hyperparameter exists therefore mainly to guarantee termination. In practice it is also useful to guard against issues such as rapidly increasing output values of the cost function, which may indicate problems with the learning rate being too large. Not included in the pseudo code in algorithm 3 are guards that check for numerical instability and terminate the learning process with some helpful messages to the user that explain the issue and what they might do to solve it.

Partial Derivatives of GloVe

The technique of gradient descent is a well researched topic for numerical optimization, i.e. finding the specific parameters for which a function is minimized, given some fixed input data. The main idea is to tweak the parameter values in a series of iterations, where each iteration should bring you closer to the preferred parameters that coincide with the minimum. It does this by taking the partial derivative of each parameter with respect to the loss function, in essence creating a 'landscape' where valleys indicate configurations of parameters that yield a low overall error. Ideally one would find the global minimum of this landscape, which would coincide with the optimal configuration of parameters. An issue with gradient descent is, however, the problem of local minima. Since gradient descent cannot view the entire landscape all at once, but instead does small steps based on the local gradients, it can get stuck in a sub-optimal state and is then not able to get out. Another issue with standard gradient descent is that it can be very slow to converge to a minimum when the slope is shallow, as the magnitude of the update is dependent on the size of the slope (gradient). Many variants of gradient descent have been proposed, and most try to deal with these issues in some way.

In order to find the optimal values of w, $\bar{w}$, b and $\bar{b}$ for a particular node in the RDF graph, we need to take their respective partial derivatives. Our modifications to the cost function in equation 5.3 do not alter these derivatives. The partial derivatives are listed in equations 5.4, 5.5 and 5.6. Let J_{ij} denote an arbitrary term from the sum in equation 5.3.

$$\frac{\partial J_{ij}}{\partial w_i} = f(X_{ij})(b_i + \bar{b}_j + w_i^T \bar{w}_j - \log X_{ij})\bar{w}_j \tag{5.4}$$

$$\frac{\partial J_{ij}}{\partial \bar{w}_j} = f(X_{ij})(b_i + \bar{b}_j + w_i^T \bar{w}_j - \log X_{ij})w_i \tag{5.5}$$

$$\frac{\partial J_{ij}}{\partial b_i} = \frac{\partial J_{ij}}{\partial \bar{b}_j} = f(X_{ij})(b_i + \bar{b}_j + w_i^T \bar{w}_j - \log X_{ij}) \tag{5.6}$$

With these equations we can calculate the gradient for each parameter and plug them into a gradient descent algorithm. Note that there are many commonalities between the partial derivatives, allowing for efficient re-use of values during their calculation.

Algorithm 3 Learn Embedding

X: *co-occurrence matrix* ▷ Arguments
d: *number of dimensions in embedding*
δ_{max}: *maximum change between iterations before convergence*
t_{max}: *maximum number of iterations*

W: *focus vectors* ▷ Variables
$\bar{W}$: *context vectors*
b: *focus bias terms*
$\bar{b}$: *context bias terms*
$t \leftarrow 0$: *iteration number*

Initialize each element in $W, \bar{W}, b, \bar{b}$ with a random number in range $\left(-\frac{0.5}{d}, \frac{0.5}{d}\right)$

repeat
  $c^{(t)} \leftarrow J(X, W^{(t)}, \bar{W}^{(t)}, b^{(t)}, \bar{b}^{(t)})$ ▷ Cost at iteration t

  for each non-zero element $X_{ij} \in X$ **do**
    $w_i^{(t+1)} \leftarrow \Psi(X_{ij}, w_i^{(t)}, \bar{w}_j^{(t)}, b_i^{(t)}, \bar{b}_j^{(t)})$ ▷ update vectors and bias terms
    $\bar{w}_j^{(t+1)} \leftarrow \Psi(X_{ij}, w_i^{(t)}, \bar{w}_j^{(t)}, b_i^{(t)}, \bar{b}_j^{(t)})$ ▷ where Ψ is one of algorithms
    $b_i^{(t+1)} \leftarrow \Psi(X_{ij}, w_i^{(t)}, \bar{w}_j^{(t)}, b_i^{(t)}, \bar{b}_j^{(t)})$ ▷ 4, 5, 6 or 7
    $\bar{b}_j^{(t+1)} \leftarrow \Psi(X_{ij}, w_i^{(t)}, \bar{w}_j^{(t)}, b_i^{(t)}, \bar{b}_j^{(t)})$

  $c^{(t+1)} \leftarrow J(X, W^{(t+1)}, \bar{W}^{(t+1)}, b^{(t+1)}, \bar{b}^{(t+1)})$ ▷ Cost at iteration t + 1

until $(t+1) > t_{max}$ **or** $|c^{(t)} - c^{(t+1)}| \leq \delta_{max}$

return normalize $\left(\frac{W+\bar{W}}{2}\right)$

5.2.2 Stochastic Gradient Descent Optimization Methods

The objective function of GloVe is usually minimized with the use of Adaptive Gradient (AdaGrad [22]) method, as Pennington et al. [52] did in their original work. There exist many alternative gradient descent methods to AdaGrad, and some may be faster and/or give better solutions, so we have experimented with three other popular gradient descent algorithms. We will explain these algorithms in this section.

Each algorithm shows a single update step of the parameters Θ. In our case, for a non-zero element $X_{ij} \in X$, these parameters are the elements in the focus vector w_i, the elements in the context vector $\bar{w}_j$, the focus bias term b_i and the context bias term $\bar{b}_j$. Performing updates on single observations in the data set is called *stochastic* gradient descent. Stochastic gradient descent can be much faster since classic batch gradient descent can perform redundant updates for similar examples in the case of large data sets. Furthermore, this approach allows for easy parallelization as all update operations for non-zero cells can be equally distributed to different threads. A possible issue with this approach is that due to the lack of locking and synchronization some calculations may be done with outdated values, also called a thread collision. However, Recht et al. [58] have shown with their algorithm called *Hogwild!* that, especially with sparse input data, meaning an update will only affect a small part of the variables, this is not a issue. In fact, omitting the usual locking and synchronization steps makes their method outperform those methods that do keep track of them.

Adaptive Gradient

Duchi et al. [22] have developed AdaGrad, which uses a separate learning rate for each parameter based on frequency of features, making it well-suited for sparse input data, such as the sparse co-occurrence matrix X in equations 5.1 and 5.3. The authors describe this as allowing them to find needles in haystacks in the form of very predictive but rarely seen features. In simple terms, their procedures give frequently occurring features, such as in our case a node in the RDF graph that occurs in the context of many other nodes, very low learning rates, and infrequent features high learning rates. The intuition here is that each time an infrequent feature is seen, the learning algorithm should take notice. Thus, the adaptation facilitates finding and identifying very predictive but comparatively rare features.

Algorithm 4 shows how this is a relatively simple parameter update scheme. On line 4 we can see that we only need to store the cumulative sum of the squared gradient for each parameter $\theta \in \Theta$. These sums are initialized as 0 on the first update step, when there is no previous iteration. The main distinction of AdaGrad compared to the original gradient descent is shown on lines 7 & 8. There we first, for a parameter θ, update the cumulative sum of the previous squared gradients of θ. Then θ itself is updated by subtracting the current gradient g_θ, modified by $\frac{\alpha}{\sqrt{v_\theta}+\varepsilon}$. From this we can gain a more intuitive understanding of why infrequent features get high learning rates. That is, frequently occurring features will have a much higher squared cumulative sum of their respective gradients. This, in turn, will depress the rate at which their respective parameters are updated. The same is true in reverse for the infrequent features, their respective parameters will have a greater update step when they are observed. Finally, the constants α and ε are usually set to 0.01 and 10^{-7} respectively

Algorithm 4 Adaptive Gradient

1: α: *learning rate* ▷ Arguments
2: ε: *'fuzz factor', small number mainly to prevent division by zero*
3: $\Theta^{(t)}$: *set of parameters to update* ▷ Variables
4: $v_\theta^{(t-1)}$: *cumulative sum of past squared gradients for parameter* θ*, or 0*
5: **for each** parameter $\theta^{(t)} \in \Theta^{(t)}$ **do**
6: $g_\theta^{(t)} \leftarrow \frac{\partial J}{\partial \theta^{(t)}}$ ▷ gradient of parameter θ
7: $v_\theta^{(t)} \leftarrow v_\theta^{(t-1)} + [g_\theta^{(t)}]^2$ ▷ add squared gradient to cumulative sum
8: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \frac{\alpha}{\sqrt{v_\theta^{(t)}+\varepsilon}} g_\theta^{(t)}$ ▷ update parameter

by default in online sources [2]. In practice, some tuning may be necessary. The authors of AdaGrad make no mention of optimal values for these hyperparameters, as they are dependent on the specifics of the data.

An issue with AdaGrad is the fact that the cumulative sums v_θ grow monotonically. This leads to a monotonically decreasing learning rate, as the current gradient g_θ is multiplied by a fraction with v_θ in the denominator. Especially over many iterations this causes the algorithm to essentially stop learning. The next algorithms that we will explain all attempt to remove or reduce this issue.

Adaptive Delta

Zeiler et al. [55] have developed Adaptive Delta (AdaDelta), which eliminates the need for a learning rate hyper-parameter by replacing it with a moving average of past changes (deltas) in the learned parameters. Therefore this method does not require manual tuning of the learning rate. Another improvement on AdaGrad is that it restricts the window in which past squared gradients are accumulated in an effort to reduce the problem of a monotonically decreasing learning rate that can be an issue with AdaGrad. The authors note that, among other things, AdaDelta is robust against noisy gradient information. This feature could make it very useful in our setting.

Algorithm 5 shows how AdaDelta is slightly more complex than AdaGrad, although the authors themselves note that their algorithm induces only a slightly increased computational overhead compared to default stochastic gradient descent. On line 8 we maintain a fraction β of previous squared gradients, and a fraction $(1-\beta)$ of the current squared gradient. Also note that the learning rate α has been replaced by $\sqrt{\delta_\theta + \varepsilon}$ on line 9. Finally, on line 10, we see how the change in parameter θ is updated for the next iteration. A fraction β of the previous changes in θ is added to a fraction $(1-\beta)$ of the current change. A potential issue with AdaDelta, although not unique, is that there are more variables to hold in memory during

[2] `https://keras.io/api/optimizers/#adagrad`

Algorithm 5 Adaptive Delta

1: β: *interpolates current and past changes in parameters* ▷ Arguments
2: ε: *'fuzz factor'*

3: $\Theta^{(t)}$: *set of parameters to update* ▷ Variables
4: $\delta_\theta^{(t-1)}$: *squared change in parameter* θ *in previous iteration, or 0*
5: $v_\theta^{(t-1)}$: *interpolated squared gradient in previous iteration, or 0*

6: **for each** parameter $\theta^{(t)} \in \Theta^{(t)}$ **do**
7: $g_\theta^{(t)} \leftarrow \frac{\partial J}{\partial \theta^{(t)}}$ ▷ gradient of parameter θ
8: $v_\theta^{(t)} \leftarrow \beta v_\theta^{(t-1)} + (1-\beta)[g_\theta^{(t)}]^2$ ▷ update squared gradient
9: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \frac{\sqrt{\delta_\theta^{(t-1)}+\varepsilon}}{\sqrt{v_\theta^{(t)}+\varepsilon}} g_\theta^{(t)}$ ▷ update parameter
10: $\delta_\theta^{(t)} \leftarrow \beta \delta_\theta^{(t-1)} + (1-\beta)[\theta^{(t+1)} - \theta^{(t)}]^2$ ▷ update with current change in parameter θ

learning. The constants β and ε are usually set to 0.95 and 10^{-6} respectively by default [3], these values are also used by the authors in their experiments.

Adaptive Moment Estimation

Kingma et al. [40] have developed Adam. Like AdaGrad, Adaptive Moment Estimation (Adam) computes adaptive learning rates for each parameter, with the addition of the exponentially decaying moving average of past gradients per parameter. This addition acts as a momentum. For example, if gradients from recent iterations were large, this can affect the current update step to be greater, even though the current gradient may be small. In this manner it is possible to sometimes avoid getting stuck in local minima or taking very long to descend shallow slopes. The authors note that their algorithm works well in settings with noisy data and sparse gradients, which makes it a very good candidate for our setting.

As shown in algorithm 6, Adam is more complex than AdaGrad. There is an additional term m_θ (line 6) that holds the exponential moving average of past gradients. Additionally, the term v_θ (line 7) now holds the exponential moving average of past squared gradients, instead of the cumulative sum. Like AdaDelta, we make use of interpolation on lines 10 & 11. Note that there are separate interpolation constants β_1 and β_2 for the first and second moment terms. There is, however, a bias in the moment estimates towards zero, because they are initialized at zero. Lines 12 & 13 introduce a correction for this bias, where $(\beta)^t$ indicates β to the power t. The corrected moment estimates are then used in the update rule on line 14. We note that the authors propose a slightly more efficient order of computation at the expense of clarity, which is used in our implementation. The default [4] values for the constants proposed by the authors are as follows: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$.

[3] `https://keras.io/api/optimizers/#adadelta`
[4] `https://keras.io/optimizers/#adam`

Algorithm 6 Adaptive Moment Estimation

α: *learning rate* ▷ Arguments
β_1: *interpolates current and previous first moment*
β_2: *interpolates current and previous second moment*
ε: *'fuzz factor'*

$\Theta^{(t)}$: *set of parameters to update* ▷ Variables
$m_\theta^{(t-1)}$: *exponential moving average of past gradients, or 0*
$v_\theta^{(t-1)}$: *exponential moving average of past squared gradients, or 0*

for each parameter $\theta^{(t)} \in \Theta^{(t)}$ **do**
    $g_\theta^{(t)} \leftarrow \frac{\partial J}{\partial \theta^{(t)}}$ ▷ gradient of parameter θ
    $m_\theta^{(t)} \leftarrow \beta_1 m_\theta^{(t-1)} + (1-\beta_1) g_\theta^{(t)}$ ▷ update first moment
    $v_\theta^{(t)} \leftarrow \beta_2 v_\theta^{(t-1)} + (1-\beta_2)[g_\theta^{(t)}]^2$ ▷ update second moment
    $\hat{m}_\theta^{(t)} \leftarrow \frac{m_\theta^{(t)}}{1-(\beta_1)^t}$ ▷ bias correction
    $\hat{v}_\theta^{(t)} \leftarrow \frac{v_\theta^{(t)}}{1-(\beta_2)^t}$ ▷ bias correction
    $\theta^{(t+1)} \leftarrow \theta^{(t)} - \frac{\alpha}{\sqrt{\hat{v}_\theta^{(t)}}+\varepsilon} \hat{m}_\theta^{(t)}$ ▷ update parameter

Adaptive Maximum Squared Gradient

Finally, Reddi et al. [59] have developed the Adaptive Maximum Squared Gradient (AMSGrad) algorithm. They pinpoint that, in some cases, the issue of occasional poor performance of other adaptive methods, such as those listed above, is with the exponential moving average of past squared gradients. It therefore uses the maximum of past squared gradients to update the parameters instead. The rationale being that during optimization only some update steps provide large and informative gradients, and using a decaying average would have quickly diminished their influence. The authors also note that the other algorithms, such as AdaDelta and Adam, do not converge in some settings and elucidate on why this is the case. They show that, in order to have guaranteed convergence, the optimization algorithm must have long term memory of past gradients.

Algorithm 7 Adaptive Maximum Squared Gradient

1: α: *learning rate* ▷ Arguments
2: β_1: *interpolates between previous and current first moment*
3: β_2: *interpolates between previous and current second moment*
4: ε: *'fuzz factor'*

5: $\Theta^{(t)}$: *set of parameters to update* ▷ Variables
6: $m_\theta^{(t-1)}$: *first moment from previous iteration, or 0*
7: $v_\theta^{(t-1)}$: *second moment from previous iteration, or 0*

8: **for each** parameter $\theta^{(t)} \in \Theta^{(t)}$ **do**
9: $g_\theta^{(t)} \leftarrow \frac{\partial J}{\partial \theta^{(t)}}$ ▷ gradient of parameter θ
10: $m_\theta^{(t)} \leftarrow \beta_1 m_\theta^{(t-1)} + (1-\beta_1) g_\theta^{(t)}$ ▷ update first moment
11: $v_\theta^{(t)} \leftarrow \max\left(\beta_2 v_\theta^{(t-1)} + (1-\beta_2)[g_\theta^{(t)}]^2,\ v_\theta^{(t-1)}\right)$ ▷ update second moment
12: $\hat{m}_\theta^{(t)} \leftarrow \frac{m_\theta^{(t)}}{1-(\beta_1)^t}$ ▷ bias correction
13: $\hat{v}_\theta^{(t)} \leftarrow \frac{v_\theta^{(t)}}{1-(\beta_2)^t}$ ▷ bias correction
14: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \frac{\alpha}{\sqrt{\hat{v}_\theta^{(t)}}+\varepsilon}\hat{m}_\theta^{(t)}$ ▷ update parameter

Algorithm 7 shows how AMSGrad is mainly a modification of Adam. The main difference is on line 11, where we ensure that the current $v_\theta^{(t)}$ is always larger than (or equal to) $v_\theta^{(t-1)}$ of the previous iteration. We note that, although the authors only show a biased version in their work, the unbiased version is shown here and also implemented in Disambed. Again, as with Adam, we make use of a slightly more efficient computation where a correction term is calculated once for each iteration in the following way. Fist, we calculate a correction term α_t once for iteration t:

$$\alpha_t = \alpha\sqrt{1-(\beta_2)^t}/(1-(\beta_1)^t)$$

and then replace line 14 with:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha_t m_\theta^{(t)} \ / \ (\sqrt{v_\theta^{(t)}}+\varepsilon)$$

Lines 12 & 13 can then be removed. The default values for the constants are mostly taken from Adam: $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-7}$.

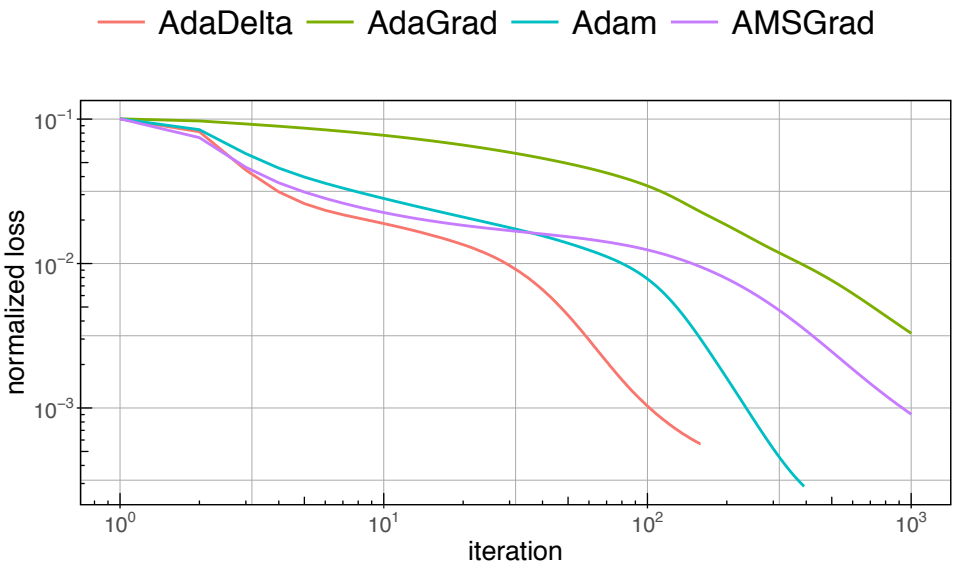
Gradient Descent Performance Comparison

We have performed four experiments to ascertain the relative performances of all gradient descent methods described above. The results of are shown in figures 5.2 and 5.3. The experiments were done on a small data set consisting of records from multiple indices of the Amsterdam City Archives. We have iteratively modified the learning rate α and the fuzz factor ε, while keeping both the maximum change between iterations d_{max} and the maximum number of iterations t_{max} constant. A co-occurrence matrix was computed using the Bookmark Coloring Algorithm specified in section 5.1.1, which was used as input for each algorithm. Each method used the same seed for random initialization and was allowed to run until it either found a local minimum or took 1000 iterations.

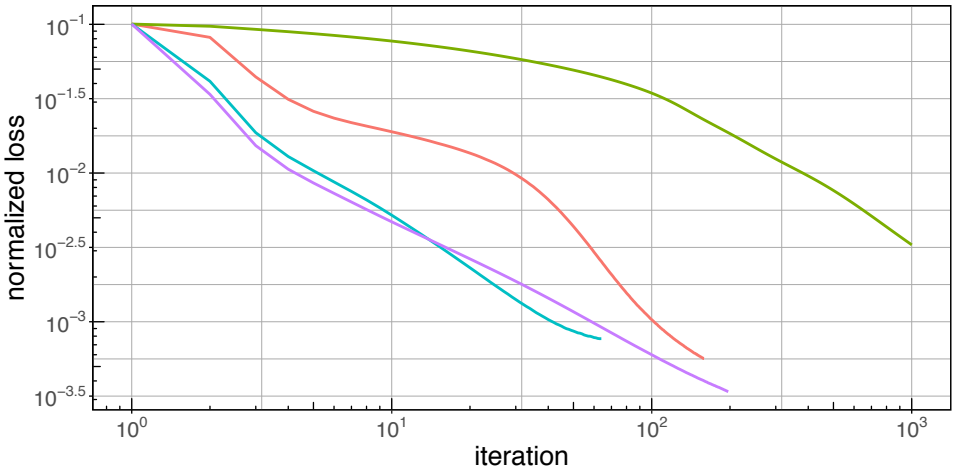
Each gradient descent method has several hyper parameters, such as the learning rate α, one or more interpolators (such as β in AdaDelta) and the 'fuzz factor' ε. While it is in most cases sufficient to keep these at their default values, there are cases where, for instance, setting ε to 0.1 or even 1 instead of 10^{-7} will cause the algorithm to converge instead of diverge. A closer look at the algorithms yields some intuition on why this may be the case. Setting ε to 1 will cause the algorithms (with the exception of AdaDelta) to default back to a global learning rate of α when the per-parameter learning rates are close to 0.

In the first experiment, shown in figure 5.2a, we tested for the performance of the hyperparameter values suggested by the authors of the algorithms and online sources. In the second experiment, figure 5.2b, we used the suggested values for ε while setting α to 0.01 for all four methods. In the third experiment, figure 5.3a, we used $\alpha = 0.01$ as before and also set the value of ε to 0.00001 for all gradient descent methods. Lastly, in the fourth experiment, figure 5.3b, we set the value of ε to 0.1. We note that each axis in the figures is on a $\log_{10}$ scale, which better highlights the difference in performance when the algorithms are near convergence. However, it can hide the speed of convergence somewhat if this is not taken into account.

It is clear that, at least in this case, AMSGrad, AdaDelta and Adam all converge much faster than AdaGrad. In fact, AdaGrad never converged to a local minimum in the allotted 1000 iterations as set by t_{max}. This could mean that the intuition behind these modifications of AdaGrad seems to be correct. That is, the monotonic growth of the cumulative sum of the gradients for each parameter reduces the speed at which it can learn, especially after many iterations. Furthermore, AMSGrad performs very well in most cases, which could indicate that their intuition is also correct in this particular case. Their intuition is that the use of an exponential moving average of all past squared gradients diminishes the influence of update steps that give large gradients. Notably, in the fourth experiment, AdaDelta failed to converge at all, as it could not fall back on a constant learning rate as the other algorithms. Care should be still taken in the choice of gradient descent algorithm. While, for instance, AMSGrad gives good performance, it does this at the expense of additional parameters that need to be held in memory. This is also the case with Adam and AdaDelta. Therefore it can be beneficial to make use of AdaGrad instead when dealing with large, e.g. hundreds of millions of nodes, data sets. This way the memory footprint is minimized while still finding a relatively good optimum, at the expense of a much longer running time.

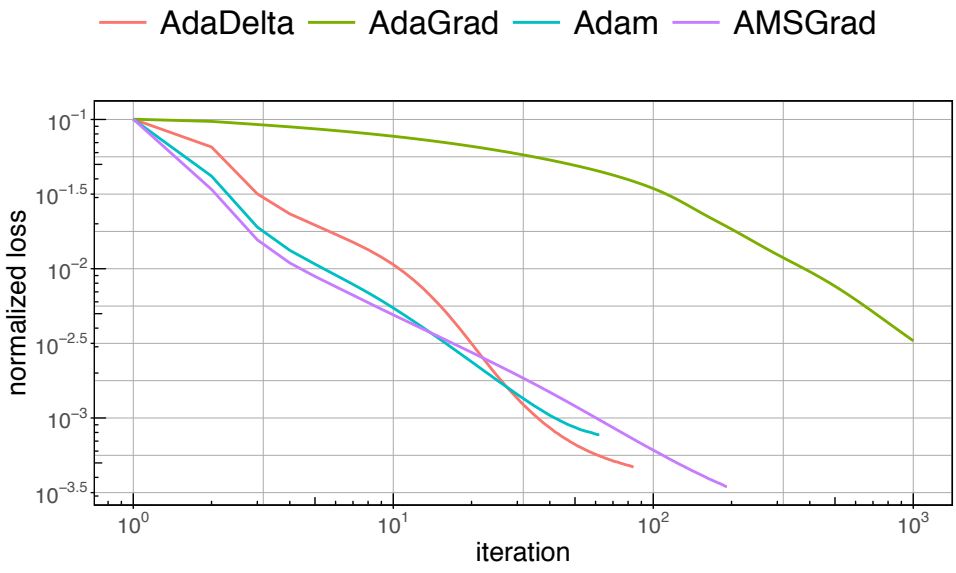


(a) *Respective recommended values for* ε *and* α

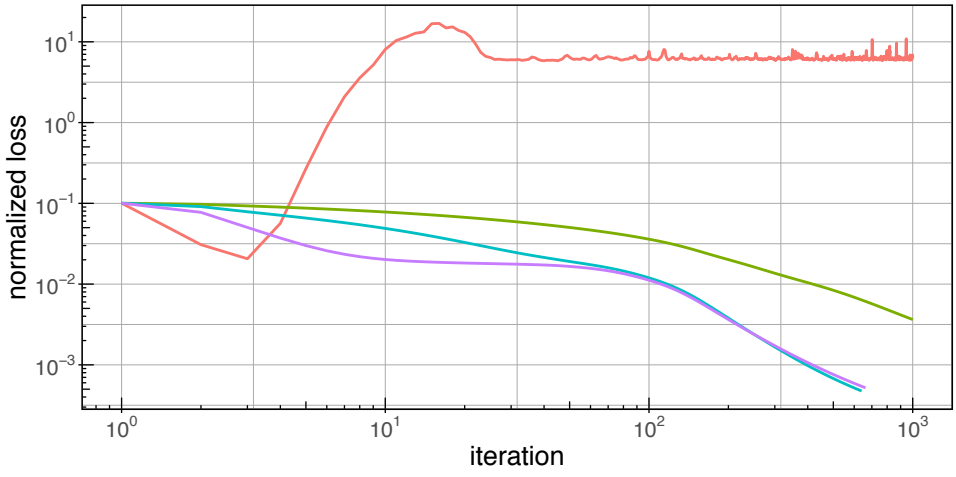


(b) *Respective recommended values for* ε *and* $\alpha = 0.01$

Figure 5.2: *A comparison of gradient descent methods and values for the learning rate* α *and fuzz factor* ε*. The experiment was performed on a small subset of the Amsterdam City Archives indices. The normalized loss is the total summed loss divided by the number of co-occurrences (non-zero values in X). Since using the author recommended settings could lead to unfair comparisons, we also experiment with three different hyperparameter settings in this figure and figure 5.3. Note that both axes are on a* $\log_{10}$ *scale, as to better show the difference in performance when respective local optima are found.*



(a) $\varepsilon = 0.00001$ *and* $\alpha = 0.01$



(b) $\varepsilon = 0.1$ *and* $\alpha = 0.01$

Figure 5.3: *A comparison of gradient descent methods and values for the learning rate* α *and fuzz factor* ε*. The experiment was performed on a small subset of the Amsterdam City Archives indices. The normalized loss is the total summed loss divided by the number of co-occurrences (non-zero values in* X*). AdaDelta is sensitive to larger values of* ε*, while the others still converge. Setting* ε *to such a high value can help reduce numerical instability in cases where many gradients are close to zero. Note that both axes are on a* $\log_{10}$ *scale, as to better show the difference in performance when respective local optima are found.*

5.3 Conclusion

Embeddings are a well known technique for creating numerical representations for entities that are otherwise difficult to process, such as words in text or, as in our case, nodes in a graph. When we assign a numerical vector representation to an entity, we say that we embed that entity. In this chapter we discuss how relevant nodes in an RDF graph (such as all nodes of `type:Person`) are embedded based on their graph neighborhood structure. A useful feature of embeddings is that, when constructed appropriately, the similarity between entities can be calculated (in parallel) using Euclidean distance or cosine similarity, enabling further processing such as clustering entities based on their similarity.

In this chapter we have shown how specific resources (entities) in an RDF graph can be embedded into a d dimensional space. These entities are embedded in such a way that, when they have strongly overlapping neighborhoods in the RDF graph, they have similar embedding vectors (i.e similar coordinates in the embedding space). We have adapted the GloVe algorithm to work more efficiently by computing a non-square co-occurrence matrix and only calculating embedding vectors for the types of resources we are interested in. For example, while each node in the graph can act as context, we only calculate the vectors for nodes of type person.

We have experimented with four different stochastic gradient descent methods in order to find the most time-efficient method that still yields an acceptable minimum cost. In our experiments, the alternative methods mostly outperform AdaGrad, which is used by default in GloVe. Care should still be taken, as the algorithms can be sensitive to the choice of learning rate and fuzz factor. The alternative algorithms are all very competitive. When choosing a good learning rate, AMSGrad (section 5.2.2 performs best, often converging to the lowest local minimum in acceptable time. There is, however, an extra cost associated with not using AdaGrad, which are the necessary hyperparameters one needs to keep track off, such as the squared gradients from previous iterations.

In the next chapter, we will show how the pairwise similarity information gained from the embedding can be used to create clusters of entities. Each cluster then represents a single real life object, such as a person.

6 Clustering with Pairwise Similarities

This chapter is based on the following publications: [2, 5]

- Entity Matching in Digital Humanities Knowledge Graphs, *Computational Humanities Research*, 2021
- Exploiting Transitivity for Entity Matching, *European Semantic Web Conference*, 2021

In the previous chapter we discussed how we create embeddings such that we can calculate pairwise similarities between entities. This method of using these pairwise similarities to classify pairs of entities as being duplicates creates a problem, however, as it may violate transitivity. For example, we may consider the pairs (A,B) and (B,C) as having high similarity, but there is no guarantee that the entities in pair (A,C) are also similar. This creates an issue when we want to create clusters for entities A, B and C. Should we group A and B together and keep C separate? Maybe the information of high similarities of B with both A and C is evidence that the low similarity between A and C was in error.

In this chapter we describe and compare the different clustering algorithms we have used in our entity resolution method. The best performing method will subsequently be used in further experiments and improvements to the general entity resolution method. First, we briefly formalize the entity resolution objective in our clustering setting. Given a starting set of knowledge graphs $\{KG_1,...,KG_n\}$ with $KG_i = (V_i,E_i)$, our objective is to find the subset of pairs of duplicate entities $\mathscr{L}$, where

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

It is generally understood in the field of entity resolution that the quadratic growth of P presents a major problem. Therefore most methods employ some sort of blocking and/or filtering methods [27, 39, 46] to reduce the number of pairs that have to be evaluated. We employ the embedding for this purpose, as detailed in the rest of this chapter.

6.1 Supervised Classification of Duplicate Entities

An evident way of creating clusters is by training classifiers on the vector representations of pairs of entities, where each entity has an associated URI. These classifiers can then be used to predict whether unlabeled pairs of entities are duplicates. All pairs which a classifier judged to be duplicates are added to a linkset. Since a linkset consists of identity relations, this will naturally create clusters. Each classifier is trained on the interaction between vector representations of known duplicate and non-duplicate pairs of entities. In this text, when we are describing the deduplication of entities that refer to persons, we use the terms 'individual' and 'person' interchangeably.

6.1.1 Obtaining Ground Truth

We validate the classifiers on a subset of the Amsterdam City Archives, encompassing 50 years. This subset has been labeled by a symbolic entity alignment method called Lenticular Lenses [37] and subsequently validated by domain experts. The symbolic method explicitly models the ways nodes that represent individuals can be disambiguated, e.g. when one observes a marriage record with a pair of individuals and later a baptism record where the names of both persons are again present. These co-occurrences form clusters and, when they accumulate, are treated as further 'evidence' that these nodes indeed represent a single individual, making the cluster more cohesive. Furthermore, when a cluster has multiple co-occurrences with another cohesive cluster, this counts as additional evidence. For example, both the husband and wife in the previous example can form their own respective cohesive clusters with multiple interconnections between these clusters when entities from both clusters co-occurred in the same record. When enough evidence has been collected and the cluster is cohesive enough, it is considered as a potentially valid cluster. These potential clusters are then further validated by domain experts, either confirming them as valid clusters or splitting the clusters up into multiple clusters when the domain expert judged that there were actually multiple individuals inside a cluster. In our clustering efforts, we attempt to recreate the domain experts' labeling. There are 2,815 labeled entities in total, which were originally grouped into 968 clusters by the Lenticular Lenses method. The domain experts then validated these clusters, which, after splitting up clusters that were invalid, yielded a total of valid 1,073 clusters.

6.1.2 Generating Training and Testing Data

In order to generate a linkset using the vector representations (i.e. embeddings) of nodes in a knowledge graph, we need to be able to decide whether two nodes in the graph are duplicates by using these associated vectors. To do this, we train a classifier to distinguish 'positive' (e.g. both vectors in the pair refer to the same individual) and 'negative' (e.g. they refer to different individual) pairs of vectors. Since each node of interest (e.g. those of type `rdf:Person`) in the knowledge graph is associated with a vector in the embedding, we will refer to nodes as vectors in this section.

First, we consider all possible pairs of vectors inside all potential clusters generated by the symbolic method. That is, as they were presented to the domain experts. For each pair,

we label it positive when the domain expert put both entities in the same cluster, and negative otherwise. This provides us with all possible positive pairs and a set of negative pairs. Note that these negative pairs represent the borderline cases which were incorrectly regarded as positive pairs by the symbolic method. At this point we have many more positive pairs than negative. This skew in the data may affect the learned bias in a classifier and reduce performance. For this reason, to balance the ratio of positive to negative pairs, we randomly sample pairs where the two vectors come from different clusters. These random pairs are assumed to be negative, since the symbolic method has apparently failed to find sufficient evidence that the two vectors refer to the same person. Even if the symbolic method made an error, and two vectors that do refer to the same person ended up in different clusters, it is highly unlikely that this pair will be included in the sample as there are about four million pairs that could be sampled. In total the generated training and testing data consist of 6,166 pairs.

6.1.3 Extracting Features

A classifier requires what are called 'features' in order to learn. Features are numerical representations of the entities on which we attempt to learn. For example, features for a person could be a list of numbers consisting of their age, gender, income, etc. Which feature are necessary depends greatly on the learning objective. In our case we consider two possible methods of creating features from the pairs of vectors we have generated above. First, we use cosine similarity between entities in a pair as the single feature. In this case, we are in essence learning a threshold above which we consider a pair to be duplicates. This will act as a baseline for a more complicated method, and has the advantage that it is fast to compute, allowing the classification of a large number of pairs in a reasonable amount of time. Secondly, for the more complicated method, the pairs are transformed to a new space with the same dimensionality as the original embedding. This new space captures the interaction between the elements in the vectors in a pair. For example, for an embedding with 50 dimensions, two vectors (of length 50) in a pair are transformed to a single vector of length 50. These 50 numbers are then treated as the features of that pair. The interaction is captured by, for each pair of vectors (u, v), calculating the Hadamard product $w = u \circ v$, where $w_i = u_i v_i$. The Hadamard product has often been used in the literature [63] and often comes out as the best or at the top of methods to capture embedding vector interactions.

6.1.4 Experimental Results

We compare the performance of six embeddings, each generated using a different combination of neighborhood traversal strategy (explained in section 5.1.1) and whether or not similar names are connected (explained in section 4.1). When names are not compared, this is called *exact*, otherwise we call it *partial*. Each embedding contains only vectors related to person entities. All models in subsections 6.1.5 and 6.1.6 have been trained and tested on the same 75-25 percent train/test split. The accuracy is recorded as the positive-negative class ratio in the training and test sets is $\frac{1}{2}$. Furthermore, we prefer high precision over recall, as false positives will inhibit linkset performance much more than missing links. Therefore we optimize for the $F_{\frac{1}{2}}$ score by using it as the criterion in the k-fold cross-validation process. We use AdaGrad [22] (as explained in algorithm 4) as our gradient descent algorithm and the

same hyper parameter settings for GloVe in each embedding, namely $\alpha = \frac{3}{4}$, $x_{max} = 1$, and $d = 300$. Table 6.1 shows the performance of each model on each embedding. It is clear that the partial matching schemes including the domain knowledge and allowing for margins of error in the data always performs best, and among these schemes, the bi-directed strategy works best in most cases.

6.1.5 Baseline cosine similarity models

As a baseline, we trained a logistic regression model on solely the cosine similarity between pairs. As can be seen in figure 6.1, the cosine similarity of negative pairs tends to be concentrated around zero, whereas the cosine similarity of positive pairs tends to be positive in most cases, but more spread out over the whole range from -1 to 1. The unrelated randomly sampled negative pairs contribute most to the high probability density around zero cosine similarity. The negative pairs which are difficult to distinguish, namely those previously labeled as belonging to the same cluster before being divided into new clusters by the domain expert, have a higher similarity.

Table 6.1: *Performance statistics of logistic regression, random forests and SVM models. The value for ε depends on the ability of GloVe to converge to a solution without infinities, the smallest value that works is chosen. For each machine learning method, we highlight the overall best results in bold.*

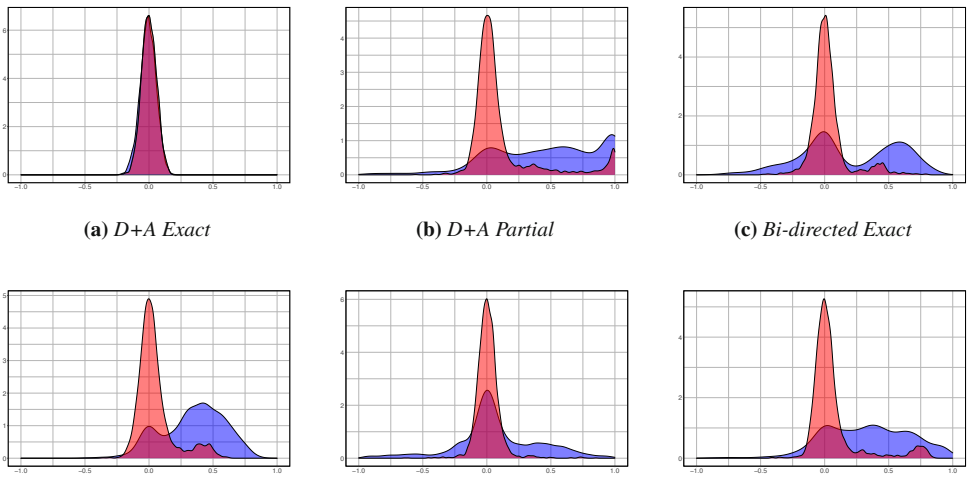
	D+A		Bi-directed		Hybrid	
BCA Settings	exact	partial	exact	partial	exact	partial
α	0.10	0.10	0.10	0.10	0.10	0.10
ε	1e-4	5e-3	5e-3	5e-3	5e-3	1e-2
Cosine Similarity LR	exact	partial	exact	partial	exact	partial
accuracy	0.52	0.78	0.68	**0.82**	0.58	0.77
precision	0.52	0.84	0.79	**0.87**	0.60	0.84
recall	0.52	0.69	0.50	**0.77**	0.45	0.65
$F\frac{1}{2}$ score	0.52	0.80	0.71	**0.85**	0.56	0.80
Hadamard EN LR	exact	partial	exact	partial	exact	partial
accuracy	0.46	0.63	0.64	**0.80**	0.59	0.72
precision	0.46	0.68	0.69	**0.82**	0.64	0.77
recall	0.45	0.48	0.53	**0.75**	0.40	0.62
$F\frac{1}{2}$ score	0.51	0.64	0.64	**0.86**	0.59	0.73
Hadamard Random Forest	exact	partial	exact	partial	exact	partial
accuracy	0.52	**0.84**	0.77	0.82	0.71	0.73
precision	0.53	**0.93**	0.87	0.89	0.74	0.82
recall	0.52	**0.73**	0.62	0.73	0.64	0.60
$F\frac{1}{2}$ score	0.52	**0.88**	0.81	0.85	0.72	0.76
Hadamard SVM	exact	partial	exact	partial	exact	partial
accuracy	0.47	0.75	0.77	**0.85**	0.65	0.73
precision	0.47	0.84	0.82	**0.88**	0.70	0.78
recall	0.47	0.62	0.69	**0.82**	0.55	0.62
$F\frac{1}{2}$ score	0.47	0.78	0.79	**0.87**	0.66	0.74

6.1.6 Hadamard-space based models

Since cosine similarity removes some of the information that is present in the embedding, such as the magnitude of the vectors, we train additional models on the features derived from the Hadamard product of vectors in a pair. Firstly, a logistic regression model with elastic net regularization (EN LR) is trained on each embedding using the *glmnet* package in `R`. The underlying idea is that the regularization will set coefficients for unimportant features, i.e. dimensions, to (near) zero. Secondly, with the same motivation as before, we train a random forest (RF) model for every embedding using the *randomForest* package in `R`. This allows for feature selection as well as a non-linear decision boundary. Thirdly, a radial kernel based support vector machine (SVM) is trained on each embedding. We estimate σ with the *sigest* function provided by the kernlab package, which estimates the range of values for the σ parameter which would return good results when used with a support vector machine. All models have their hyper parameters tuned with 10-fold cross validation and the best model is chosen using the $F\frac{1}{2}$-score. Table 6.2 shows an overview of the possible values for each hyper parameter and which value was chosen in the case of an embedding generated with the bi-directed partial scheme. While this does not allow for easy feature selection, we can achieve a highly non-linear decision boundary.

6.1.7 Performance with small training sets

To build a model which can classify duplicate entities we need a set of labeled pairs to train on (sometimes called a *seed*). This often requires the work of domain experts and is expensive, in both (financial) cost and time. For this reason it is preferred that only a small number of



(a) *D+A Exact* **(b)** *D+A Partial* **(c)** *Bi-directed Exact*

(d) *Bi-directed Partial* **(e)** *Hybrid Exact* **(f)** *Hybrid Partial*

Figure 6.1: *Distributions of cosine similarities (x-axis) of all pairs in the data set generated in section 6.1.2. The red and blue shaded areas represent negative and positive pairs respectively.*

Table 6.2: *Overview of hyper parameter searches per classifier.*

	R package	hyper parameter	possible values
EN LR	glmnet	α	0.1, 0.55, 1
		λ	0.00016, 0.0016, 0.016
RF	randomForest	mtry	2, 151, 300
		ntree	500
SVM	kernlab	C	4, 8, 16
		σ	sigest(X)

pairs need to be labeled while retaining the performance of the classifier. We simulate this situation by repeatedly training the Cosine Similarity logistic regression model on a randomly chosen small fraction of the data. We choose a train/test split with 102 training examples, and with 10,000 random splits for each scheme. Figure 6.2 shows that there is nearly no loss in performance for the bi-directed partial model, when compared to table 6.1.

6.1.8 Issues with this supervised approach

An issue with this approach is that the distributions in figure 6.1 are misleading. This is because the true number of negative pairs is much larger than the number of positive pairs. The blue shaded regions, when scaled appropriately, are much smaller than the red shaded regions. This means that false positive pairs will dominate the true positive pairs. It is therefore necessary to perform a pre-selection on which pairs to choose, such as the candidate pairs explained in section 6.2.1. Secondly, although the classifiers show good performance, even with very little training data, we did not pursue this avenue of research further as this approach inherently requires training data for the classifiers. It is one of the assumptions of the project that a ground truth is either very limited or (more frequently) completely absent. An unsupervised approach is therefore more prudent in our case.

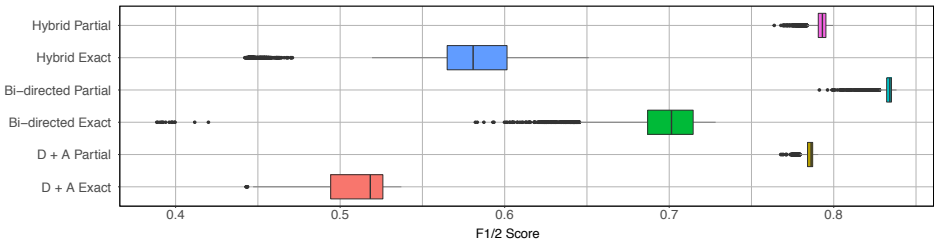


Figure 6.2: *Box-plot of $F\frac{1}{2}$ scores of the Cosine Similarity LR model, for* 10,000 *different random samples of 102 examples.*

6.2 An Unsupervised Approach to Duplicate Detection

An alternative approach to the supervised method described above is to first deal with the skewed distribution issue by creating proto-clusters using (approximate) nearest neighbor search in an embedding. These proto-clusters are then split up using clustering algorithms that can make use of pairwise similarities. It is possible to treat these proto-clusters as connected components in a graph $G = (V,E)$, where each vertex $v \in V$ corresponds to an embedded entity and there exists an edge $(i, j) \in E$ between entities i and j when one entity is the nearest neighbor of the other. Clustering these entities is then analogous to modifying the connected components such that they consist of one or more disjoint cliques. The most principled approach to the problem of modifying graphs into a set of cliques using pairwise weights is called *Correlation Clustering* [8] or *Cluster Editing* [28]. There exists an optimal solution for this problem, however finding the best solution is computationally expensive (NP-hard). We therefore experiment with heuristic algorithms as possible alternatives. These specific heuristic algorithms were chosen because they were either used in related work (sections 6.2.1 and 6.2.1) or claimed to be very good alternatives to the exact solutions (section 6.2.1). An advantage of this general method of clustering is that it does not require the user to specify a parametric form for the clusters, nor the number of clusters. The rest of this chapter is devoted to this new unsupervised approach.

6.2.1 Candidate Pairs

As stated in the entity resolution objective, we wish to determine the subset $\mathscr{L} \subseteq P$ that contains the duplicate pairs. However, the vast majority of pairs in P do not link duplicate entities, so duplicate pairs are very rare [32]. 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 [10, 30] 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 of the 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 u_{ij} for all pairs in P_1, and name all pairs where the similarity exceeds a threshold θ the *candidate pairs*.

When the candidate pairs are treated as edges in an undirected graph, the resulting graph typically consists of a number of connected components. This is illustrated in figure 6.3. 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 $w_{ij} = u_{ij} - \theta$. Note that $w_{ij} > 0$ for all candidate pairs. In general, $w_{ij} > 0$ if $u_{ij} > \theta$ and $w_{ij} \leq 0$ otherwise. We call the resulting

modified component 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.

In algorithm 8 we show how one can efficiently find all candidate pairs, and afterwards create a set of connected components which can then each be clustered separately. First, for each embedded entity i (line 4), we calculate the k approximate nearest neighbors (line 5). As mentioned before, finding the exact nearest neighbors in a high dimensional space such as our embedding can be very time consuming. It is therefore important to find the approximate nearest neighbors. Then, for each neighbor entity j, if the cosine similarity between i and j is equal to, or greater than, θ (line 7), we add entities i and j to the candidate pair set (line 9).

Algorithm 8 Create the Candidate Pair Graph

1: θ: *similarity threshold* ▷ Arguments
2: M: *set of embedded entities*

3: $P \leftarrow \emptyset$: *Set of candidate pairs* ▷ Variables

4: **for each** embedded entity $i \in M$ **do**
5: N_i^k : *set of k (approximate) nearest neighbors of i*
6: **for each** neighbor entity $j \in N_i^k$: **do**
7: **if** $u_{ij} \geq \theta$ **then**
8: $p \leftarrow (\min(i,j),\ \max(i,j))$ ▷ Create candidate pair p
9: $P \leftarrow P \cup \{p\}$ ▷ Add p to set

10: $G = (V,E)$ where $(i,j) \in E$ iff $(i,j) \in P$ and $V = M$ ▷ Define the graph
return G

When dealing with large knowledge graphs, we may have to find connected components from millions of embedded entities. It is therefore also important to efficiently find all the connected components after we have created the candidate pair graph G in algorithm 8. It is most efficient to visit each node $v \in V$ only once, and to be able to quickly find all adjacent nodes to v when necessary. In algorithm 9 we show how connected components are found. First, we build an adjacency list using the candidate pairs stored in G (lines $4-6$). Then, while there remain any nodes that we have not processed, a breadth first search is performed to find a connected component (lines $7-19$). During the search, we hold any adjacent nodes that are found in a first-in-first-out queue. For each node u in the queue, we add it to the component and afterwards add all nodes adjacent to u to the queue. This process is then repeated until the queue is empty. All nodes that are part of this component are marked as processed and not evaluated again (lines 11 & 18).

Figure 6.3 shows an example of how a set of candidate pairs can form a graph with three connected components. Each edge in the graph corresponds to a candidate pair. Note the long chains in the top and bottom components, our intuition here is that the most likely way to cluster this graph is to, for the top component, remove the weakest edge (B-C) of weight 0.72 and add the other missing edge (C-H). Lastly the bottom component can be split up in

Algorithm 9 Finding Connected Components

```
G = (V, E): candidate pair graph                          ▷ Arguments

A: adjacency list                                         ▷ Variables
C: connected components, set of sets

for each pair (i, j) ∈ V do                               ▷ Fill the adjacency list
    A_i ← A_i ∪ {j}                                       ▷ Add j to list of i
    A_j ← A_i ∪ {i}                                       ▷ Add i to list of j

for each unprocessed node v ∈ V do                        ▷ Perform a breadth first search
    Q ← ∅: empty FIFO queue
    c: connected component, a set of nodes
    Q.push(v)                                             ▷ Add seed node to queue
    mark node v as processed
    while Q not empty do
        u ← Q.pop()                                       ▷ Take oldest element in queue
        c ← c ∪ {u}                                       ▷ Add node u to the component
        for each node w ∈ A_u do                          ▷ Search adjacent nodes to u
            if w not processed then
                Q.push(w)
                mark node w as processed

    C ← C ∪ c

return C
```

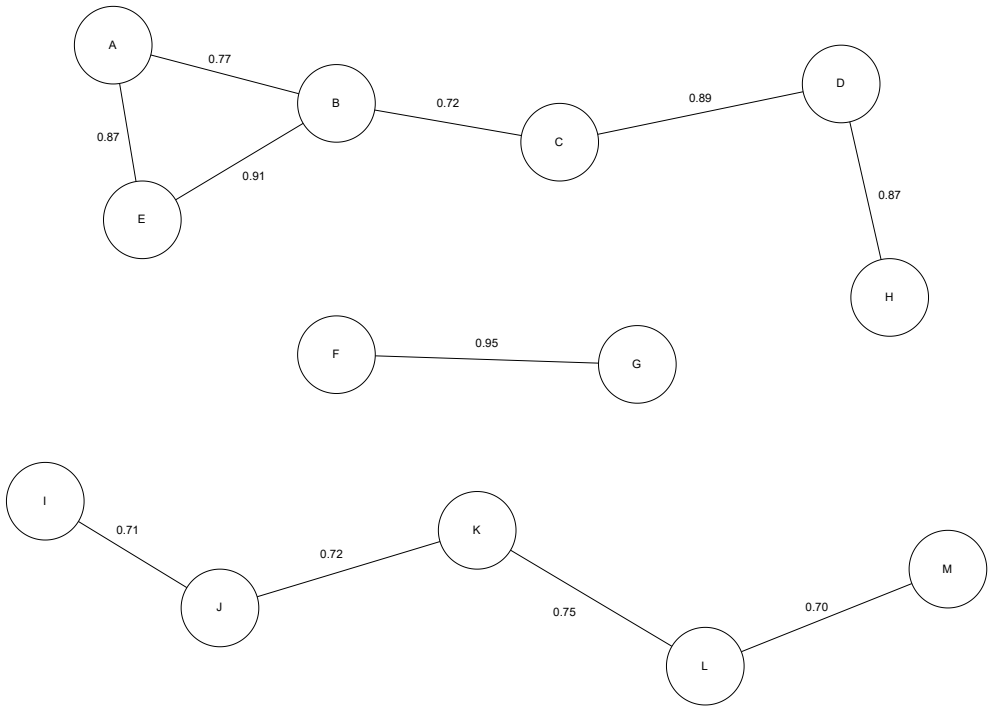


Figure 6.3: *An example of a graph formed by candidate pairs. In this case, the threshold θ was set to 0.7, and therefore all edges in the graph have a weight higher than this value. Note the possible formation of long and weakly connected chains, such as with the bottom component $\{ I, J, K, L, M \}$. Since there are no other edges in this component, they very probably all had weights lower than θ, and/or k was set to some very low value such as 1.*

many ways, depending on the weights of the missing edges. For instance, the best split can be all singleton clusters, or the clusters $\{I, J\}$, $\{K, L\}$ and $\{M\}$, or $\{I\}$, $\{J, K, L\}$, $\{M\}$.

In the next sections we will explain all clustering algorithms that we have used in our experiments, each provided with pseudo code. A clustering algorithm is given a connected component as input and outputs one or more clusters.

Transitive Closure Baseline

Transitive Closure, which is used as the baseline, generates the transitive closure of the relations (edges) between entities involved in a connected component. When the transitive closure is taken of a connected component this will result in a clique on that component, which means it is a single cluster. In other words, each connected component C' is treated as a single cluster, regardless of the values of the 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.

Figure 6.3 shows how, for some components, this method can lead to very low precision and high recall. For instance, the very weakly inter-connected component $\{ I, J, K, L, M \}$ will have many edges added to it to turn it into a clique. These new edges have a very high likelihood of being incorrect.

Regardless, for very high values of θ, since we are only considering candidate pairs of very high confidence, this method can still yield good precision. This is of course at the expense of recall. The main downside is that this method is very sensitive to the choice of θ, which is not always obvious.

Center Clustering

The first clustering algorithm we discuss is called Center Clustering. It was developed by Hassanzadeh et al. [33] and the pseudocode is shown in algorithm 10. Center Clustering works by partitioning the graph such that that each cluster has a center and all nodes in the cluster are similar to that center. An advantage of this clustering algorithm is that it can be performed with only a single iteration through the sorted list of entity pairs. Although a sorting step on all pairs is required first, which necessitates a comparison between these entity pairs. In total, we require $\frac{n(n-1)}{2}$ comparisons for an input graph with n nodes. Afterwards the sorting can be done in $\mathcal{O}(n \log n)$ time, with an additional linear scan to cluster entities.

As can be seen in algorithm 10, Center Clustering first sorts all entity pairs in the input graph in descending order by their weights (lines $6-8$). In our case, the input graph is a connected component. The entities in the sorted list of entities pairs (line 9) are then clustered according to the following rules:

1. If neither node has already been 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 (line 11).

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 (lines 13 & 15).

The Center Clustering algorithm has a tendency to create many small clusters based on strong links, as the pairs with highest similarity are processed first, yielding a high precision but often at the expense of recall. Finally, unclustered nodes become singleton clusters (lines 16 & 17).

Algorithm 10 Center Clustering

θ: *similarity threshold* ▷ Arguments
N: *nodes in component*
u: *cosine similarities between embeddings of pairs of nodes* (i, j); $i, j \in N$

$C \leftarrow \emptyset$: *clusters assignments, set of sets* ▷ Variables
w: *weights between all pairs of nodes* (i, j); $i, j \in N$

for each pair of nodes (i, j); $i, j \in N$ **do**
    $w_{ij} \leftarrow u_{ij} - \theta$ ▷ Calculate weights
$w \leftarrow \text{sort}(w)$ ▷ Sort large to small

for each weight $w_{ij} \in w$ **do**
    if i *not clustered* **and** j *not clustered* **then**
        $C \leftarrow C \cup \{\{i, j\}\}$ ▷ add nodes to a new cluster with i as center
    else if i *is center* **and** j *not clustered* **then**
        $C_{(i)} \leftarrow C_{(i)} \cup \{j\}$ ▷ add node j to cluster of node i
    else if j *is center* **and** i *not clustered* **then**
        $C_{(j)} \leftarrow C_{(j)} \cup \{i\}$ ▷ add node i to cluster of node j

for each non-clustered node $n \in N$ **do**
    $C \leftarrow C \cup \{\{n\}\}$ ▷ add node as singleton cluster

return C

Merge-Center Clustering

The second clustering algorithm we discuss is called Merge-Center Clustering. This algorithm was also developed by Hassanzadeh et al. [33]. As can be seen in algorithm 11, Merge-Center Clustering adds another step to the Center Clustering algorithm where if one of the nodes is a center node, and the other node has already been assigned to a different cluster, then the two clusters are merged (lines $12-17$). These steps tend to result in fewer and larger clusters than Center Clustering. Merge-Center Clustering has identical properties as Center Clustering regarding the time complexity.

One issue with this algorithm is that, at low threshold values of θ, it tends to create very large clusters. This behaviour is similar to that of the transitive closure method and leads to very low precision below a certain value θ, as can be seen in the experimental results in figures 6.4 and 6.5.

Algorithm 11 Merge-Center Clustering

1: θ: *similarity threshold* ▷ Arguments
2: N: *nodes in component*
3: u: *cosine similarities between embeddings of pairs nodes* (i, j); $i, j \in N$

4: $C \leftarrow \emptyset$: *clusters assignments, set of sets* ▷ Variables
5: w: *weights between all pairs of nodes* (i, j); $i, j \in N$

6: **for each** unordered pair of nodes (i, j); $i, j \in N$ **do**
7: $\quad w_{ij} \leftarrow u_{ij} - \theta$ ▷ Calculate weights
8: $w \leftarrow \text{sort}(w)$ ▷ Sort large to small

9: **for each** weight $w_{ij} \in w$ **do**
10: $\quad$ **if** *i not clustered* **and** *j not clustered* **then**
11: $\quad\quad C \leftarrow C \cup \{\{i, j\}\}$ ▷ add nodes to a new cluster with i as center
12: $\quad$ **else if** *i is center* **and** *j clustered* **and** *i, j not in same cluster* **then**
13: $\quad\quad C_{(i)} \leftarrow C_{(i)} \cup C_{(j)}$ ▷ merge clusters $C_{(i)}$ and $C_{(j)}$
14: $\quad\quad C \leftarrow C \setminus C_{(j)}$
15: $\quad$ **else if** *j is center* **and** *i clustered* **and** *i, j not in same cluster* **then**
16: $\quad\quad C_{(j)} \leftarrow C_{(j)} \cup C_{(i)}$ ▷ merge clusters $C_{(j)}$ and $C_{(i)}$
17: $\quad\quad C \leftarrow C \setminus C_{(i)}$
18: $\quad$ **else if** *i is center* **and** *j not clustered* **then**
19: $\quad\quad C_{(i)} \leftarrow C_{(i)} \cup \{j\}$ ▷ add node j to cluster of node i
20: $\quad$ **else if** *j is center* **and** *i not clustered* **then**
21: $\quad\quad C_{(j)} \leftarrow C_{(j)} \cup \{i\}$ ▷ add node i to cluster of node j

22: **for each** non-clustered node $n \in N$ **do**
23: $\quad C \leftarrow C \cup \{\{n\}\}$ ▷ add node as singleton cluster

return C

Vote Clustering

Vote Clustering (algorithm 12), proposed by Elsner et al. [26] , is given as an efficient heuristic alternative to the NP-hard Correlation Clustering problem. They show that simple algorithms, such as Vote, are already close to optimality. From their experiments, the Vote Clustering algorithm came out as best, which is why we have used it in our experiments. The Vote algorithm processes the nodes in C' in arbitrary order. In their experiments the authors ran 100 different permutations and reported the run with the best objective value. They report slightly better results than the average run. We, however, only perform a single run, as computing many runs defeats the purpose of using a faster heuristic alternative. From our experimental results this seems sufficient for adequate performance.

We shall briefly go trough the pseudocode shown in algorithm 12. First, each node i is assigned to the cluster with the largest positive sum of weights with regard to i (lines $6-11$). If there are no clusters yet, or no cluster has a positive sum, then a new cluster is created for i (lines 4 & 12). For high values of $\theta \approx 1$, this will cause most sums to be negative, thereby creating more clusters.

The algorithm is relatively efficient as each non clustered node is only compared to other already clustered nodes and each node, once considered, is always assigned a cluster. Thus, for n nodes, we do a total of $\frac{n(n-1)}{2}$ entity pair comparisons. The sorting step of Center and Merge-Center clustering is omitted. Vote clustering has been suggested as a heuristic alternative to the exact correlation clustering, explained next.

Algorithm 12 Vote Clustering

1: θ: *similarity threshold* ▷ Arguments
2: N: *nodes in component*
3: u: *cosine similarities between embeddings of nodes* $\in N$

4: $C \leftarrow \emptyset$: *clusters assignments, set of sets* ▷ Variables

5: **for each** non-clustered node $i \in N$ **do**

6: **for each** cluster $C_k \in C$ **do**
7: $s_k \leftarrow \sum_{j \in C_k} u_{ij} - \theta$ ▷ calculate sum for cluster C_k

8: $m \leftarrow \text{argmax}_k(s_k)$ ▷ index of cluster with largest sum

9: **if** $s_m > 0$ **then**
10: $C_m \leftarrow C_m \cup \{i\}$ ▷ assign i to cluster with highest weight
11: **else**
12: $C \leftarrow C \cup \{\{i\}\}$ ▷ there was no good cluster, make new

return C

Correlation Clustering

Correlation Clustering [8] partitions the nodes of $C' = (V, E)$ into a number of disjoint subsets (clusters) such that the sum of the between-cluster positive weights minus the sum of the within-cluster negative weights is minimized. More formally, let $\Gamma = \{V_1, V_2, \ldots\}$ 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 intra(Γ) denote the collection of edges with both endpoints in the same partition (cluster), and inter(Γ) the collection of edges with both endpoints in a different partition (cluster). The objective is to find the clustering Γ that minimizes the cost:

$$\text{cost}(C', \Gamma) = \sum_{(i,j) \in \text{inter}(\Gamma) \cap E^+} w_{ij} - \sum_{(i,j) \in \text{intra}(\Gamma) \cap E^-} w_{ij} \tag{6.1}$$

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 results in fewer but larger clusters. Due to the NP-hardness of the correlation clustering problem, we only perform correlation clustering on components no larger than 50 nodes. Larger components are processed with the vote algorithm.

Integer Linear Programming Setup

While the above cost function (6.1) gives a good intuitive understanding on the Correlation Clustering problem, it can be difficult to see how a solution can be found computationally. We therefore also present a different formulation of Correlation Clustering problem, called Cluster Editing [28]. In fact, since we have calculated weights for each edge in a connected component, our particular case is the *weighted* Cluster Editing problem, presented in equation 6.2.

$$\underset{\boldsymbol{X}}{\text{argmin}} \sum_{(i,j) \in \Lambda} w_{ij} - \sum_{1 \le i < j \le n} w_{ij}\, x_{ij} \tag{6.2}$$

$$\begin{aligned}
\text{subject to} \quad & +x_{ij} + x_{jk} - x_{ik} \le 1 && \text{for all } 1 \le i < j < k \le n, && (6.3)\\
& +x_{ij} - x_{jk} + x_{ik} \le 1 && \text{for all } 1 \le i < j < k \le n, && (6.4)\\
& -x_{ij} + x_{jk} + x_{ik} \le 1 && \text{for all } 1 \le i < j < k \le n, && (6.5)\\
& x_{ij} \in \{0, 1\} && \text{for all } 1 \le i < j \le n && (6.6)
\end{aligned}$$

Here w_{ij} is the weight given to the pair (i, j), Λ denotes the set of edges in a component with a positive weight, and $x_{ij} = 1$ when a link between i and j is part of the solution, and zero otherwise. $\boldsymbol{X}$ denotes the strictly upper triangular $n \times n$ matrix containing all x_{ij} values, where n is the total number of entities in the connected component. The $3\binom{n}{3}$ constraints (6.3) - (6.5) together ensure that the solution satisfies transitivity. The objective function (6.2) can be interpreted as minimizing the difference between the starting situation containing only edges with positive weight (first sum), and a possible solution (second sum).

Since the first sum is a constant and does not depend on the decision variables x_{ij}, the objective function can be simplified with the following steps:

First, we write the first sum as some constant c:

$$\underset{\boldsymbol{X}}{\operatorname{argmin}}\ c\ -\sum_{1\leq i<j\leq n} w_{ij}\, x_{ij}$$

Then, since the solution does not depend on c, we set $c = 0$:

$$\underset{\boldsymbol{X}}{\operatorname{argmin}}\ 0\ -\sum_{1\leq i<j\leq n} w_{ij}\, x_{ij}\ =\ \underset{\boldsymbol{X}}{\operatorname{argmin}}\ -\sum_{1\leq i<j\leq n} w_{ij}\, x_{ij}$$

Now we can simply flip the sign and, instead of minimizing, maximize with the following equation:

$$\underset{\boldsymbol{X}}{\operatorname{argmax}} \sum_{1\leq i<j\leq n} w_{ij} x_{ij} \tag{6.7}$$

Where the simplified equation 6.7 is subject to the same constraints as equation 6.2.

Now that we have an equation (and constraints) that can be solved by off-the-shelf (integer) linear programming software (we have used ojAlgo [50]), extracting the final clusters can be done by finding the connected components in the result with the use of algorithm 9. We note that a faster algorithm can be obtained with a linear programming relaxation and cutting plane approach [11]. We did not implement this due to time constraints and leave this to future work.

6.3 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 [37]. In this section, we first explain the used datasets, i.e. SAA and DBLP. Without getting into the technical details of the application of our methodology in this experimental setup, we only note that the nodes of these knowledge graphs are embedded into a separate 50-dimensional space.

6.3.1 City Archives of Amsterdam

We shall briefly explain again the 2019 version of the structure of the City Archives of Amsterdam[1] (in Dutch *Stadsarchief Amsterdam*, abbreviated SAA) that was used in these experiments. The SAA is a collection of registers, acts and books from the 16^{th} century up to modern times. The original data are in the form of handwritten books that have been scanned and

[1] `https://archief.amsterdam/indexen`

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, Notice of 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 notice of 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 Ecartico [2] data set, which is a comprehensive collection of biographical data from more well-known people from the Dutch golden age. Back in chapter 3, figure 3.2 we have shown 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.

6.3.2 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 release[3] of DBLP,[4] 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 76,397 new URIs are 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 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.

[2] http://www.vondel.humanities.uva.nl/ecartico
[3] http://data.linkeddatafragments.org/dblp
[4] https://dblp.uni-trier.de

Table 6.3: *The maximum F-$\frac{1}{2}$ and F-1 Scores and associated similarity threshold θ for the SAA KGs and DBLP KG*

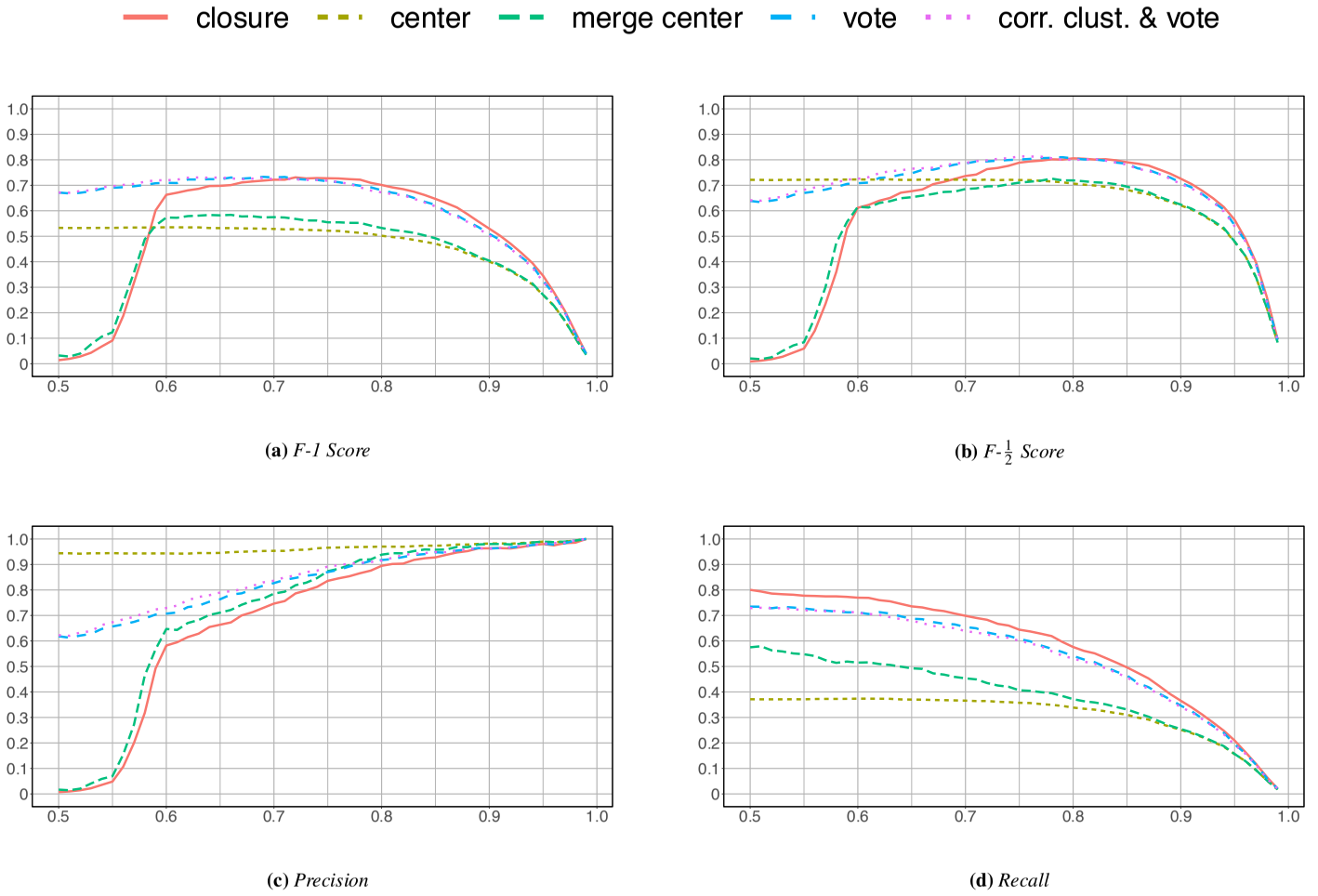
	SAA				DBLP			
	F-$\frac{1}{2}$ score		F-1 score		F-$\frac{1}{2}$ score		F-1 score	
	θ	max	θ	max	θ	max	θ	max
closure	0.80	0.81	0.72	0.73	0.83	0.82	0.79	0.72
center	0.63	0.72	0.60	0.54	0.74	0.61	0.65	0.43
merge-center	0.78	0.73	0.66	0.58	0.80	0.68	0.76	0.53
vote	0.79	0.81	0.69	0.73	0.80	0.82	0.72	0.73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

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.

6.4 Experimental Results

Table 6.3 together with Figures 6.4 and 6.5 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 6.3 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 6.4 presents the overall scores of the clustering methods for various similarity thresholds applied to SAA, while figure 6.5 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 6.4c, 6.4d, 6.5c and 6.5d. Note that both center and merge-center have a much lower maximum F-score than the other methods. From Figures 6.4b and 6.5b 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.



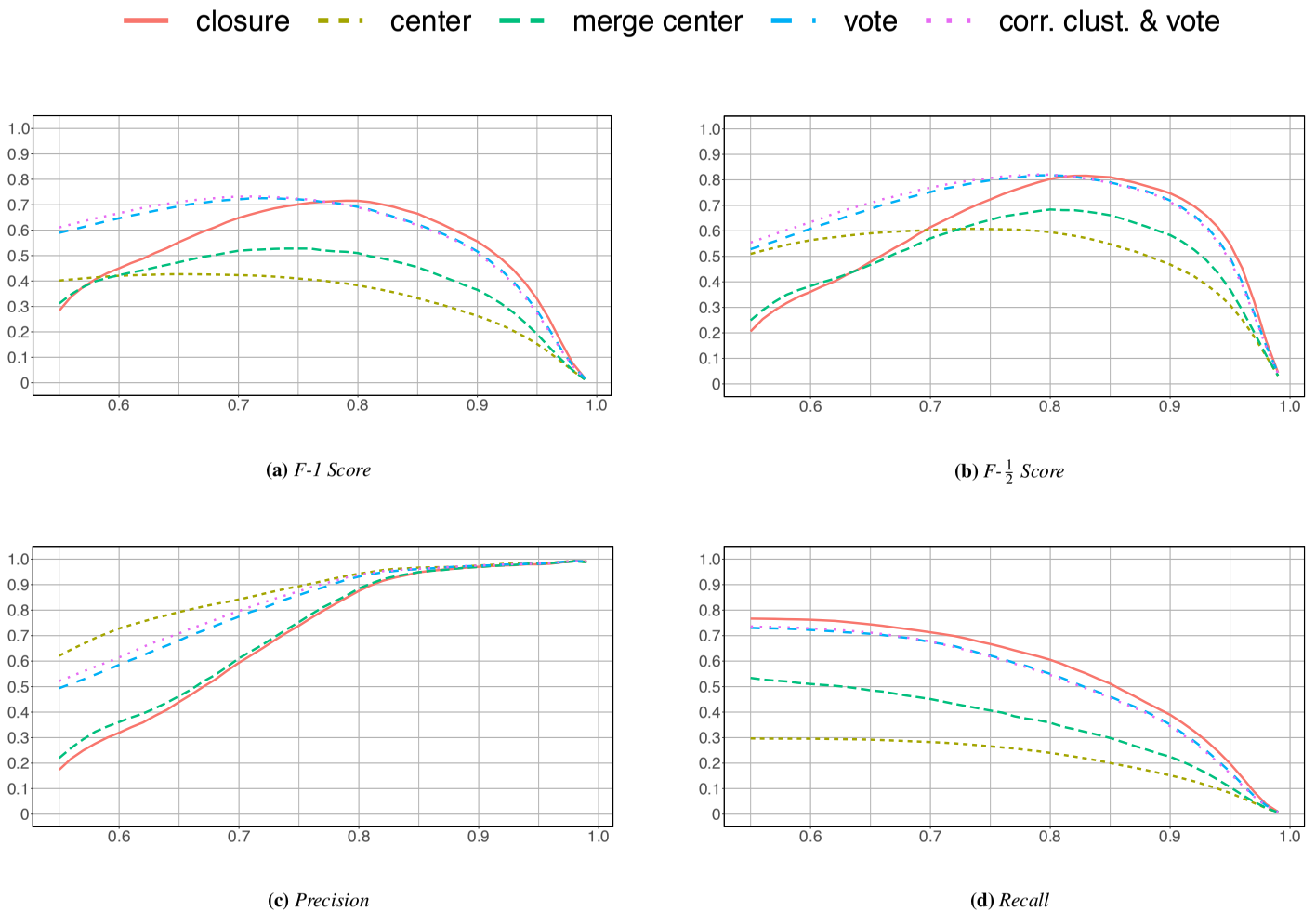
(a) *F-1 Score*

(b) *F-$\frac{1}{2}$ Score*

(c) *Precision*

(d) *Recall*

Figure 6.4: *Results for a range of similarity thresholds θ (horizontal axis) for the SAA KGs.*



(a) *F-1 Score*

(b) *F-$\frac{1}{2}$ Score*

(c) *Precision*

(d) *Recall*

Figure 6.5: *Results for a range of similarity thresholds θ (horizontal axis) for the DBLP KG.*

6.5 Conclusion

In this chapter, 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. Pairwise similarity information, such as the cosine similarity from an embedding, can be used to group entities into connected components, which act as proto-clusters. These components are then further spit up into cliques, where each clique acts as a cluster of entities which we regard as all referring to the same real life object (such as a person).

Both supervised and unsupervised methods yield acceptable results. The main issue with the supervised approach is that a ground truth is necessary. It is one of the assumptions of this project that this is either not the case, or very expensive to obtain. A domain expert would have to manually label example pairs of entities, where immediately the additional problem arises of which pairs to label. This can be somewhat mitigated by using data from another method. However, the main problem of requiring labeled pairs persists.

The unsupervised method uses no information other than a user-set threshold θ and an embedding. 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 next chapter we will experiment with introducing additional symbolic features, such as logical rules about how certain properties relate, to create a system that combines both symbolic and sub-symbolic information.

7 Incorporating Domain Knowledge

This chapter is based on the following publications: [6, 7]

- Combining Node Embeddings with Domain Knowledge for Identity Resolution, *Graphs and Networks in the Humanities*, 2021
- Adding Domain Knowledge to Improve Entity Resolution in 17th and 18th Century Amsterdam Archival Records, *SEMANTiCS*, 2022

In the previous chapter we described how clustering techniques can be used to perform entity resolution on embeddings. Nevertheless, these models may produce errors in the entity resolution outcome. That is, a type 1 error can occur (false positive), where pairs of entities may be identified as duplicate while in reality they are not. Alternatively a type 2 error may happen (false negative), where two duplicate entities are assigned to different clusters. These errors may come about due to inherent weaknesses in the embedding technique, or incomplete and unreliable information in the original data. Due to the lack of ground truth, a very common occurrence in the field of Digital Humanities, it is not viable to switch to a supervised method in an attempt to improve performance. Therefore, it is necessary to use other types of information, such as rational axioms and domain knowledge, to improve the entity resolution method. In this chapter we focus on resolving type 1 errors.

We argue that domain-specific knowledge can be used to detect and correct matching errors, and propose an approach to incorporate domain knowledge in an existing entity resolution algorithm. Examples of such domain-specific knowledge are

1. Two entities cannot be identical if they occur in the same civil registration record. This captures many different semantics, depending on the record type. For example, this captures the idea that one cannot marry oneself, one cannot report oneself as deceased, and one cannot be a witness to an event where one is the principal participant.

2. Two entities, one with birth date x, the other with marriage date y cannot be identical if $x > y$. These kinds of rules are dependent on the record type, as records of other types do not contain the necessary information. For example, a rule that uses semantics on marriage and birth needs a marriage record to get a marriage date and another record

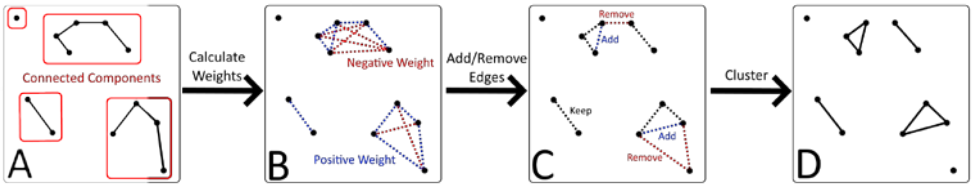


Figure 7.1: *A quick overview of the entity resolution process*

with birth date information, or, for example, a baptism record to approximate the birth date.

These rules were created with the help of domain- and data experts. The purpose of these rules is to test the conclusion of a sub-symbolic method that two entities are the same. Furthermore, a complication to the use of such rules is that domain-specific knowledge can involve uncertainty due to the ambiguity in the data. For example, if we want to exploit the fact that a person cannot be born after they have died, one must be able to identify the born and dead persons unambiguously. The difficulty is that in this type of archival sources it is not a given that the person is actively involved in the registration event when they are mentioned. It could even be that the person is already deceased and is solely used as a disambiguating description for someone else (e.g. *Claartje Jans, widow of Cornelis Pieters*). If this happens in a burial registration or the registration of a testament, we need a rule that is aware of this knowledge and we do not treat the fact of a person's death as 100% certain, but see this as relative to the number of persons involved in the event. This uncertainty then propagates to the conclusion of the rule.

In this chapter we present 1. An experiment with the incorporation of these rules with an existing embedding. 2. A case study where we apply the method to a more challenging setting with more heterogeneous and voluminous data. We also make a distinction between the terms 'experiment' and 'case study' to indicate that we have no sufficient ground truth available for the latter. The main motivation for the case study was to apply the method to our data to make a linkset, which is then incorporated in the Golden Agents infrastructure. The purpose of the experiment is to test whether the inclusion of domain knowledge improves entity resolution performance and to provide estimates of starting parameters that lead to an optimal configuration of the method. This configuration is then used in the case study.

As a quick reminder we briefly describe the entity resolution method. We start with a set of knowledge graphs that are combined into a single graph by comparing literals on their values and semantic context. The resources we wish to disambiguate in the merged graph, such as all nodes of type `rdf:Person`, are then embedded into a d dimensional space using a modification of the GloVe algorithm. That is, these relevant entities are represented as an embedding $\mathscr{E} = \{v_1, \ldots, v_m\}$ where each vector $v \in \mathscr{E}$ corresponds to a (non-unique) entity (e.g. a person resource in our data set) which may have duplicates (i.e. other resources that refer to the same real-life object). Then, we construct a new graph $G = (V, E)$ where each vertex in V corresponds to a vector in the embedding $\mathscr{E}$. We then take the k (approximate) nearest neighbours (based on euclidean distance between the embedding vectors) of each en-

tity $i \in V$, denoted by the set N_i^k, and create an edge between that entity and its neighbour $j \in N_i^k$ if their cosine similarity $u_{ij} = cosim(v_i, v_j)$ exceeds some threshold θ. Such constructed graphs, which are illustrated in figure 6.3 in chapter 6 and here on panel A in figure 7.1, consist of a number of connected components. The number and size of these components depend on the choice of θ: values of θ close to 1 result in many small components and lower values result in fewer but larger components. With each possible pair of entities i and j within each component we associate a weight $w_{ij} = u_{ij} - \theta$. Panel B illustrates how these weights can be both positive (blue, similar) and negative (red, dissimilar). Components are subsequently subdivided into disjoint cliques using either integer linear programming (ILP) or an alternative heuristic clustering algorithm (both are illustrated with panel C). Each algorithm works by associating a cost for omitting a pair with a positive weight, i.e. assign to different clusters, and for retaining a pair with a negative weight, i.e. assign to same cluster, in a solution. The partitioning with the (approximate) lowest cost is then selected, illustrated in panel D. This partitioning also represents the final output of the disambiguation method. This output can take the form of a list of clusters, which is useful for domain experts to validate, or a set of `sameAs` relations between entity pairs. This set of `sameAs` relations, often called a linkset, can then be used by SPARQL query engines to reason about which properties belong to which resources.

7.1 Representing Rules

In the remainder of this chapter, we explain our extension to the work described above. That is, how domain knowledge is encoded and integrated into this process at two points. This domain knowledge is encoded as a set of rules provided by domain experts. These rules, which aim at identifying non-duplicate entities, have the general form: *if a certain condition holds for a pair of entities i and j, then it is ruled out that i and j are duplicates*. We assume that we have a data set that contains additional knowledge about the entities on which the condition of these rules can be checked. This data can, for instance, come in the form of an RDF graph, where rules are encoded as filter clauses in SPARQL queries. We have provided an example SPARQL query in our repository[1]. Such rules can be divided into two categories:

1. **Definite Rules:** There are rules for which we know in advance that applying them results in conclusions with high certainty, that is, the two entities on which the definite rule is applied are not duplicates. For example, *if entity i and j both occur in the same marriage event, then it is ruled out that entities i and j are duplicates* as one can not be a bride/groom and witness at the same time.

2. **Probabilistic Rules:** For other rules, we may not be as confident in the evaluation of the premise. This can, for instance, be due to uncertainty or ambiguity in the data. For example, consider the rule *if the burial date of entity i is before the marriage date of entity j, then it is ruled out that entities i and j are duplicates*. The burial record is ambiguous as it contains in addition to entity i also an entity k, without mentioning who was buried. This means that, in the absence of any other evidence, the probability that entity i was the one who died is 50%, and the probability of the conclusion of the rule that entities i and j are ruled out as duplicates also becomes 50%.

[1] `https://github.com/knaw-huc/golden-agents-occasional-poetry`

We use $\mathscr{S}_{ij}$ to denote the set of all definite rules applied to i and j, meaning i and j satisfy the premise. Furthermore, $\mathscr{R}_{ij} = \{r_1, \ldots, r_n\}$ denotes the set of all probabilistic rules that can be applied to entities i and j. First, the definite rules are used to further cull the approximate nearest neighbours found in the embedding, next to already removing neighbours that have a similarity which falls below the threshold θ. That is, we only create an edge between entities i and j in the graph G if $u_{ij} \geq \theta$ and no definite rule is satisfied for that pair. Since multiple probabilistic rules could be satisfied on an entity pair, the rules and their probabilities need to be aggregated first. To this end, for $r \in \mathscr{R}_{ij}$, we use $p(r)$ to denote the probability that i and j are ruled out as duplicates, e.g. $p(r) = 0.5$ is read as in 50% of the cases it is ruled out that i and j are duplicates. When evaluating a set of rules on a pair of entities i and j, the outcomes are aggregated as follows:

$$p_{ij} = 1 - \left(\prod_{r \in \mathscr{R}_{ij}} 1 - p(r) \right), \tag{7.1}$$

where p_{ij} is the total probability that it is ruled out that i and j are duplicates.

Here we assume for convenience that rules are independent, as it would be very difficult to quantify the dependencies between all of them. As is, for example, the case with the naive Bayes classifier, we expect that the independence assumption still gives a reasonable approximation.

Using equation (7.1), we calculate a penalty s_{ij} for considering a given pair of entities i and j as duplicates:

$$s_{ij} = \begin{cases} 10^6 & \text{if } \mathscr{S}_{ij} \neq \emptyset \\ 0 & \text{else if } \mathscr{R}_{ij} = \emptyset \\ 1 - \sqrt{1 - p_{ij}} & \text{otherwise} \end{cases} \tag{7.2}$$

This yields a very large penalty if any definite rule was applied for the pair of entities i and j and no penalty if no rule was applied. This large penalty guarantees that pairs of entities satisfy a definite rule in the final results. In all other cases, it transitions smoothly between $s_{ij} = 1$ for $p_{ij} = 0$, and $s_{ij} = 0$ for $p_{ij} = 1$. The square root is taken to reduce the penalty to prevent the probabilistic rules from acting as definite rules in some edge cases.

Finally, we calculate the updated final weight w_{ij}, used for the partitioning of connected components, with the following equation:

$$w_{ij} = u_{ij} - s_{ij} - \theta \tag{7.3}$$

Note that s_{ij} represents the amount of evidence against the conclusion that i and j are duplicates. Also, note that rule application can never lead to an increase in the weight since a lack of evidence to rule out a pair of entities as duplicates does not constitute evidence that they are the same. At this point, the connected components are then partitioned into cliques as we describe in chapter 6. Each clique then corresponds to a unique real-life object (panel D). For example, in our experiment, each vertex (entity) denotes a *reference* to a person, and each clique corresponds to a single real-life person (object).

7.2 Experimental Setup

For both the experiment and the case study, we use domain knowledge to improve the results. The experiment shows by how much the rules improve performance in both precision and recall, while the case study only gives us precision. The experiment makes it possible to determine reasonable values of θ for the case study when both precision and recall are considered.

7.2.1 Domain Knowledge Experiment

For our experiment to determine the effectiveness of applying rules we have used four data sets containing real historical data from the cultural heritage domain. We made use of a subset from the 2021 version of the Amsterdam City Archives and a subset of the ECARTICO data set containing information on marriage registrations. Combined, the above-described subsets form a data set that contains 12,517 entities referring to (non-unique) persons, and a partial ground truth of 1073 clusters, obtained with manual validation by domain experts [37].

We have designed two definite rules based on expert domain knowledge, namely that two entities co-occurring in the same record are not duplicates, and that two entities originating from ECARTICO are not duplicates. The latter rule is based on the knowledge that ECARTICO is an expert-curated biographical data set that contains a single unique entry for a single person. Furthermore, we have constructed 8 probabilistic rules that consider the dates at which the events took place, and how many years occur between them. Below we list three example rules for entities i and j:

1. Entity i is mentioned in the role of groom or bride in a notice of marriage registration at date d_1, and j is mentioned in the role of husband, wife, father, or mother in a marriage or baptism registration respectively at date d_2. If more than 30 years occur between d_1 and d_2, then it is ruled out that i and j are duplicates.

2. Entity i is mentioned in the role of groom or bride at date d_1. If there is less than 17 years between the birth date of j and d_1, then it is ruled out that i and j are duplicates.

3. Entity i is mentioned in the role of groom or bride at date d_1. If there is more than 60 years between the birth date of j and d_1, then it is ruled out that i and j are duplicates.

Note that in many cases, a rule does not apply because one or both attributes are missing for a given pair of persons. All rules are combinations of what the dates represent and how they relate to each other. The rules that contain a burial date in the premise have been given a probability of 0.5 for reasons explained in section 3.1.3. All other rules have been assigned a range of probabilities ranging from 0.25 to 0.95, with the justification that, for cultural reasons, these kinds of rules can be trusted with different levels to be correct in their assessment. We test our method four times, each time with a different set of rules: (a) no rules, (b) only definite rules, (c) only probabilistic rules, and (d) all rules.

We apply a combination of the Correlation Clustering [8] and Vote [26] algorithms, as these algorithms performed best in our previous work [2]. Due to the NP-hardness of the Correlation Clustering (ILP) algorithm, we apply it only to connected components smaller

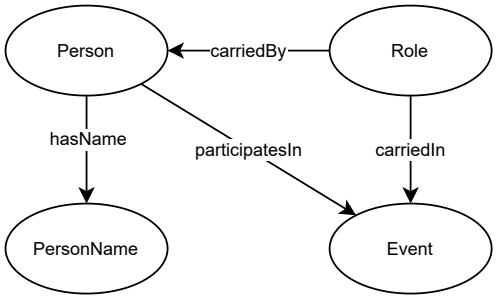


Figure 7.2: *Basic structure of our RDF-model. Persons participate (either actively or passively) in an event. In the event, they carry a specific role: the role in which they are mentioned. Every person has one or more names.*

than 100 vertices. Larger components are handled with the Vote algorithm. This is mostly an issue when using low values of θ, where large components can be generated. As θ increases, Correlation Clustering is used more, until it is solely used when $\theta \geq 0.70$. We have shown in previous work [5] that in our setting the Vote algorithm is very competitive with Correlation Clustering. These clusters are then compared to a partial ground truth which has been validated by domain experts.

7.2.2 Occasional Poetry Case Study

In the case study we focus on combining two data sets that come from two content providers: 1. The indices from the Amsterdam City Archives, and 2. the Occasional Poetry data. The latter is used to make a selection of these data sets by which we create a subset that is both relevant for our case study and is workable in terms of size.

We limit the data to resources related to the actors in the Occasional Poetry data set, in particular only the persons that we reconciled with at least one mention in the Amsterdam City Archives data. We create this subset using a SPARQL query that constructs a copy of the data that only includes events in which at least one disambiguated person was mentioned. For each of these events, it gives the date of the event, the type of event, the persons participating in the event, and the role in which they participate (e.g. bride, or witness). In total, this produces a data set that contains 7,339 events and 22,073 persons, of whom 3,839 have been disambiguated (i.e. they participate in at least two events: one from the Occasional Poetry data set, and one from the City Archives' data sets). The resources are following a basic format that models resources as part of an event in a particular role, as shown in figure 7.2. The SPARQL query and the subset itself can be seen in our documentation on GitHub.[2]

The goal is to maximise the disambiguation of entities in the subset, so that, ideally, a single (disambiguated) person participates in more than one event. With this, we will be able to gain insight into the lives of persons in this data set, such as their lifespan and their social or professional network, and the motivation behind creative production in the 17th and 18th

[2] `https://github.com/knaw-huc/golden-agents-occasional-poetry`

century in Amsterdam. As mentioned in section 3.4, we already made connections from the Occasional Poetry data set to the data sets of the Amsterdam City Archives using the Lenticular Lens tool [35]. These links were subsequently all manually validated.

Then we extend these links by making use of the node embeddings method to in particular disambiguate peripheral entities. These are most often entities that do not occur in the role of bride, groom, father, or mother, and that only are mentioned in the Amsterdam City Archives data set. We use three definite rules and no probabilistic rules in the case study. Entities i and j are considered not to be duplicates if any of the following conditions hold:

1. Both i and j participate in the same event.
2. Both i and j are mentioned in the role of child in a baptism record (assuming there are no duplicate baptism records).
3. i is mentioned in the role of a child in a baptism record and j is mentioned anywhere else in any role at an earlier date.

Finally, a domain expert validates the results of the embedding to assess the quality of the results as well as their usability for future research. This is done by indicating whether or not a person resource (represented by a contextualised URI) refers to the same entity as other URIs in the same cluster. Extra metadata is given to ease the validation process, such as a person's name, their role, and the type of event they participate in. If necessary, the expert can inspect a scan of the original handwritten document. Though not needed for the scoring of the method, we do generate an RDF linkset from the result for usage in the Golden Agents project which can be found, together with the data, in the repository.[3]

7.3 Results

Since we have a ground truth for the experiment on domain knowledge, we report both precision and recall. Then we use the F-Score to determine optimal values for θ. This optimal value is used again in the case study. Table 7.1 shows the results of our experiment. High θ values will produce very high precision, as many small (or even singleton) connected components are created. Each component will likely be composed of pairs with similarly high weights, suggesting a high likelihood of them referring to the same real-life object. On the other hand, high θ values will yield a low recall, as we exclude many pairs of entities with lower (but still reasonable) cosine similarities. We discuss each performance metric in more detail below.

7.3.1 Precision

From table 7.1 we can see that precision increases for all three combinations of rules. For very high values of θ, precision does not increase as the rules are unlikely to exclude pairs with very high cosine similarities, showing that the embedding correctly captured (part of) the likeness between entities. When we lower θ to more reasonable values, the rules start to

[3] https://github.com/knaw-huc/golden-agents-occasional-poetry

	Precision						Recall					
θ	0.60	0.65	0.70	0.75	0.80	0.85	0.60	0.65	0.70	0.75	0.80	0.85
N	0.65	0.74	0.82	0.86	0.91	0.93	0.69	0.66	0.62	0.58	0.52	0.42
D	0.87	0.90	0.93	0.93	0.94	0.95	0.68	0.66	0.61	0.57	0.51	0.42
P	0.69	0.77	0.85	0.89	0.92	0.94	0.68	0.65	0.61	0.57	0.51	0.42
A	0.87	0.91	0.93	0.93	0.94	0.95	0.68	0.65	0.61	0.57	0.51	0.42
	F1-Score						$F\frac{1}{2}$-Score					
θ	0.60	0.65	0.70	0.75	0.80	0.85	0.60	0.65	0.70	0.75	0.80	0.85
N	0.67	0.70	0.71	0.69	0.66	0.58	0.66	0.72	0.77	0.79	0.79	0.75
D	0.76	0.76	0.74	0.71	0.66	0.58	0.82	0.84	0.84	0.83	0.81	0.76
P	0.68	0.70	0.71	0.69	0.65	0.58	0.69	0.74	0.79	0.80	0.79	0.75
A	0.76	0.76	0.74	0.71	0.66	0.58	0.83	0.84	0.84	0.83	0.81	0.76

Table 7.1: *Precision, recall, F1 and $F\frac{1}{2}$-Scores for a range of different θ values and the following rule sets: **N**o rules, **D**efinite rules, **P**robabilistic rules and **A**ll rules*

exclude some of the pairs of entities which the embedding encoded as likely but not certainly referring to the same entity.

7.3.2 Recall

Table 7.1 shows that there is a very slight decrease in recall when rules are applied. We surmise that the decreases in recall are caused by a conflict between the data and the judgement of the domain experts who created the ground truth. For instance, some rules can cause false-negative pairs to be created when a domain expert previously judged a pair to be correct, even though it conflicts with a rule, thereby decreasing recall. This is usually caused by errors or uncertainty in the data, which is not uncommon in these kinds of corpora.

7.3.3 F-Scores

The F-Score can be used to pick an appropriate value for θ based on both precision and recall. It is possible to adjust the F-Score to give greater weight to either precision or recall, depending on the situation. In our case, we report both F1 and $F\frac{1}{2}$-Score, which weighs precision twice as high as recall, as in our particular case precision is more important than recall. This is due to the fact that errors in the entity resolution process can complicate the further analysis of the data by historians. A threshold of $\theta = 0.70$ (i.e. the score with the highest $F\frac{1}{2}$-Score) gives the optimum result when precision is valued twice as high as recall, otherwise the threshold $\theta = 0.65$ is best.

7.3.4 Results of the Occasional Poetry Case Study

We based the choice of $\theta = 0.70$ (as explained in section 7.2) on the outcome of the experiment, also taking into consideration that this rendered a result size that can be validated in a reasonable time (within a week) by a domain expert. This resulted in $9,151$ entities (references to persons) being clustered into $3,400$ distinct clusters, where each cluster represents the occurrences of a single real-life person. Of these $9,151$ entities, $8,326$ were correctly clustered according to the domain expert. Furthermore, 45 entities could not be confidently attributed as either correct or incorrect due to the lack of available information needed for the validation. This means that we achieve a precision between 0.91, if all 45 entities are incorrect, and 0.92 if they are all correct. We are unable to compute the recall, as it is not feasible due to the size, scope, and selection process of the data to determine, for each cluster, which entities are missing. However, we are able to present a list of common errors and points of improvement that were found during the validation process:

- In some cases, the logical ordering of baptism in the role of child, then notice of marriage as husband or wife, then baptism again in the role of father or mother could have been taken into account. Nonetheless, it should be said that people could and did remarry, so strict enforcement of a rule that states that baptism always takes place after a marriage, could introduce additional errors while removing others.
- Some entities were erroneously clustered. Adding additional relations to other nodes (e.g. religion types, church information) in combination with formulating extra rules could have been used to refine the disambiguation.
- The common occurrence of some patrician family names causes them to be put together into a single cluster, e.g. in the case of the name ‘Jan Six’ that appears in every generation of this family.
- In some cases, two entities with very different names were put into the same cluster. This type of error can occur when two entities are clustered together with a similarity higher than the threshold θ. It is relatively simple to remove these errors with an additional string similarity check. (This is something we plan for future work.)

7.4 Conclusion

We have shown that it is possible to combine symbolic and sub-symbolic knowledge in such a way that it improves the performance of an existing entity resolution method. However, the interaction between the two is not always obvious. Introducing rules which, at first glance, should always improve performance may, in fact, worsen performance if there are errors in the data. In future work we plan on including positive evidence as well, that is, the inclusion of rules which, if applicable, indicate that entities are more likely to be duplicates. This can be done with logical rules, as well as by including information from symbolic methods such as Idrissou et al. [36].

The case study shows that the method works on a larger data set as well and that it gives good results. In future work, we plan to include more contextual information and distinctive attributes to the resources in the graph, such as religions and locations, which potentially improves the entity resolution outcome. Additionally, having a data set with thousands of disambiguated interconnected entities makes it possible to perform community detection. This could reveal previously hidden patterns such as larger networks of people who were somehow connected to each other, either by social or professional relation. In particular, our method of entity resolution shows promise to work well in other deed types of the Amsterdam Notarial Archives, as they are very similar in their structure.

8 Conclusions

In this dissertation we have presented a novel framework called Disambed for the disambiguation of entities in knowledge graphs using node embeddings. Our framework is specialized in dealing with the specific issues that occur in, but are not limited to, historical data. In general, the use of semantic web technology and machine learning techniques in the field of computational humanities still has much potential for further growth. Where previously there were major hurdles in the quality and availability of data, this has only recently been partially overcome with the release of relatively large data sets. These sets contain hundreds of millions of triples and have no major errors that would preclude comparisons on a large scale, such as cross-contamination between properties, in addition to comprehensive ontologies. Cross-contamination between properties can be, for example, concatenating the year of birth with the name of a person. However, one major hurdle remains: the lack of labeled data, a ground truth consisting of matching entity pairs. This will be necessary if one is to apply in the future, for instance, recent deep-learning techniques that, while able to learn complicated structures and rules, need a large quantity of labeled data to do so. We discuss the possibilities of such methods in the future work section of this chapter.

With Disambed we have been able to start disambiguating ordinary persons from Amsterdam during the Dutch Golden Age. By using multiple data sets, each with its own context, we were able to link them with high precision and acceptable recall. This is an important first step that enables further research into the relationships and communities these people formed. We have worked in the confines of the Golden Agents project where, in addition to a scientific contribution, certain requirements had to be met. These requirements were contributions to the project goals, which in this case was software that could be used by others to perform entity resolution. As such, this software had to be configurable and able to run with no third party dependencies. In addition to this, we worked with the limitations that occur in the field of Digital Humanities, such as having little access to a ground truth. This gave requirements such as on what specific data the framework had to perform well and added restrictions such as the need to prepare the data when it was not of sufficient quality. Much work needed to be done to first properly prepare the data for disambiguation, as there are errors and ambiguities in the data that needed correcting. Errors such as cross-contamination of attributes (e.g. appending the birth date after the name) needed to be fixed at the source by data experts. Then we spent time finding good representations of entities, where for instance we had to take into

account that slight variations in peoples names do not make their respective contexts disjoint. Additionally, we had to research ways of clustering these persons with the added difficulties of not knowing the total number of clusters and also what the sizes of those clusters were. There are, for instance, many singleton clusters, as many people only occur once in the data. Others, like those who had many children and whose records were preserved, might occur as many as 15 times. Lastly, we introduced domain knowledge in order to add additional constraints to the clustering algorithms, thereby refining the clusters by excluding impossible or very unlikely matches. In the rest of this chapter we will give a short summary per major topic and discuss what we have learned and how we might do things in the future.

8.1 Historical Data

Semantic web technology is increasingly used by researchers in the humanities such as (cultural) historians, literary scholars and art historians, to answer important - yet with traditional means often hard to answer - questions in their respective (sub) fields. Not only does this technology make it possible for the data to be more easily accessed and large scaled, it also facilitates the integration of other relevant data sources, allowing for the answering of questions that are only answerable when different data sets are combined. The conversion of existing data sets to RDF remains a work in progress and is done by experts of both semantic web technology and the cultural domain of the data set. In chapters 3 and 4 we have seen that much effort is required to first clean up and then properly structure the data with a custom ontology. The data from historical sources can have a multitude of issues that will need to be handled first before it can be properly processed.

In addition to generating RDF data and creating an ontology, another step is required in order for the method described in this dissertation to work. That is, the input knowledge graphs first need to be combined in such a way that, when the entity neighborhoods are computed, these neighborhoods contain the maximum amount of relevant entities for the task at hand. When this is indeed the case and these neighborhoods do contain relevant entities, such as possible duplicates, the embedding algorithm will in turn allocate similar vector coordinates to these possible duplicates. Therefore they tend to end up in the same cluster. It is therefore important to make as many connections between potentially relevant entities between different (and inside of) knowledge graphs as possible. This is not as easy as it may seem, as there are many possible data types and one has to also consider the semantic context of each property in order to properly compare them. This inherently requires at least some input of domain experts. That is, a domain expert can tell which properties are best compared and in what way. For example, the cultural context may be important when comparing properties such as the dates of baptism and birth events, or comparing notice of marriage dates with the dates of the marriages themselves.

Many things were learnt during the creation of the processes described above. For instance, the fact that decentralised data is only possible under assumption of at least some coherent vision on how the data should be modeled. Of course, differences in ontology and structure between data sets are to be expected, and one can argue that this is the whole point of the endeavor. However, each data set itself should be properly structured and coherent with

regards to its own ontology. There are errors that preclude a data set from being properly integrated. An example of such an error is cross-contamination between properties, where for example a date of birth is combined with a persons name, which is meant as an informative label but does not fit the semantics of the used property. Another example is having no particular rules on how to represent uncertainties. For instance, when the exact year of an individuals birth is unknown, this is sometimes represented as '165X'. Such a date is invalid and most software will either not load a knowledge graph that contains these dates, or give an error during query time. Instead, a more principled way is to use the graph structure to model the uncertainty. For example, one can split an uncertain date into two parts, the earliest date possible and the latest date possible respectively. The process of perfecting the data is very costly and time consuming and requires expertise on multiple levels. This is very difficult to achieve under strict budgets, where data quality does not have high priority. Therefore, in most realistic circumstances, the assumption of a coherent vision does not hold. This leads to the requirement of a centralized authority which can enforce certain structures and rules, in addition to having an accessible repository for the data.

Much future work can be done in regard to the data described in this dissertation. For instance, the probate inventory data can be improved by adding roles to the participants of the events. In addition to this, a check can be done to ascertain who was deceased in the case of testaments in the notarial archives. Another example of future work would be the linking of data sets containing production data, such as the STCN (described in section 3.3) and occasional poetry (described in section 3.4). When these two data sets are properly linked on the level of person entities with, for example, Disambed, many other endeavors become possible. For example, the application of community detection in order to find out who worked with who and which factors played a role (such as religion) in the formation of these communities. Another example is research into the contribution of minorities in Amsterdam, such as artists from Antwerp and the black community. More generally, a national effort in standardization, e.g. by linking to common thesauri, on concepts such as person types, event types, role types, document types, locations types and occupation types may be necessary. In addition to the standardization, data sustainability is also important. Researchers require a place to store valuable interpretations, which preferably can be accessed by third parties, such as with a triple-store. The absence of which makes it difficult to integrate these contributions in new research and important insights may be lost.

8.2 Machine Learning Techniques

Embeddings are a well known machine learning technique for creating numerical representations for entities that are otherwise difficult to process, such as words in a corpus or, as in our case, nodes in a graph. When we assign a numerical vector representation to an entity, we say that we embed that entity. In chapter 5 we have discussed how relevant nodes in an RDF graph (such as all nodes of type `foaf:Person`) are embedded, based on their graph neighborhood structure, into a d dimensional space. These entities are embedded in such a way that, when they have strongly overlapping neighborhoods in the RDF graph, they have similar embedding vectors (i.e similar coordinates in the embedding space). A useful feature of embeddings is that, when constructed appropriately, the similarity between entities can be

calculated (in parallel) using Euclidean distance or cosine similarity, enabling further processing such as clustering entities based on these computed similarities.

A disadvantage of this method of embedding entities is that it is very hard, once two entities are embedded, to determine exactly why they were assigned their respective vectors. Additionally, the semantic meaning of the dimensions in the embedding is very hard to understand. This makes it difficult to debug and even to determine whether an error occurred in the first place. Related to this issue, much work has been done in the field of explainable AI. This field aims at creating tools and methods that help in explaining why certain decisions or predictions were made by AI algorithms. This helps in creating (public) trust in the outcomes of these algorithms and allows scientists to refine the algorithms to create better output. In this case 'better' can be a higher performance in terms of precision and recall, but it can also mean a fairer outcome, where for instance people with certain (combinations of) properties in the data are not unfairly excluded from outcomes such as the allocation of resources. In light of the desire to be more explainable and transparent in the *output* specifically, we highlight the beginnings of a potential solution below. The intuition here is that, if one can easily interpret the output, it is also easier to determine which patterns were learned by the algorithm. For example, a domain expert could determine that all patterns are based on family relations.

One can take the view that determining whether two entities are duplicates can be achieved by searching for the existence of certain important paths in the graph. These paths together contribute to an increase of the probability that two entities are duplicates. In this case, a path is defined as a sequence of predicates. Two important advantages to such a description are that it is easy to interpret by a domain expert by observing the predicates and that a path can also be trivially encoded as triple patterns for a SPARQL query. A triple pattern is like a triple, except that each of the subject, predicate and object may be a variable. This interpretability of the output can greatly help the issue of explainability which may be lacking in other methods, such as embedding based methods like Disambed. We give an example for a path in the case of the Amsterdam City Archives data. In this case there can exist a path from a person entity that participates in event A to another person entity in event B, containing a third person entity that participated in both event A and event B. From the predicates in this path we can infer that the third person has taken the role of mother and witness in events A and B respectively. The existence of this path can contribute to an increase of the probability that these first two entities are duplicates. The objective is then to extract these relevant paths from all possible paths between the first two entities. Naturally, the search space of all possible paths is very large and would need to be constrained by, for example, only searching locally in the graph by restricting the minimum and maximum lengths of paths. We highlight a possible solution next.

In recent years deep learning techniques, such as attention, which have been developed in the field of large language models, may help in achieving this goal of finding paths. In general, attention works by focusing the learning algorithm on the input that is most relevant for the task at hand. First, we note that in the case of knowledge graphs the unit of operation (also called a token) is a triple. A sequence of triples, where the object of one triple is the subject of the next triple is the same as a path in the knowledge graph. Using this notion of triples, the technique of attention can be leveraged to focus only on those paths that are

important for observing a duplicate pair. An advantage to this method would be that the extracted paths can be combined as a conjunction in a SPARQL query. Finding all duplicate pairs would then amount to running this query. Another advantage is that this query is easy to interpret, thus a domain expert can at least infer that the results make sense in the context of the data. Two disadvantages of this method are that many example duplicate pairs are needed for training and that, like previous machine learning methods, it is not guaranteed to be transitive in its judgment of which entities are duplicates.

8.3 Clustering Entities

In the previous section we briefly discussed how we create embeddings, their associated issues and what possible future work lies in the direction of applying machine learning techniques on knowledge graphs for the purpose of entity resolution. One of the issues of these supervised machine learning techniques based on pairwise matches is their inability to enforce transitivity in their results. For example, when given a trained classifier and entities A, B and C, the classifier may consider the pairs (A,B) and (B,C) to be duplicates, but there is no guarantee that the algorithm also considers the entities in pair (A,C) to be duplicates. When a situation occurs such as where the pair (A,C) is classified as not a duplicate, a compromise must be reached. One solution is to simply list all pairs classified as duplicate and then take the transitive closure, which will include any missing pairs that can be inferred by transitivity. A more principled way may be to not use a classifier but instead use a clustering algorithm that will take into account all or most of the computed similarities from the embedding. The output of this clustering algorithm should be a set of clusters such that, first, as few low similarity pairs are put in the same cluster as possible, and second, where as few high similarity pairs are put in different clusters as possible. In chapter 6 we have described and compared a number of different clustering algorithms which perform this task. Furthermore, we have shown that our method can handle the complexities of real knowledge graphs from the cultural heritage domain and is able to yield a result with high precision while still having good recall. Pairwise similarity information, such as the cosine similarity from an embedding, can be used to group entities into connected components, which act as preliminary clusters. These components are then further spit up into cliques, where each clique acts as a cluster of entities which we regard as all referring to the same real life object (such as a person).

Both supervised and unsupervised methods yield acceptable results. The main issue with the supervised approach is that a ground truth is necessary. It is one of the assumptions of this project that this is either not the case, or very expensive to obtain. A domain expert would have to manually label example pairs of entities, where immediately the additional problem arises of which pairs to label. This can be somewhat mitigated by using data from another method. However, the main problem of requiring labeled pairs persists. The unsupervised method requires no information other than a user-set threshold θ and an embedding. The experiment with the DBLP KG shows that the results can be replicated on a different KG with similar structural properties.

8.4 Including Domain Knowledge

In the previous section we described how clustering techniques can be used to perform entity resolution on embeddings. Nevertheless, these models may produce errors in the entity resolution outcome. That is, a false positive (type 1 error) can occur, where pairs of entities may be identified as duplicate while in reality they are not. Alternatively a false negative (type 2 error) may happen, where two duplicate entities are assigned to different clusters. These errors may come about due to inherent weaknesses in the embedding technique, or incomplete and unreliable information in the original data. Due to the lack of ground truth, a very common occurrence in the field of Digital Humanities, it is not viable to switch to a supervised method in an attempt to improve performance. Therefore, it is necessary to use other types of information, such as rational axioms and domain knowledge, to improve the entity resolution method. In chapter 7, we have focused on resolving type 1 errors.

Two examples of such domain-specific knowledge are:

1. Two entities cannot be identical if they occur in the same civil registration record. This captures many different semantics, depending on the record type. For example, this captures the idea that one cannot marry oneself, one cannot report oneself as deceased, and one cannot be a witness to an event where one is the principal participant.

2. Two entities, one with birth date x, the other with marriage date y cannot be identical if $x > y$. These kinds of rules are dependent on the record type, as records of other types do not contain the necessary information. For example, a rule that uses semantics on marriage and birth needs a marriage record to get a marriage date and another record with birth date information, or, for example, a baptism record to approximate the birth date.

These rules are created with the help of domain- and data experts. The purpose of these rules is to test the conclusion of a sub-symbolic method that two entities are duplicates. Furthermore, a complication to the use of such rules is that domain-specific knowledge can involve uncertainty due to the ambiguity in the data. For example, if we want to exploit the fact that a person cannot be born after they have died, one must be able to identify the born and dead persons unambiguously. The difficulty is that in this type of archival sources it is not a given that the person is actively involved in the registration event when they are mentioned. It could even be that the person is already deceased and is solely used as a disambiguating description for someone else (e.g. *Claartje Jans, widow of Cornelis Pieters*). If this happens in a burial registration or the registration of a testament, we need a rule that is aware of this knowledge and we do not treat the fact of a person's death as 100% certain, but see this as relative to the number of persons involved in the event. This uncertainty then propagates to the conclusion of the rule.

We have shown that it is possible to combine symbolic and sub-symbolic knowledge in such a way that it improves the performance of an existing entity resolution method. However, the interaction between the two types of knowledge is not always obvious. Introducing rules which, at first glance, should always improve performance may, in fact, worsen performance if there are errors in the data. In future work we plan on including positive evidence as well,

that is, the inclusion of rules which, if applicable, indicate that entities are more likely to be duplicates. This can be done with logical rules, as well as by including information from symbolic methods such as presented by Idrissou et al. [36].

Bibliography

[1] Manel Achichi, Zohra Bellahsene, Mohamed Ben Ellefi, and Konstantin Todorov. Linking and disambiguating entities across heterogeneous RDF graphs. *J. Web Semant.*, 55: 108–121, 2019. doi: 10.1016/j.websem.2018.12.003. URL https://doi.org/10.1016/j.websem.2018.12.003.

[2] Juriaan Baas, Mehdi M. Dastani, and Ad Feelders. Entity matching in digital humanities knowledge graphs. In Maud Ehrmann, Folgert Karsdorp, Melvin Wevers, Tara Lee Andrews, Manuel Burghardt, Mike Kestemont, Enrique Manjavacas, Michael Piotrowski, and Joris van Zundert, editors, *Proceedings of the Conference on Computational Humanities Research, CHR2021, Amsterdam, The Netherlands, November 17-19, 2021*, volume 2989 of *CEUR Workshop Proceedings*, pages 1–15. CEUR-WS.org, 2021. URL https://ceur-ws.org/Vol-2989/long_paper5.pdf.

[3] Jurian Baas, Mehdi Dastani, and A. J. Feelders. Graph embeddings for enrichment of historical data. In *Workshop on Graph Embedding and Data Mining (GEM) 2019*, 2019.

[4] Jurian Baas, Mehdi Dastani, and Ad Feelders. Tailored graph embeddings for entity alignment on historical data. In Maria Indrawan-Santiago, Eric Pardede, Ivan Luiz Salvadori, Matthias Steinbauer, Ismail Khalil, and Gabriele Kotsis, editors, *iiWAS '20: The 22nd International Conference on Information Integration and Web-based Applications & Services, Virtual Event / Chiang Mai, Thailand, November 30 - December 2, 2020*, pages 125–133. ACM, 2020. doi: 10.1145/3428757.3429111. URL https://doi.org/10.1145/3428757.3429111.

[5] Jurian Baas, Mehdi Dastani, and A. J. Feelders. Exploiting transitivity for entity matching. In Ruben Verborgh, Anastasia Dimou, Aidan Hogan, Claudia d'Amato, Ilaria Tiddi, Arne Bröring, Simon Maier, Femke Ongenae, Riccardo Tommasini, and Mehwish Alam, editors, *The Semantic Web: ESWC 2021 Satellite Events - Virtual Event, June 6-10, 2021, Revised Selected Papers*, volume 12739 of *Lecture Notes in Computer Science*, pages 109–114. Springer, 2021. doi: 10.1007/978-3-030-80418-3_20. URL https://doi.org/10.1007/978-3-030-80418-3_20.

[6] Jurian Baas, Mehdi M. Dastani, and A. J. Feelders. Combining node embeddings with domain knowledge for identity resolution. *Graphs and Networks in the Humanities*, 6, 2021.

[7] Jurian Baas, Leon van Wissen, Jirsi Reinders, Mehdi M. Dastani, and A. J. Feelders. Adding domain knowledge to improve entity resolution in 17th and 18th century amsterdam archival records. In Anastasia Dimou, Sebastian Neumaier, Tassilo Pellegrini, and Sahar Vahdati, editors, *Towards a Knowledge-Aware AI - SEMANTiCS 2022 - Proceedings of the 18th International Conference on Semantic Systems, 13-15 September 2022, Vienna, Austria*, volume 55 of *Studies on the Semantic Web*, pages 90–104. IOS Press, 2022. doi: 10.3233/SSW220012. URL https://doi.org/10.3233/SSW220012.

[8] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Mach. Learn.*, 56(1-3):89–113, 2004. doi: 10.1023/B:MACH.0000033116.57574.95. URL https://doi.org/10.1023/B:MACH.0000033116.57574.95.

[9] Pavel Berkhin. Bookmark-coloring approach to personalized pagerank computing. *Internet Math.*, 3(1):41–62, 2006. doi: 10.1080/15427951.2006.10129116. URL https://doi.org/10.1080/15427951.2006.10129116.

[10] E. Bernhardsson. Annoy at github. https://github.com/spotify/annoy, 2015.

[11] Sebastian Böcker, Sebastian Briesemeister, and Gunnar W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60(2):316–334, 2011. doi: 10.1007/s00453-009-9339-7. URL https://doi.org/10.1007/s00453-009-9339-7.

[12] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

[13] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Comput. Networks*, 56(18):3825–3833, 2012. doi: 10.1016/j.comnet.2012.10.007. URL https://doi.org/10.1016/j.comnet.2012.10.007.

[14] Lei Cen, Eduard C. Dragut, Luo Si, and Mourad Ouzzani. Author disambiguation by hierarchical agglomerative clustering with adaptive stopping criterion. In Gareth J. F. Jones, Paraic Sheridan, Diane Kelly, Maarten de Rijke, and Tetsuya Sakai, editors, *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, pages 741–744. ACM, 2013. doi: 10.1145/2484028.2484157. URL https://doi.org/10.1145/2484028.2484157.

[15] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In Carles Sierra,

editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 1511–1517. ijcai.org, 2017. doi: 10.24963/ijcai.2017/209. URL https://doi.org/10.24963/ijcai.2017/209.

[16] Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 3998–4004. ijcai.org, 2018. doi: 10.24963/ijcai.2018/556. URL https://doi.org/10.24963/ijcai.2018/556.

[17] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Comput. Surv.*, 53(6), December 2020. ISSN 0360-0300. doi: 10.1145/3418896.

[18] Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. Global RDF vector space embeddings. In Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, volume 10587 of *Lecture Notes in Computer Science*, pages 190–207. Springer, 2017. doi: 10.1007/978-3-319-68288-4_12. URL https://doi.org/10.1007/978-3-319-68288-4_12.

[19] Jan de Mooij, Can Kurtan, Jurian Baas, and Mehdi Dastani. A multiagent framework for querying distributed digital collections. In Ana Paula Rocha, Luc Steels, and H. Jaap van den Herik, editors, *Proceedings of the 12th International Conference on Agents and Artificial Intelligence, ICAART 2020, Volume 1, Valletta, Malta, February 22-24, 2020*, pages 515–521. SCITEPRESS, 2020. doi: 10.5220/0009154005150521. URL https://doi.org/10.5220/0009154005150521.

[20] Jan de Mooij, Can Kurtan, Jurian Baas, and Mehdi Dastani. A computational framework for organizing and querying cultural heritage archives. *ACM Journal on Computing and Cultural Heritage*, 15(3):45:1–45:25, 2022. doi: 10.1145/3485843. URL https://doi.org/10.1145/3485843.

[21] Kien Do, Truyen Tran, and Svetha Venkatesh. Knowledge graph embedding with multiple relation projections. In *24th International Conference on Pattern Recognition, ICPR 2018, Beijing, China, August 20-24, 2018*, pages 332–337. IEEE Computer Society, 2018. doi: 10.1109/ICPR.2018.8545027. URL https://doi.org/10.1109/ICPR.2018.8545027.

[22] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011. doi: 10.5555/1953048.2021068. URL https://dl.acm.org/doi/10.5555/1953048.2021068.

[23] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. Distributed representations of tuples for entity resolution. *Proc. VLDB Endow.*, 11(11):1454–1467, 2018. doi: 10.14778/3236187.3236198. URL http://www.vldb.org/pvldb/vol11/p1454-ebraheem.pdf.

[24] I. Efremova. *Mining social structures from genealogical data*. Phd thesis, School of Mathematics and Computer Science Technische Universiteit Eindhoven, April 2016.

[25] Julia Efremova, Bijan Ranjbar Sahraei, Hossein Rahmani, Frans A. Oliehoek, Toon Calders, Karl Tuyls, and Gerhard Weiss. Multi-source entity resolution for genealogical data. In Gerrit Bloothooft, Peter Christen, Kees Mandemakers, and Marijn Schraagen, editors, *Population Reconstruction*, pages 129–154. Springer, 2015. doi: 10.1007/978-3-319-19884-2_7. URL https://doi.org/10.1007/978-3-319-19884-2_7.

[26] Micha Elsner and Warren Schudy. Bounding and comparing methods for correlation clustering beyond ilp. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, page 19–27, USA, 2009. Association for Computational Linguistics. ISBN 9781932432350.

[27] Phan H Giang. A machine learning approach to create blocking criteria for record linkage. *Health care management science*, 18:93–105, 2015. doi: 10.1007/s10729-014-9276-0. URL https://doi.org/10.1007/s10729-014-9276-0.

[28] Martin Grötschel and Yoshiko Wakabayashi. A cutting plane algorithm for a clustering problem. *Math. Program.*, 45(1-3):59–96, 1989. doi: 10.1007/BF01589097. URL https://doi.org/10.1007/BF01589097.

[29] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM, 2016. doi: 10.1145/2939672.2939754. URL https://doi.org/10.1145/2939672.2939754.

[30] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3887–3896. PMLR, 2020. URL http://proceedings.mlr.press/v119/guo20h.html.

[31] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 192–202. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/d16-1019. URL https://doi.org/10.18653/v1/d16-1019.

[32] Mohammad Al Hasan and Mohammed J. Zaki. A survey of link prediction in social networks. In Charu C. Aggarwal, editor, *Social Network Data Analytics*, pages 243–275. Springer, 2011. doi: 10.1007/978-1-4419-8462-3_9. URL https://doi.org/10.1007/978-1-4419-8462-3_9.

[33] Oktie Hassanzadeh and Renée J. Miller. Creating probabilistic databases from duplicated data. *VLDB J.*, 18(5):1141–1166, 2009. doi: 10.1007/s00778-009-0161-2. URL https://doi.org/10.1007/s00778-009-0161-2.

[34] Barry Hendriks, Paul Groth, and Marieke van Erp. *Recognising and linking entities in old dutch text: A case study on voc notary records*, volume 2810 of *CEUR Workshop Proceedings*, pages 25–36. CEUR Workshop Proceedings, 2021. Publisher Copyright: Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[35] Al Idrissou, Leon Van Wissen, and Veruska Zamborlini. The lenticular lens: Addressing various aspects of entity disambiguation in the semantic web, February 2022. Graphs and Networks in the Humanities 2022, 3-4 February. Amsterdam, The Netherlands.

[36] Al Koudous Idrissou, Rinke Hoekstra, Frank van Harmelen, Ali Khalili, and Peter van den Besselaar. Is my: sameas the same as your: sameas?: Lenticular lenses for context-specific identity. In Óscar Corcho, Krzysztof Janowicz, Giuseppe Rizzo, Ilaria Tiddi, and Daniel Garijo, editors, *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, pages 23:1–23:8. ACM, 2017. doi: 10.1145/3148011.3148029. URL https://doi.org/10.1145/3148011.3148029.

[37] Al Koudous Idrissou, Veruska Zamborlini, Frank van Harmelen, and Chiara Latronico. Contextual entity disambiguation in domains with weak identity criteria: Disambiguating golden age amsterdamers. In Mayank Kejriwal, Pedro A. Szekely, and Raphaël Troncy, editors, *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, Marina Del Rey, CA, USA, November 19-21, 2019*, pages 259–262. ACM, 2019. doi: 10.1145/3360901.3364440. URL https://doi.org/10.1145/3360901.3364440.

[38] Al Koudous Idrissou, Veruska Zamborlini, Frank van Harmelen, and Chiara Latronico. Contextual entity disambiguation in domains with weak identity criteria: Disambiguating golden age amsterdamers. In Mayank Kejriwal, Pedro A. Szekely, and Raphaël Troncy, editors, *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, Marina Del Rey, CA, USA, November 19-21, 2019*, pages 259–262. ACM, 2019. doi: 10.1145/3360901.3364440. URL https://doi.org/10.1145/3360901.3364440.

[39] Anja Jentzsch, Robert Isele, and Christian Bizer. Silk - generating RDF links while publishing or consuming linked data. In Axel Polleres and Huajun Chen, editors, *Proceedings of the ISWC 2010 Posters & Demonstrations Track: Collected Abstracts, Shanghai, China, November 9, 2010*, volume 658 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010. URL https://ceur-ws.org/Vol-658/paper519.pdf.

[40] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[41] Mikko Koho, Petri Leskinen, and Eero Hyvönen. Integrating historical person registers as linked open data in the warsampo knowledge graph. In Eva Blomqvist, Paul Groth, Victor de Boer, Tassilo Pellegrini, Mehwish Alam, Tobias Käfer, Peter Kieseberg, Sabrina Kirrane, Albert Meroño-Peñuela, and Harshvardhan J. Pandit, editors, *Semantic Systems. In the Era of Knowledge Graphs - 16th International Conference on Semantic Systems, SEMANTiCS 2020, Amsterdam, The Netherlands, September 7-10, 2020, Proceedings*, volume 12378 of *Lecture Notes in Computer Science*, pages 118–126. Springer, 2020. doi: 10.1007/978-3-030-59833-4_8. URL https://doi.org/10.1007/978-3-030-59833-4_8.

[42] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press, 2014. ISBN 978-1107077232. URL http://www.mmds.org/.

[43] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013. URL http://arxiv.org/abs/1301.3781.

[44] Alexis Morris, Yannis Velegrakis, and Paolo Bouquet. Entity identification on the semantic web. In Aldo Gangemi, Johannes Keizer, Valentina Presutti, and Heiko Stoermer, editors, *Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP2008), Rome, Italy, December 15-17, 2008*, volume 426 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008. URL https://ceur-ws.org/Vol-426/swap2008_submission_29.pdf.

[45] Markus Nentwig, Anika Groß, and Erhard Rahm. Holistic entity clustering for linked data. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 194–201. IEEE Computer Society, 2016. doi: 10.1109/ICDMW.2016.0035. URL https://doi.org/10.1109/ICDMW.2016.0035.

[46] Axel-Cyrille Ngonga Ngomo and Sören Auer. LIMES - A time-efficient approach for large-scale link discovery on the web of data. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2312–2317. IJCAI/AAAI, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-385. URL https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-385.

[47] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors,

Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011, pages 809–816. Omnipress, 2011. URL https://icml.cc/2011/papers/438_icmlpaper.pdf.

[48] Hao Nie, Xianpei Han, Le Sun, Chi Man Wong, Qiang Chen, Suhui Wu, and Wei Zhang. Global structure and local semantics-preserved embeddings for entity alignment. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3658–3664. ijcai.org, 2020. doi: 10.24963/ijcai.2020/506. URL https://doi.org/10.24963/ijcai.2020/506.

[49] Xing Niu, Shu Rong, Haofen Wang, and Yong Yu. An effective rule miner for instance matching in a web of data. In Xue-wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki, editors, *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 1085–1094. ACM, 2012. doi: 10.1145/2396761.2398406. URL https://doi.org/10.1145/2396761.2398406.

[50] Optimatika. ojAlgo oj! algorithms, 2003–2023. URL https://www.ojalgo.org.

[51] Matteo Paganelli, Paolo Sottovia, Francesco Guerra, and Yannis Velegrakis. Tuner: Fine tuning of rule-based entity matchers. In Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu, editors, *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, pages 2945–2948. ACM, 2019. doi: 10.1145/3357384.3357854. URL https://doi.org/10.1145/3357384.3357854.

[52] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. doi: 10.3115/v1/d14-1162. URL https://doi.org/10.3115/v1/d14-1162.

[53] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani, editors, *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710. ACM, 2014. doi: 10.1145/2623330.2623732. URL https://doi.org/10.1145/2623330.2623732.

[54] Lodewijk Petram, Elvin Dechesne, and Gijsbert Kruithof. Person name vocabulary, 2019. URL https://w3id.org/pnv. Version 1.1.

[55] Zhijian Qu, Shengao Yuan, Rui Chi, Liuchen Chang, and Liang Zhao. Genetic optimization method of pantograph and catenary comprehensive monitor status prediction model based on adadelta deep neural network. *IEEE Access*, 7:23210–23221, 2019. doi: 10.1109/ACCESS.2019.2899074. URL https://doi.org/10.1109/ACCESS.2019.2899074.

[56] Joe Raad, Nathalie Pernelle, and Fatiha Saïs. Detection of contextual identity links in a knowledge base. In Óscar Corcho, Krzysztof Janowicz, Giuseppe Rizzo, Ilaria Tiddi, and Daniel Garijo, editors, *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, pages 8:1–8:8. ACM, 2017. doi: 10.1145/3148011.3148032. URL https://doi.org/10.1145/3148011.3148032.

[57] Joe Raad, Rick Mourits, Auke Rijpma, Ruben Schalk, Richard Zijdeman, Kees Mandemakers, and Albert Meroño-Peñuela. Linking dutch civil certificates. In Alessandro Adamou, Enrico Daga, and Albert Meroño-Peñuela, editors, *WHiSe 2020 Workshop on Humanities in the Semantic Web 2020*, CEUR Workshop Proceedings, pages 47–58. CEUR-WS, October 2020. 3rd Workshop on Humanities in the Semantic Web, WHiSe 2020 ; Conference date: 02-06-2020.

[58] Benjamin Recht, Christopher Ré, Stephen J. Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 693–701, 2011. URL https://proceedings.neurips.cc/paper/2011/hash/218a0aefd1d1a4be65601cc6ddc1520e-Abstract.html.

[59] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=ryQu7f-RZ.

[60] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In Rada Mihalcea, Joyce Yue Chai, and Anoop Sarkar, editors, *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1119–1129. The Association for Computational Linguistics, 2015. doi: 10.3115/v1/n15-1118. URL https://doi.org/10.3115/v1/n15-1118.

[61] Alieh Saeedi, Markus Nentwig, Eric Peukert, and Erhard Rahm. Scalable matching and clustering of entities with FAMER. *Complex Syst. Informatics Model. Q.*, 16:61–83, 2018. doi: 10.7250/csimq.2018-16.04. URL https://doi.org/10.7250/csimq.2018-16.04.

[62] Alieh Saeedi, Eric Peukert, and Erhard Rahm. Using link features for entity clustering in knowledge graphs. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam, editors, *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 576–592. Springer, 2018. doi: 10.1007/978-3-319-93417-4_37. URL https://doi.org/10.1007/978-3-319-93417-4_37.

[63] Yu Shi, Huan Gui, Qi Zhu, Lance M. Kaplan, and Jiawei Han. Aspem: Embedding learning by aspects in heterogeneous information networks. In Martin Ester and Dino Pedreschi, editors, *Proceedings of the 2018 SIAM International Conference on Data Mining, SDM 2018, May 3-5, 2018, San Diego Marriott Mission Valley, San Diego, CA, USA*, pages 144–152. SIAM, 2018. doi: 10.1137/1.9781611975321.16. URL https://doi.org/10.1137/1.9781611975321.16.

[64] Bayu Distiawan Trisedya, Jianzhong Qi, and Rui Zhang. Entity alignment between knowledge graphs using attribute embeddings. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 297–304. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.3301297. URL https://doi.org/10.1609/aaai.v33i01.3301297.

[65] René Van Weeren and Tine De Moor. Counting couples: The marriage banns registers of the city of amsterdam, 1580–1810: Social and economic history. *Research Data Journal for the Humanities and Social Sciences*, 6(1):1 – 45, 2021. doi: https://doi.org/10.1163/24523666-06010002.

[66] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk - A link discovery framework for the web of data. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Kingsley Idehen, editors, *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. URL https://ceur-ws.org/Vol-538/ldow2009_paper13.pdf.

[67] Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1859–1866. AAAI Press, 2015. URL http://ijcai.org/Abstract/15/264.

[68] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1112–1119. AAAI Press, 2014. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531.

[69] Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. Neighborhood matching network for entity alignment. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6477–6487. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.578. URL https://doi.org/10.18653/v1/2020.acl-main.578.

[70] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In Balaji Krishna-

puram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 353–362. ACM, 2016. doi: 10.1145/2939672.2939673. URL https://doi.org/10.1145/2939672.2939673.

[71] Chen Zhao and Yeye He. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2413–2424. ACM, 2019. doi: 10.1145/3308558.3313578. URL https://doi.org/10.1145/3308558.3313578.

[72] Qiannan Zhu, Xiaofei Zhou, Jia Wu, Jianlong Tan, and Li Guo. Neighborhood-aware attentional representation for multilingual knowledge graphs. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 1943–1949. ijcai.org, 2019. doi: 10.24963/ijcai.2019/269. URL https://doi.org/10.24963/ijcai.2019/269.

Summary

Semantic web technology is increasingly being used in projects of humanities researchers, such as historians and literary scholars. This technology makes it easier to access large-scale data sets from the cultural heritage world, such as the indexes on persons and locations of the Amsterdam City Archives. Semantic web technology also facilitates the integration of different data sources into knowledge graphs, which in turn enables cross-data set analyzes that were previously infeasible. This makes it possible, for example, to reconstruct someone's life on the basis of primary archival sources. However, the integration of different historical data sets entails a number of complications. Because the majority of these archival data sets are aimed at providing quick and easy access, it is likely that the same person is included multiple times within a single data set, or may also appear multiple times in different data sets, each time with a new entry and unique identifier. Until these unique entries are resolved and the duplicate entities are disambiguated, it is not yet possible to conduct this type of investigation.

This dissertation offers a solution to this problem and describes a method to reduce, if not solve, the number of duplicates in a set of knowledge graphs by clustering these unique entries, where each cluster represents a single real life object. To this end, it describes the application of so-called 'embeddings': a technique for making computer-readable representations of entities such as nodes in a knowledge graph. The method described in this thesis constructs the embeddings such that a high similarity between two nodes is indicative of a duplicate. However, relying on these pairwise matches is not without risks. For example, the application of a threshold value can lead to a transitivity violation. That is, the entity pairs (A,B) and (B,C) can both have high similarity, but this need not be the case for pair (A,C). To address this issue, we employ algorithms that make use of the pairwise similarities to find clusters that conform as best as possible to the computed similarities. Nevertheless, it happens that these clustering algorithms produce false positives and negatives. To counteract this and significantly improve the clustering results, this work describes the use of domain-specific knowledge and constraints to detect and correct clustering errors. An example of such a restriction is that one cannot marry oneself or that a person is first baptized and then buried.

Samenvatting

Semantic-webtechnologie wordt steeds vaker gebruikt in projecten van onderzoekers uit de geesteswetenschappen. Deze technologie maakt het eenvoudiger om toegang te krijgen tot grootschalige datasets uit de erfgoedwereld, zoals bijvoorbeeld de indexen op personen en locaties van het Stadsarchief Amsterdam. Tevens faciliteert de techniek de integratie van verschillende databronnen, die op haar beurt datasetoverstijgende analyses mogelijk maakt die voorheen onuitvoerbaar waren. Zo wordt het hiermee bijvoorbeeld mogelijk om iemands leven te reconstrureren aan de hand van primaire archiefbronnen. De integratie van verschillende historische datasets brengt echter een aantal complicaties met zich mee. Omdat het merendeel van deze archiefdatasets gericht is op het bieden van een snelle en gemakkelijke toegang, is het waarschijnlijk dat dezelfde persoon meerdere keren is opgenomen in één dataset, of ook meermaals voorkomt in verschillende datasets, telkens met een nieuwe vermelding en unieke identificatiecode. Totdat deze unieke vermeldingen zijn opgelost en de dubbele entiteiten zijn gedisambigueerd, is het nog niet mogelijk om dit type onderzoek uit te voeren.

Dit proefschrift biedt een oplossing voor dit probleem en beschrijft een methode om het aantal duplicaten in een dataset terug te dringen, zo niet op te lossen, door deze unieke vermeldingen van personen te clusteren. Hiervoor beschrijft het de toepassing van zogeheten 'embeddings': een techniek om computerleesbare representaties te maken van entiteiten zoals knopen in een netwerk. De in dit proefschrift beschreven methode construeert de embeddings zodanig dat een hoge gelijkenis tussen twee knopen indicatief is voor een duplicaat. Echter, afgaan op deze paarsgewijze overeenkomsten is niet zonder risicoś. Zo kan de toepassing van een drempelwaarde leiden tot een schending van transitiviteit. De entiteitsparen (A,B) en (B,C) kunnen bijvoorbeeld beide een hoge gelijkenis hebben, maar dit hoeft niet het geval te zijn voor paar (A,C). Om dit probleem op te lossen worden zowel deze paarsgewijze overeenkomsten als eerder berekende overeenkomsten gecombineerd bij het clusteren van entiteiten. Desalniettemin komt het voor dat deze clusteringsalgoritmen valspositieven en -negatieven produceren. Om dit tegen te gaan en de clusteringsresultaten significant te verbeteren beschrijft dit werk de inzet van domeinspecifieke kennis en restricties om clusteringsfouten op te sporen en te corrigeren. Een voorbeeld van dergelijke restrictie is dat men niet met zichzelf kan trouwen of dat een persoon eerst wordt gedoopt en dan wordt begraven.

Acknowledgements

First I would like to thank my daily supervisor Ad Feelders. Our many, sometimes long, conversations about a wide range of topics have made my PhD experience fun and rewarding, especially when the stress of deadlines loomed. I have learned so much from these talks and hope to have many more in the future. Next, I thank my first promotor Mehdi Dastani. If not for you I would not have had the opportunity to write this dissertation and attain my PhD. You have helped me many times over the years and I have learned much from you, especially how to become a better writer. For this I will always be thankful. Finally I would like to thank my second promotor Els Stronks. I have always enjoyed our conversations together and I have always appreciated your insights from a historians perspective.

My deepest thanks also go to Leon van Wissen. Your insights, fun conversations and deep knowledge on the data which were immensely helpful in my own understanding, and Jirsi Reinders, for sharing your knowledge on the historical background of the data and, of course, for your time and effort in validating the large amounts of clusters that were generated by Disambed.

I would also like to thank the committee members Arianna Betti, Emiel Caron, Joris van Eijnatten, Charles van den Heuvel, Yannis Velegrakis for spending their time and effort in evaluating this dissertation and providing valuable feedback.

My special thanks go to my father Ruud for his valuable help in designing the cover and bookmark and also for preparing this dissertation for printing. Lastly, I thank my girlfriend Renate. You have been my partner in life throughout the years and your support and love have made this journey worthwhile and fulfilling.

Curriculum Vitae

Vocational Experience

2018-2023 **PhD candidate**, *Utrecht University*, Utrecht.
Enhancing knowledge graphs using data mining techniques

2017-2022 **Scientific Programmer**, *Utrecht University*, Utrecht.
Developing a multi-agent framework for combining data from the Dutch Golden age

2012-2013 **Software Developer**, *Triple Interactive B.V.*, Reeuwijk.
Developing and maintaining websites and highly customized backend systems

2009-2015 **Owner**, *JB Technologies*, Gouda.
Developing and maintaining websites

Education

2015-2017 **Artificial Intelligence**, *Master*, Utrecht University,
Majored in Intelligent Agent Logics, Multi-Agent Systems and -Learning.
Minored in Pattern Recognition, Data Mining, Games and Agents and Computational Geometry

2008-2015 **Computer Science**, *Bachelor*, Rotterdam University of Applied Sciences, *Minored in Computer Security and Cryptography.*

Master thesis

Title *Predicting crime trends using weather data and machine learning*
Organisation KNMI
Supervisors Dr. Mirna van Hoek (KNMI), Dr. Ad Feelders (Utrecht University)

Bachelor thesis

Title *Categorizing e-mails using machine learning*
Company Technolution
Supervisor Drs. Michael Dubbeldam

External courses -

The Netherlands Research School for Information and Knowledge Systems

2019 **Trends and Topics in Multi Agent Systems**, An overview of methods such as multi agent reinforcement learning, and interactive information retrieval and extraction.

2019 **Advances in Information Retrieval**, An overview of methods such as semantic search, indexing, cross-domain recommendation.

2019 **Foundations of Data Science: Data Mining**, An overview of methods such as deep learning, active learning and causal inference.

2019 **Research Methods and Methodology**, A hands-on course to enable PhD students to make a good research design for their own research project.

External courses - Online

2017 **Learning from Data**, *Taught by Yaser Abu-Mostafa , Professor, California Institute of Technology*.
An introductory course in machine learning that covers the basic theory, algorithms and applications.

2015 **Machine Learning**, *Taught by Andrew Ng, Associate Professor, Stanford University*.
A broad introduction to machine learning, data mining, and statistical pattern recognition.

Research Output

2019 **Graph Embeddings for Enrichment of Historical Data**, *Workshop on Graph Embedding and Data Mining (GEM)*.

2020 **Tailored Graph Embeddings for Entity Alignment on Historical Data**, *International Conference on Information Integration and Web-based Applications and Services*.
DOI: 10.1145/3428757.3429111

2020 **A multi-agent framework for querying distributed digital collections**, *International Conference on Agents and Artificial Intelligence*.
DOI: 10.5220/0009154005150521

2021 **Entity Matching in Digital Humanities Knowledge Graphs**, *Computational Humanities Research*.
NBN:NL:UI:10-1874-423193

2021 **Exploiting Transitivity for Entity Matching**, *European Semantic Web Conference*.
DOI: 10.1007/978-3-030-80418-3_20

2021 **A Computational Framework for Organizing and Querying Cultural Heritage Archives**, *Journal on Computing and Cultural Heritage*.
DOI: 10.1145/3485843

2021 **Combining Node Embeddings with Domain Knowledge for Identity Resolution**, *Graphs and Networks in the Humanities*.
DOI: To be determined

2022 **Adding Domain Knowledge to Improve Entity Resolution in 17th and 18th Century Amsterdam Archival Records**, *SEMANTiCS*.
DOI: 10.3233/SSW220012

Reviewer for Conferences

2023 **GrapHNR 2023**, *Joint Conference of the HNR Community and Graphs & Networks in the Humanities.*

SIKS Dissertation Series

2016
01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
04 Laurens Rietveld (VU), Publishing and Consuming Linked Data
05 Evgeny Sherkhonov (UVA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
06 Michel Wilson (TUD), Robust scheduling in an uncertain environment
07 Jeroen de Man (VU), Measuring and modeling negative emotions for virtual training
08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data
09 Archana Nottamkandath (VU), Trusting Crowdsourced Information on Cultural Artefacts
10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
11 Anne Schuth (UVA), Search Engines that Learn from Their Users
12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
13 Nana Baah Gyan (VU), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UVA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VU), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VU), Refining Statistical Data on the Web
19 Julia Efremova (Tu/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UVA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground

22 Grace Lewis (VU), Software Architecture Strategies for Cyber-Foraging Systems
23 Fei Cai (UVA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VU), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (UvT), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring
32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
33 Peter Bloem (UVA), Single Sample Statistics, exercises in learning from just one example
34 Dennis Schunselaar (TUE), Configurable Process Trees: Elicitation, Analysis, and Enactment
35 Zhaochun Ren (UVA), Monitoring Social Media: Summarization, Classification and Recommendation
36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies
37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
40 Christian Detweiler (TUD), Accounting for Values in Design
41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
42 Spyros Martzoukos (UVA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
44 Thibault Sellam (UVA), Automatic Assistants for Database Exploration
45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
46 Jorge Gallego Perez (UT), Robots to Make you Happy
47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
48 Tanja Buttler (TUD), Collecting Lessons Learned

49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
50 Yan Wang (UVT), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

---

2017 01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
05 Mahdieh Shadi (UVA), Collaboration Behavior
06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
08 Rob Konijn (VU) , Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TUE), Robust Process Mining with Guarantees
13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (UvT), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UVA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UVA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
22 Sara Magliacane (VU), Logics for causal inference under uncertainty
23 David Graus (UVA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VU), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors

28 John Klein (VU), Architecture Practices for Complex Contexts
29 Adel Alhuraibi (UvT), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"
30 Wilma Latuny (UvT), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
35 Martine de Vos (VU), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TUE), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
46 Jan Schneider (OU), Sensor-based Learning Support
47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration
48 Angel Suarez (OU), Collaborative inquiry-based learning

2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations
02 Felix Mannhardt (TUE), Multi-perspective Process Mining
03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction
04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
05 Hugo Huurdeman (UVA), Supporting the Complex Dynamics of the Information Seeking Process
06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems
08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems

09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations
10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology
11 Mahdi Sargolzaei (UVA), Enabling Framework for Service-oriented Collaborative Networks
12 Xixi Lu (TUE), Using behavioral context in process mining
13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
14 Bart Joosten (UVT), Detecting Social Signals with Spatiotemporal Gabor Filters
15 Naser Davarzani (UM), Biomarker discovery in heart failure
16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
17 Jianpeng Zhang (TUE), On Graph Sample Clustering
18 Henriette Nakad (UL), De Notaris en Private Rechtspraak
19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
20 Manxia Liu (RUN), Time and Bayesian Networks
21 Aad Slootmaker (OUN), EMERGO: a generic platform for authoring and playing scenario-based serious games
22 Eric Fernandes de Mello Araújo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
26 Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
27 Maikel Leemans (TUE), Hierarchical Process Mining for Scalable Software Analysis
28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
29 Yu Gu (UVT), Emotion Recognition from Mandarin Speech
30 Wouter Beek, The "K" in "semantic web" stands for "knowledge": scaling semantics to the web

2019 01 Rob van Eijk (UL),Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification
02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
03 Eduardo Gonzalez Lopez de Murillas (TUE), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
05 Sebastiaan van Zelst (TUE), Process Mining with Streaming Data
06 Chris Dijkshoorn (VU), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
07 Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes

09 Fahimeh Alizadeh Moghaddam (UVA), Self-adaptation for energy efficiency in software systems
10 Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
11 Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
12 Jacqueline Heinerman (VU), Better Together
13 Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
14 Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
15 Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
16 Guangming Li (TUE), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
17 Ali Hurriyetoglu (RUN),Extracting actionable information from microtexts
18 Gerard Wagenaar (UU), Artefacts in Agile Team Communication
19 Vincent Koeman (TUD), Tools for Developing Cognitive Agents
20 Chide Groenouwe (UU), Fostering technically augmented human collective intelligence
21 Cong Liu (TUE), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
22 Martin van den Berg (VU),Improving IT Decisions with Enterprise Architecture
23 Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
24 Anca Dumitrache (VU), Truth in Disagreement - Crowdsourcing Labeled Data for Natural Language Processing
25 Emiel van Miltenburg (VU), Pragmatic factors in (automatic) image description
26 Prince Singh (UT), An Integration Platform for Synchromodal Transport
27 Alessandra Antonaci (OUN), The Gamification Design Process applied to (Massive) Open Online Courses
28 Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations
29 Daniel Formolo (VU), Using virtual agents for simulation and training of social skills in safety-critical circumstances
30 Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems
31 Milan Jelisavcic (VU), Alive and Kicking: Baby Steps in Robotics
32 Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games
33 Anil Yaman (TUE), Evolution of Biologically Inspired Learning in Artificial Neural Networks
34 Negar Ahmadi (TUE), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES
35 Lisa Facey-Shaw (OUN), Gamification with digital badges in learning programming
36 Kevin Ackermans (OUN), Designing Video-Enhanced Rubrics to Master Complex Skills

37 Jian Fang (TUD), Database Acceleration on FPGAs
38 Akos Kadar (OUN), Learning visually grounded and multilingual representations

2020
01 Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour
02 Marcos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models
03 Mostafa Deghani (UvA), Learning with Imperfect Supervision for Language Understanding
04 Maarten van Gompel (RUN), Context as Linguistic Bridges
05 Yulong Pei (TUE), On local and global structure mining
06 Preethu Rose Anish (UT), Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support
07 Wim van der Vegt (OUN), Towards a software architecture for reusable game components
08 Ali Mirsoleimani (UL),Structured Parallel Programming for Monte Carlo Tree Search
09 Myriam Traub (UU), Measuring Tool Bias and Improving Data Quality for Digital Humanities Research
10 Alifah Syamsiyah (TUE), In-database Preprocessing for Process Mining
11 Sepideh Mesbah (TUD), Semantic-Enhanced Training Data AugmentationMethods for Long-Tail Entity Recognition Models
12 Ward van Breda (VU), Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment
13 Marco Virgolin (CWI), Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming
14 Mark Raasveldt (CWI/UL), Integrating Analytics with Relational Databases
15 Konstantinos Georgiadis (OUN), Smart CAT: Machine Learning for Configurable Assessments in Serious Games
16 Ilona Wilmont (RUN), Cognitive Aspects of Conceptual Modelling
17 Daniele Di Mitri (OUN), The Multimodal Tutor: Adaptive Feedback from Multimodal Experiences
18 Georgios Methenitis (TUD), Agent Interactions & Mechanisms in Markets with Uncertainties: Electricity Markets in Renewable Energy Systems
19 Guido van Capelleveen (UT), Industrial Symbiosis Recommender Systems
20 Albert Hankel (VU), Embedding Green ICT Maturity in Organisations
21 Karine da Silva Miras de Araujo (VU), Where is the robot?: Life as it could be
22 Maryam Masoud Khamis (RUN), Understanding complex systems implementation through a modeling approach: the case of e-government in Zanzibar
23 Rianne Conijn (UT), The Keys to Writing: A writing analytics approach to studying writing processes using keystroke logging
24 Lenin da Nóbrega Medeiros (VUA/RUN), How are you feeling, human? Towards emotionally supportive chatbots
25 Xin Du (TUE), The Uncertainty in Exceptional Model Mining
26 Krzysztof Leszek Sadowski (UU), GAMBIT: Genetic Algorithm for Model-Based mixed-Integer opTimization
27 Ekaterina Muravyeva (TUD), Personal data and informed consent in an educational context

28 Bibeg Limbu (TUD), Multimodal interaction for deliberate practice: Training complex skills with augmented reality
29 Ioan Gabriel Bucur (RUN), Being Bayesian about Causal Inference
30 Bob Zadok Blok (UL), Creatief, Creatieve, Creatiefst
31 Gongjin Lan (VU), Learning better – From Baby to Better
32 Jason Rhuggenaath (TUE), Revenue management in online markets: pricing and online advertising
33 Rick Gilsing (TUE), Supporting service-dominant business model evaluation in the context of business model innovation
34 Anna Bon (MU), Intervention or Collaboration? Redesigning Information and Communication Technologies for Development
35 Siamak Farshidi (UU), Multi-Criteria Decision-Making in Software Production

2021 01 Francisco Xavier Dos Santos Fonseca (TUD),Location-based Games for Social Interaction in Public Space
02 Rijk Mercuur (TUD), Simulating Human Routines: Integrating Social Practice Theory in Agent-Based Models
03 Seyyed Hadi Hashemi (UVA), Modeling Users Interacting with Smart Devices
04 Ioana Jivet (OU), The Dashboard That Loved Me: Designing adaptive learning analytics for self-regulated learning
05 Davide Dell'Anna (UU), Data-Driven Supervision of Autonomous Systems
06 Daniel Davison (UT), "Hey robot, what do you think?" How children learn with a social robot
07 Armel Lefebvre (UU), Research data management for open science
08 Nardie Fanchamps (OU), The Influence of Sense-Reason-Act Programming on Computational Thinking
09 Cristina Zaga (UT), The Design of Robothings. Non-Anthropomorphic and Non-Verbal Robots to Promote Children's Collaboration Through Play
10 Quinten Meertens (UvA), Misclassification Bias in Statistical Learning
11 Anne van Rossum (UL), Nonparametric Bayesian Methods in Robotic Vision
12 Lei Pi (UL), External Knowledge Absorption in Chinese SMEs
13 Bob R. Schadenberg (UT), Robots for Autistic Children: Understanding and Facilitating Predictability for Engagement in Learning
14 Negin Samaeemofrad (UL), Business Incubators: The Impact of Their Support
15 Onat Ege Adali (TU/e), Transformation of Value Propositions into Resource Re-Configurations through the Business Services Paradigm
16 Esam A. H. Ghaleb (UM), Bimodal emotion recognition from audio-visual cues
17 Dario Dotti (UM), Human Behavior Understanding from motion and bodily cues using deep neural networks
18 Remi Wieten (UU), Bridging the Gap Between Informal Sense-Making Tools and Formal Systems - Facilitating the Construction of Bayesian Networks and Argumentation Frameworks
19 Roberto Verdecchia (VU), Architectural Technical Debt: Identification and Management
20 Masoud Mansoury (TU/e), Understanding and Mitigating Multi-Sided Exposure Bias in Recommender Systems

21 Pedro Thiago Timbó Holanda (CWI), Progressive Indexes
22 Sihang Qiu (TUD), Conversational Crowdsourcing
23 Hugo Manuel Proença (LIACS), Robust rules for prediction and description
24 Kaijie Zhu (TUE), On Efficient Temporal Subgraph Query Processing
25 Eoin Martino Grua (VUA), The Future of E-Health is Mobile: Combining AI and Self-Adaptation to Create Adaptive E-Health Mobile Applications
26 Benno Kruit (CWI & VUA), Reading the Grid: Extending Knowledge Bases from Human-readable Tables
27 Jelte van Waterschoot (UT), Personalized and Personal Conversations: Designing Agents Who Want to Connect With You
28 Christoph Selig (UL), Understanding the Heterogeneity of Corporate Entrepreneurship Programs

2022 01 Judith van Stegeren (UT), Flavor text generation for role-playing video games
02 Paulo da Costa (TU/e), Data-driven Prognostics and Logistics Optimisation: A Deep Learning Journey
03 Ali el Hassouni (VUA), A Model A Day Keeps The Doctor Away: Reinforcement Learning For Personalized Healthcare
04 Ünal Aksu (UU), A Cross-Organizational Process Mining Framework
05 Shiwei Liu (TU/e), Sparse Neural Network Training with In-Time Over-Parameterization
06 Reza Refaei Afshar (TU/e), Machine Learning for Ad Publishers in Real Time Bidding
07 Sambit Praharaj (OU), Measuring the Unmeasurable? Towards Automatic Co-located Collaboration Analytics
08 Maikel L. van Eck (TU/e), Process Mining for Smart Product Design
09 Oana Andreea Inel (VUA), Understanding Events: A Diversity-driven Human-Machine Approach
10 Felipe Moraes Gomes (TUD), Examining the Effectiveness of Collaborative Search Engines
11 Mirjam de Haas (UT), Staying engaged in child-robot interaction, a quantitative approach to studying preschoolers' engagement with robots and tasks during second-language tutoring
12 Guanyi Chen (UU), Computational Generation of Chinese Noun Phrases
13 Xander Wilcke (VUA), Machine Learning on Multimodal Knowledge Graphs: Opportunities, Challenges, and Methods for Learning on Real-World Heterogeneous and Spatially-Oriented Knowledge
14 Michiel Overeem (UU), Evolution of Low-Code Platforms
15 Jelmer Jan Koorn (UU), Work in Process: Unearthing Meaning using Process Mining
16 Pieter Gijsbers (TU/e), Systems for AutoML Research
17 Laura van der Lubbe (VUA), Empowering vulnerable people with serious games and gamification
18 Paris Mavromoustakos Blom (TiU), Player Affect Modelling and Video Game Personalisation
19 Bilge Yigit Ozkan (UU), Cybersecurity Maturity Assessment and Standardisation

20 Fakhra Jabeen (VUA), Dark Side of the Digital Media - Computational Analysis of Negative Human Behaviors on Social Media
21 Seethu Mariyam Christopher (UM), Intelligent Toys for Physical and Cognitive Assessments
22 Alexandra Sierra Rativa (TiU), Virtual Character Design and its potential to foster Empathy, Immersion, and Collaboration Skills in Video Games and Virtual Reality Simulations
23 Ilir Kola (TUD), Enabling Social Situation Awareness in Support Agents
24 Samaneh Heidari (UU), Agents with Social Norms and Values - A framework for agent based social simulations with social norms and personal values
25 Anna L.D. Latour (LU), Optimal decision-making under constraints and uncertainty
26 Anne Dirkson (LU), Knowledge Discovery from Patient Forums: Gaining novel medical insights from patient experiences
27 Christos Athanasiadis (UM), Emotion-aware cross-modal domain adaptation in video sequences
28 Onuralp Ulusoy (UU), Privacy in Collaborative Systems
29 Jan Kolkmeier (UT), From Head Transform to Mind Transplant: Social Interactions in Mixed Reality
30 Dean De Leo (CWI), Analysis of Dynamic Graphs on Sparse Arrays
31 Konstantinos Traganos (TU/e), Tackling Complexity in Smart Manufacturing with Advanced Manufacturing Process Management
32 Cezara Pastrav (UU), Social simulation for socio-ecological systems
33 Brinn Hekkelman (CWI/TUD), Fair Mechanisms for Smart Grid Congestion Management
34 Nimat Ullah (VUA), Mind Your Behaviour: Computational Modelling of Emotion & Desire Regulation for Behaviour Change
35 Mike E.U. Ligthart (VUA), Shaping the Child-Robot Relationship: Interaction Design Patterns for a Sustainable Interaction

---

2023 01 Bojan Simoski (VUA), Untangling the Puzzle of Digital Health Interventions
02 Mariana Rachel Dias da Silva (TiU), Grounded or in flight? What our bodies can tell us about the whereabouts of our thoughts
03 Shabnam Najafian (TUD), User Modeling for Privacy-preserving Explanations in Group Recommendations
04 Gineke Wiggers (UL), The Relevance of Impact: bibliometric-enhanced legal information retrieval
05 Anton Bouter (CWI), Optimal Mixing Evolutionary Algorithms for Large-Scale Real-Valued Optimization, Including Real-World Medical Applications
06 António Pereira Barata (UL), Reliable and Fair Machine Learning for Risk Assessment
07 Tianjin Huang (TU/e), The Roles of Adversarial Examples on Trustworthiness of Deep Learning
08 Lu Yin (TU/e), Knowledge Elicitation using Psychometric Learning
09 Xu Wang (VUA), Scientific Dataset Recommendation with Semantic Techniques
10 Dennis J.N.J. Soemers (UM), Learning State-Action Features for General Game Playing

11 Fawad Taj (VUA), Towards Motivating Machines: Computational Modeling of the Mechanism of Actions for Effective Digital Health Behavior Change Applications
12 Tessel Bogaard (VUA), Using Metadata to Understand Search Behavior in Digital Libraries
13 Injy Sarhan (UU), Open Information Extraction for Knowledge Representation
14 Selma Čaušević (TUD), Energy resilience through self-organization
15 Alvaro Henrique Chaim Correia (TU/e), Insights on Learning Tractable Probabilistic Graphical Models
16 Peter Blomsma (TiU), Building Embodied Conversational Agents: Observations on human nonverbal behaviour as a resource for the development of artificial characters
17 Meike Nauta (UT), Explainable AI and Interpretable Computer Vision – From Oversight to Insight
18 Gustavo Penha (TUD), Designing and Diagnosing Models for Conversational Search and Recommendation
19 George Aalbers (TiU), Digital Traces of the Mind: Using Smartphones to Capture Signals of Well-Being in Individuals
20 Arkadiy Dushatskiy (TUD), Expensive Optimization with Model-Based Evolutionary Algorithms applied to Medical Image Segmentation using Deep Learning
21 Gerrit Jan de Bruin (UL), Network Analysis Methods for Smart Inspection in the Transport Domain
22 Alireza Shojaifar (UU), Volitional Cybersecurity
23 Theo Theunissen (UU), Documentation in Continuous Software Development
24 Agathe Balayn (TUD), Practices Towards Hazardous Failure Diagnosis in Machine Learning