Temporal graph-based clustering for historical record linkage

Work-in-progress paper

Charini Nanayakkara
Research School of Computer Science, The Australian National University
Canberra, ACT, Australia
charini.nanayakkara@anu.edu.au

Peter Christen
Research School of Computer Science, The Australian National University
Canberra, ACT, Australia
peter.christen@anu.edu.au

Thilina Ranbaduge
Research School of Computer Science, The Australian National University
Canberra, ACT, Australia
thilina.ranbaduge@anu.edu.au

ABSTRACT
Research in the social sciences is increasingly based on large and complex data collections, where individual data sets from different domains are linked and integrated to allow advanced analytics. A popular type of data used in such a context are historical censuses, as well as birth, death, and marriage certificates. Individually, such data sets however limit the types of studies that can be conducted. Specifically, it is impossible to track individuals, families, or households over time. Once such data sets are linked and family trees spanning several decades are available it is possible to, for example, investigate how education, health, mobility, employment, and social status influence each other and the lives of people over two or even more generations. A major challenge is however the accurate linkage of historical data sets which is due to data quality and commonly also the lack of ground truth data being available. Unsupervised techniques need to be employed, which can be based on similarity graphs generated by comparing individual records. In this paper we present initial results from clustering birth records from Scotland where we aim to identify all births of the same mother and group siblings into clusters. We extend an existing clustering technique for record linkage by incorporating temporal constraints that must hold between births by the same mother, and propose a novel greedy temporal clustering technique. Experimental results show improvements over non-temporary approaches, however further work is needed to obtain links of high quality.

1 INTRODUCTION
Databases that contain personal information, such as censuses or historical civil registries [25], generally contain multiple records describing the same individual (entity) or group of individuals such as children or homes, where each individual will occur in such databases with different types of roles [7, 8]. A baby is born, then recorded as a daughter or son in a census, and later she or he might marry (as a bride or groom) and become the mother or father of her or his own children. Being able to link such records across different databases will allow the reconstruction of whole populations and open a multitude of studies in the health and social sciences that currently are not feasible on individual databases [3, 20].

The process of identifying the sets of records that correspond to the same individual is known as record linkage, entity resolution, or data matching [6]. Record linkage involves comparing pairs of records to decide if the records of a pair refer to the same entity (known as a match) or to different entities (a non-match). In such a comparison process generally the similarities between the values of a selected set of attributes are compared to decide if a pair of records is similar enough to be classified as a match (if for example the similarities are above a pre-define threshold value). In many application domains this simple pair-wise linkage process does however not provide enough information to identify the relationships between different individuals [7, 11].

Recently, in contrast to traditional pair-wise record linkage, group linkage [24] has received significant attention because of its applicability of linking groups of individuals, such as families or households [8, 15]. The identification of relationships between individuals can enrich data and improve the quality of data, and thus facilitate more sophisticated analysis of different socio-economic factors (such as health, wealth, occupation, and social structure) of large populations [13, 16]. Studying these issues are important to identify how societies evolve over time and discover the changes that influenced and contributed for social evolution [12].

Historical record linkage involves the linkage of historical records, including records from censuses as well as from birth, death, and marriage certificates, to construct longitudinal data sets about a population. Over the past two decades researchers working in different domains have studied the problem of historical record linkage. In 1996 Dillon investigated an approach to link census records from the US and Canada to generate a longitudinal database to examine changes in household structures [10]. The Integrated Public Use Microdata Series (IPUMS, see: https://www.ipums.org/) is a large project initiated by the Minnesota Population Centre (MPC) for linking large demographic data collections. The Life-M project is another example of transforming records from birth, marriage, and
death certificates as well as census records into an intergenerational longitudinal database [2]. The project considers US data from the 19th and 20th centuries and aims to use birth certificates as a basis for historical record linkage of large historical databases.

The Digitising Scotland project [9], which this work is a part of, aims to transcribe and link all civil registration events recorded in Scotland between 1856 and 1973. Around 14 million birth, 11 million death, and 4 million marriage records need to be linked to create a linked database covering the whole population of Scotland spanning more than a century to allow researchers in various domains to conduct studies that are currently impossible to do.

Here we present work-in-progress on a specific step used in traditional family reconstruction as conducted by demographers and historians [25, 27]: the bundling (clustering) of birth records by the same mother to identify siblings. Once siblings groups have been identified, they can be linked to census, marriage, and death records using group linkage techniques [14]. Linked bundles of siblings allow a variety of studies for example about fertility and mortality and how these have changed over time [25].

Contributions: In this paper we investigate how clustering techniques for entity resolution [19, 26] can be employed for bundling birth records by the same mother, where temporal constraints can be incorporated to ensure no biologically impossible birth records by the same mother are linked together. We propose and evaluate a novel greedy temporal clustering approach, and compare it with a temporal variation of an existing clustering technique for entity resolution which has shown to work well in a previous study [26]. We conduct an empirical study on a data set from Scotland which has been extensively linked semi-manually by domain experts [25] providing us with ground truth data to calculate linkage quality. We show that temporal clustering techniques can outperform the linkage using non-temporal techniques in terms of linkage quality.

2 RELATED WORK

Record linkage has been an active field of research for over half a century in several research domains. Several recent books and surveys provide different perspectives of this area [6, 11, 18, 22].

Classification techniques for record linkage can be categorised into supervised and unsupervised techniques. Clustering techniques, which are unsupervised, view record linkage as the problem of how to identify all records that refer to the same entity and to group these records into the same cluster. Hassanzadeh et al. [19] presented a framework to comparatively evaluate different clustering techniques for record linkage. Saeedi et al. [26] recently proposed a framework to perform clustering for record linkage on a parallel platform using Apache Flink. Both these frameworks have implemented and evaluated several clustering approaches. In the evaluation by Saeedi et al. [26] star clustering (as described and modified in Section 3.3) was one of the overall best performing techniques compared to other clustering techniques. Neither of the two frameworks, however, has considered temporal constraints.

The linkage of historical data collections with the aim to produce large temporal linked data sets has recently received increased attention within the context of population reconstruction [3, 20]. Such linked population databases can be an exciting resource in areas such as health, history, and demography because these databases allow answering complex questions about temporal changes of a society that so far have been impossible to address. Most projects in historical record linkage are challenged by low data quality (due to scanning and transcription errors of handwritten forms), as well as a lack of ground truth data (which is difficult and expensive to obtain). Therefore, research in this area has concentrated on either exploiting the structure in such data sets (such as households and families) and developed group linkage methods [8, 13, 14, 24] or collective techniques [7]. Alternative approaches explore the use of limited ground truth data for evaluating linkage quality [1, 2].

3 TEMPORAL GRAPH LINKAGE

Our overall approach to temporal graph linkage consists of two major phases which we describe in detail in this section. First we generate an undirected graph based on pair-wise similarity calculations between individual records (birth certificates in our case). This is followed by a clustering of records (nodes) in this graph where we do take temporal constraints between records into account, as we describe in Section 3.2. In Sections 3.3 and 3.4 we discuss two temporal clustering approaches, the first based on the extension of an existing star-based clustering approach [19, 26], while the second approach generates clusters in a greedy temporal manner.

For notation we use bold letters for lists, sets and clusters (with type letters for numbers and text). Lists are shown with square and curly brackets, where lists have an order but sets do not.

Algorithm 1: Pair-wise similarity graph generation

| Input: |
| --- |
| 
| R: List of records to be linked |
| A: List of attributes from R to be compared |
| S: List of similarity functions to be applied on attributes from A |
| w: List of weights given to attribute similarities, with \(|w| = |S| |
| b, r: Number of bands and band size for min-hash based LSH blocking |
| \(s_{min}: Minimum similarity for record pairs to be added to the generated graph |
| G: Undirected pair-wise similarity graph |

| Output: |
| --- |
| G: Undirected pair-wise similarity graph |

```
1: \( V = \emptyset, E = \emptyset, G = (V, E) \) // Initialise empty graph
2: \( L = \text{MinHashSHIndexing}(R, b, r) \) // Generate Min-hash index
3: for \( i \in L \) do // Loop over all Min-hash blocks
4: \( s_{i,j} = \text{CompareRecords}(r_i, r_j, A, S, w) \) // Compute similarities
5: \( s_{i,j} = \text{Normalise}(s_{i,j}, w) \) // Normalise the similarity
6: if \( s_{i,j} \geq s_{min} \) then
7: \( \text{AddNodes}(G, V, (r_i, r_j)) \) // Create two new nodes in G
8: \( \text{AddEdge}(G, E, (r_i, r_j), s_{i,j}) \) // Create an edge in G
9: return G
```

3.1 Similarity Graph Generation

The steps involved in the pair-wise similarity calculation phase are outlined in Algorithm 1. The main input to the algorithm is a list of records, \( R \), which we aim to link and cluster (in our case we aim to determine which birth records are by the same mother). We assume each record has a unique numerical identifier, \( r.i.d \), and a time-stamp, \( r.t \), which in our case is the registration date of a birth certificate. We use the list \( A \) of attributes which we will compare between records using the list of similarity functions \( S \). These are approximate string matching functions such as Jaro-Winkler or edit distance [4], or functions specific to the content of an attribute like a numerical year difference function [6]. We also provide a list of
weights, \( w \), to be assigned to the calculated similarities. The value of the similarity \( s_a \) for attribute \( a \in A \) between two records \( r_i \) and \( r_j \) will be calculated as \( s_a(r_i, r_j) = S_a(r_i, r_j) \cdot w_a \), where \( w_a \) is the weight for attribute \( a \in A \) and \( S_a \) is the similarity function used on \( a \). The attributes and corresponding weight values we use in our experiments are shown in Table 1 in Section 4.

In order to prevent a full pair-wise comparison of each record in \( R \) with every other record in \( R \) (which has a complexity of \( O(|R|^2) \)), we employ min-hashing based on locality sensitive hashing (LSH) [21] which requires the two parameters \( b \) (the number of min-hash bands) and \( r \) (the band size). Furthermore, we provide a minimum similarity threshold \( s_{\text{min}} \), which determines which record pairs are to be included in the similarity graph \( G \) being generated.

Algorithm 1 starts by initialising an empty graph, followed by the generation of the min-hash index \( L \) which consists of blocks of records, \( I \). Each block \( I \in L \) contains one or more records from \( R \) that share the same min-hash value based on the content of the attribute values in \( A \). In lines 3 and 4 of the algorithm we loop over these blocks \( I \in L \) and generate all unique pairs of records in each block \( I \). In line 5 we compare the unique record pairs \( (r_i, r_j) \) from block \( I \) to calculate a vector of similarities \( s_{i,j} \). We then normalise these similarities into \( 0.0 \leq s_{i,j} \leq 1.0 \) in line 6. If this normalised similarity is at least the minimum similarity threshold \( s_{\text{min}} \) then in lines 8 and 9 we insert the two records \( r_i \) and \( r_j \) as nodes into the similarity graph \( G \), and we create an undirected edge between \( r_i \) and \( r_j \) where the edge attribute is the normalised similarity \( s_{i,j} \).

We finally in line 10 return the generate graph \( G \) which is used in the second phase of our approach to conduct clustering of the nodes in this graph. While in the pair-wise similarity calculation algorithm we do not consider any temporal constraints, we could add a temporal plausibility calculation step after line 6 and only insert a record pair into \( G \) if the pair is both similar enough and also temporally possible, as we describe next.

### 3.2 Modelling Temporal Constraints

Within the context of clustering birth records by the same mother, we model temporal constraints as a list \( T \) of time intervals where it is \textit{plausible} for a mother to have given birth to two babies. As illustrated in Figure 1, we need to consider issues such as data quality as well as multiple births (like twins and triplets, which potentially are born on two consecutive days). For each day difference \( \Delta t \) between two birth records (i.e. the number of days between two births) we calculate a \textit{plausibility} value \( p_{\Delta t} \) (with \( 0.0 \leq p_{\Delta t} \leq 1.0 \)), where \( p_{\Delta t} = 1.0 \) for day differences where two births by the same mother are possible, and \( p_{\Delta t} = 0.0 \) for day differences where it is biologically not possible for the same mother to have given birth to two babies. To account for wrongly recorded dates of birth we apply linear discounting of plausibility values, as shown in Figure 1.

![Temporal constraints as the plausibility for the same mother to be able to give birth to two children, where the horizontal axis shows the time difference (in days) and the vertical axis the plausibility \( p_{\Delta t} \) that two birth records are possible for a certain time difference. Due to errors in registration dates, for multiple births we allow for a few days difference for twins and triplets, and then have a plausible interval between birth from 9 months onwards up to 35 years. Two births by the same woman more than 40 years apart is deemed not to be plausible.](image)

We can use these temporal plausibility values to modify the similarity values between records by multiplying normalised record pair similarities \( s_{i,j} \), as calculated in Algorithm 1) with plausibility values, and then not considering record pairs in the graph \( G \) where their new modified similarity is below a given threshold.

We can apply these temporal constraints during the pair-wise similarity calculation step described in Section 3.1 (to only include record pairs into the graph \( G \) that are plausible from a temporal point of view). In the clustering step described in Sections 3.3 and 3.4 below, we also need to check for every pair of records in a cluster if they are temporally plausible. A cluster can contain pairs of records that are not in \( G \) because their similarity \( s_{i,j} \) is below the threshold \( s_{\text{min}} \), and these pairs also need to be plausible with regard to the given temporal constraints. Formally, for a given cluster \( c \), it must hold: \( \forall (r_i, r_j) \in c : p_{\Delta t} \geq p_{\text{min}} \), where \( p_{\text{min}} \) is a minimum plausibility threshold (similar to the similarity threshold \( s_{\text{min}} \) used in Algorithm 1). If this condition is not fulfilled for a record \( r_i \in c \) with all other records in \( c \), then \( r_i \) needs to be removed from \( c \).

While we currently set these temporal intervals of plausible births by the same mother based on discussions with domain experts, in the future we aim to learn temporal plausibility values from ground truth data. Besides temporal constraints between birth records by the same mother, in our application (where we aim to reconstuct populations by linking birth, death, marriage, and census records) there are other constraints we can consider. For example, a death of an individual can only occur on the same day or after the person’s birth. A marriage should only occur once a person has reached a minimum age. Similarly, records of the births by a mother can only occur once she has reached a certain minimum age, and before she has reached a certain maximum age.

### 3.3 Star Clustering

The second phase of our approach is to use a clustering algorithm to group all births by the same mother. We selected star clustering because this algorithm has shown to be one of the best performers in a previous evaluation study of clustering algorithms for entity resolution [26]. Our contribution to improve star clustering is two-fold: (a) we introduce temporal constraints as discussed in the previous section, and (b) we develop several methods for cluster centre selection and post-processing of overlapping clusters. Algorithm 2 outlines our modified star clustering algorithm.

Our modified algorithm is able to either consider temporal constraints (if the list of constraints \( T \) is provided) or ignore them (if \( T \) is empty) when generating clusters. The input to the algorithm are the pair-wise similarity graph, \( G \), as generated by Algorithm 1, and
the list T of temporal constraints. We also require the minimum plausibility \( p_{min} \) and minimum similarity \( s_{min} \) thresholds to decide if a node is added to a cluster, and the sorting and overlap resolving methods, \( m_{sort} \) and \( m_{reso} \), which we discuss in detail below.

The algorithm starts by initialising an empty list of clusters, \( C \), and an empty list \( U \) which will hold information about the nodes that are not yet assigned to clusters. Initially, all nodes in the similarity graph \( G \) are marked as unassigned by adding them to \( U \) in the loop starting in line 3. For each node \( v_i \in G \), using the function \( GetSimNeighbours() \) in line 4 we get the set of its neighbours \( n_i \in G \) that have an edge similarity of at least \( s_{min} \). We count the number of these neighbours as the degree \( d_i \) of node \( v_i \) in line 5, and also calculate the average similarity of all edges between \( v_i \) and its similar neighbours in \( n_i \). In line 7 we append a tuple containing \( v_i, d_i, n_i \), and \( a_i \) to the list of unassigned nodes \( U \).

Once tuples for all nodes in \( G \) have been added into \( U \), we sort \( U \) such that the best node to select as a cluster centre is at the beginning of this list. We investigate three different methods of how to order nodes based on the sorting method provided in \( m_{sort} \):

- **Avr-all**: We average the similarities between the node \( v_j \) and all the nodes in a cluster it is connected to in the similarity graph \( G \) by dividing this similarity sum by \( n - 1 \) where \( n \) is the number of nodes in the cluster (including \( v_j \)), i.e. we do take nodes in a cluster which are not connected to \( v_j \) in \( G \) into account.

- **Avr-high**: We calculate the average similarity between the node \( v_j \) and all the nodes in a cluster which are not connected to \( v_j \) in \( G \).

- **Edge-ratio**: In this method we count the number of edges between \( v_j \) and nodes in a cluster that have a similarity of at least \( s_{min} \) and divide this number by \( n - 1 \) where \( n \) is the number of nodes in the cluster (including \( v_j \)).

For each node \( v_j \in v_{rep} \), we assign it to the cluster with the highest value according to the selected method to resolve overlaps. For all three methods, if for a given node \( v_j \) two or more clusters have the same calculated score then we assign \( v_j \) to the cluster where \( v_j \) has the highest number of similar edges to. At the end of this process, the final list of clusters \( C \) contains no overlapping clusters.

### 3.4 Greedy Temporal Clustering

The second temporal clustering approach is based on the idea of iteratively adding nodes to clusters using a greedy selection method, as illustrated in Figure 2. We initially create one cluster per record, and insert these singleton clusters into a priority queue that is sorted according to time-stamps (i.e. the dates of birth registrations in our case) with the smallest time-stamp first. We then process the earliest cluster first, and aim to expand this cluster with a new
Temporal graph-based clustering for historical record linkage

Algorithm 3: Greedy temporal clustering

Input:
- \( G \): Undirected pair-wise similarity graph
- \( T \): List of temporal constraints (as discussed in Section 3.2)
- \( p_{\text{min}} \): Minimum plausibility for record pairs to be considered
- \( m_{\text{sel}} \): Method on how to select the next node to add to a cluster

Output:
- \( C \): Final list of clusters

1: \( G_D = \text{GenerateTempDirGraph}(G) \)  // A temporal directed graph
2: \( C = \{ \} \)  // Initialise an empty list of clusters
3: \( Q = \{ \} \)  // Initialise an empty priority queue
4: for \( v \in G_D, V_d \) do  // Loop over all nodes in \( G_D \)
5:  if \( (|v.\text{in}|) = 0 \land (|v.\text{out}|) = 0 \) then:  // A singleton
6:     \( C.\text{add}(\{v\}) \)  // Add to the final list of clusters
7: else:
8:     \( Q.\text{add}((v, t, \{v\})) \)  // Add node with its time-stamp to queue \( Q \)
9: Sort(Q)  // Sort queue according to time-stamps (earliest first)
10: while \( Q \neq \{ \} \) do  // Loop over temporal clusters until \( Q \) is empty
11:     \( (t, e_{\text{tmp}}) = Q.\text{pop}() \)  // Get first cluster tuple in \( Q \)
12:     \( o = \cup v_j, o \text{.out}(), v_j \in e_{\text{tmp}} \)  // Set of all outgoing nodes
13: if \( o = \emptyset \) do  // No outgoing nodes found in \( e_{\text{tmp}} \)
14:     \( C.\text{add}(e_{\text{tmp}}) \)  // Add \( e_{\text{tmp}} \) to the final list of clusters
15: else:
16:     if \( m_{\text{sel}} = \text{Next do} \) do  // Select node with smallest time-stamp
17:         \( v_n = v_j \in o : \text{argmin}(v_j.\text{t} : v_j \in o) \)
18:     if \( m_{\text{sel}} = \text{Max-sim do} \) do  // Select node with the highest similarity
19:         \( v_n = v_j \in o : \text{argmax}(v_j.\text{sim} : v_j \in e_{\text{tmp}}, v_j \in o) \)
20:     if \( m_{\text{sel}} = \text{Avr-sim do} \) do  // Select node with highest average similarity
21:         \( v_n = v_j \in o : \text{argmax}(|\sum_j|v_j.\text{sim} : v_j \in e_{\text{tmp}}, v_j \in o)\}
22: \( \Delta t = \text{CheckTempConstr}(v_n.\text{t}, e_{\text{tmp}}, T) \)  // Temporal plausibility
23: if \( \Delta t > p_{\text{min}} \) do
24:     \( Q.\text{add}((v_n.\text{t}, e_{\text{tmp}} \cup \{v_n\})) \)  // Add expanded \( e_{\text{tmp}} \) to \( Q \)
25: Sort(Q)  // Sort queue according to time-stamps (earliest first)
26: else:
27:     \( C.\text{add}(e_{\text{tmp}}) \)  // Add \( e_{\text{tmp}} \) to the list of final clusters
28: return \( C \)

Figure 2: Example of the greedy temporal linkage approach described in Section 3.4, showing nodes (records) and edges (similarities) from the directed similarity graph \( G_D \). Records \( r_1 \) to \( r_3 \) show an existing cluster, and the question now is which best future record (from \( r_4, r_5, \) and \( r_6 \)) is to be added to the cluster next. We consider three selection methods: (a) the earliest next possible (according to temporal constraints) record in the graph \( G \) (in this example \( r_4 \)), (b) the future record with the highest maximum similarity \( (r_5) \), or (c) the future record with the highest average similarity \( (r_6) \).

A record that is in the future (of the latest record in the cluster), as Figure 2 shows. In this greedy approach the question is how to select the best future node (record) to add to a cluster. We implement (and evaluate in Section 4) three different such selection methods:

- **Next**: Select the temporal next record (with the smallest time-stamp) that is connected via an edge in the graph \( G \) to any record in the cluster. This method does neither consider the similarities between nodes (besides the edges in \( G \)) nor their connectivities and serves as a greedy baseline.

- **Max-sim**: Select the record in the future that is connected via an edge in the graph \( G \) to any record in the cluster and that has the highest similarity \( s_{ij} \) with any record in the cluster. This method generates clusters where nodes are connected via edges of high similarities, however, these clusters might not be dense.

- **Avr-sim**: Select the record in the future that is connected via an edge in the graph \( G \) to one or more records in the cluster and that has the highest average similarity over these edges. This method generates dense clusters with high similarity edges.

As with star clustering, we can consider temporal constraints when selecting the next record to be added into a cluster, or we can ignore any temporal constraints. Algorithm 3 outlines the steps involved in this temporal greedy clustering approach.

The main input to the algorithm are the pair-wise similarity graph, \( G \), and a list of temporal constraints, \( T \), as discussed in Section 3.2. We also input a minimum plausibility threshold \( p_{\text{min}} \) which is used to consider which record pairs are to be added into clusters based on their temporal constraints, and the selection method \( m_{\text{sel}} \) which determines which nodes (records) to add into a cluster.

We first (in line 1) convert the undirected similarity graph \( G \) into a directed graph where each node (birth record) has an outgoing edge to any future node, as shown in Figure 2. The function GenerateTempDirGraph() generates a directed graph \( G_D \) by considering the time differences between the pairs of nodes in \( G \), such that \( \forall (v_i, v_j) \in G_D, E : v_j.\text{t} \geq v_i.\text{t} \). In line 4, the algorithm then loops over each node \( v \in G_D \) and adds \( v \) to the final list of clusters \( C \) if \( v \) does not have any incoming or outgoing edges to other nodes (lines 5 and 6), i.e. the node is a singleton. Otherwise, a new cluster is created containing only node \( v \), and this cluster is added together with its time-stamp, \( v.\text{t} \), as a tuple into the priority queue \( Q \) for further processing (line 8).

In line 9 we sort \( Q \) according to the time-stamps of each cluster such that the cluster with the smallest time-stamp is at the beginning of the queue. The main loop of the algorithm starts in line 10 where in each iteration we retrieve the cluster \( e_{\text{tmp}} \) with the earliest time-stamp (line 11). We then find for each node \( v_c \in e_{\text{tmp}} \) all its outgoing nodes in \( G_D \), and in line 12 we combine these into the set \( o \) of all outgoing nodes for \( e_{\text{tmp}} \). If \( o \) is empty for the current cluster \( e_{\text{tmp}} \) then \( e_{\text{tmp}} \) is added to the final list of clusters \( C \) in line 14 because it cannot be expanded further.

On the other hand, if there are outgoing nodes (i.e. \( o \) is not empty), then based on the selection method \( m_{\text{sel}} \), as explained above, the algorithm selects the next best node, \( v_n \), to be added into the current cluster \( e_{\text{tmp}} \) in lines 16 to 21. Using the function CheckTempConstr() in line 22 we then check the temporal plausibility \( p_{\Delta t} \) between node \( v_n \) and all nodes in \( e_{\text{tmp}} \) based on the list of temporal constraints \( T \) (if this list is empty, i.e. no temporal constraints are given, then we set \( p_{\Delta t} = 1 \)). If the calculated \( p_{\Delta t} \) is at least \( p_{\text{min}} \) (i.e. \( v_n \) is temporally plausible with all other nodes in \( e_{\text{tmp}} \)), then \( v_n \) is added to the current cluster \( e_{\text{tmp}} \) and the expanded cluster is added as a new tuple into \( Q \) with \( v_n.\text{t} \) as the tuple’s time-stamp (line 24). \( Q \) is sorted again in line 25 to ensure the cluster with the smallest time-stamp (of its temporarily last record) is selected in the next iteration (line 25). If \( v_n \) is not temporally plausible with at least one node in \( e_{\text{tmp}} \) then \( e_{\text{tmp}} \) is added to the final list of clusters \( C \) in line 27 because it cannot be expanded further.
We evaluate our proposed temporal clustering approaches using Table 2: The five most frequent values and their correspond-
tagged elements
these non-matches will be clustered incorrectly. This is caused by the greedy nature of both clustering algorithms, where after an incorrect node is selected as the next best node the actual true matches are never offered a chance to be clustered together. This behaviour is mostly accentuated when only parent names are used to calculate the similarities between certificates. This is because the distribution of parent names is the most skewed, as Figure 3 shows.

As both Figures 4 and 5 show, when temporal constraints are included in the clustering phase then precision generally increases considerably while recall only decreases little. The overall best performing approach was using unweighted similarities of only parent names. The overall highest precision and recall results without temporal constraints were 0.877 and 0.897, while when applying temporal constraints they were 0.925 and 0.888, respectively.

The result plots also show that overall star clustering achieves better results with regard to recall than the temporal greedy technique, however the similarity based selection methods for temporal greedy clustering achieve overall higher minimum precision results.

5 CONCLUSIONS AND FUTURE WORK

In this work-in-progress paper we have developed and evaluated two clustering approaches for linking birth certificates in the context of historical record linkage. Both algorithms are based on a graph that represents the similarities calculated between individual birth certificates. We have evaluated six approaches how this graph is generated based on comparing different attribute combinations in a weighted or unweighted fashion, and how the characteristics of this graph affect the final clustering outcomes. Our experimental evaluation on a real Scottish data set have shown that incorporating temporal constraints (when a woman can give birth or not) can improve the quality of the final linked data set.

As future work we aim to improve our proposed greedy temporal clustering algorithm as well as temporal star clustering to obtain better linkage results. We aim to investigate why certain birth certificates are not linked (missed true matches, lowering recall) while others are falsely linked (wrong matches, lowering precision). We then aim to expand our graph-based clustering techniques to also incorporate links across birth, marriage, death, and census certificates by generating a single large similarity graph where nodes represent certificates and edges similarities between them, and where edges can be of different types [7]. Such a graph will not only allow temporal constraints to be considered but also gender and role-type specific constraints [7, 8]. We plan to model temporal aspects of how the records about a certain individual will occur in historical population databases. Our ultimate aim is to develop unsupervised techniques for the accurate and efficient linkage of large and complex historical population databases in order to provide researchers in areas such as health and the social sciences with high quality longitudinal data sets.

ACKNOWLEDGEMENTS

This work was supported by ESRC grants ES/K00574X/2 Digitising Scotland and ES/L007487/1 Administrative Data Research Centre – Scotland. We like to thank Alice Reid of the University of Cambridge and her colleagues Ros Davies and Eilidh Garrett for their work on the Isle of Skye database, and their helpful advice on historical Scottish demography. This work was partially funded by the Australian Research Council under DP130101801.

REFERENCES

[1] Luiza Antonie, Kris Inwood, Daniel J. Lizotte, and J. Andrew Ross. 2014. Tracking people over time in 19th century Canada for longitudinal analysis. Machine Learning 95 (2014), 129–146.
[2] Martha Bailey, Connor Cole, Morgan Henderson, and Catherine Massey. 2017. How Well Do Automated Methods Perform in Historical Samples? Evidence from New Ground Truth. Technical Report. National Bureau of Economic Research.
[3] Gerrit Bloothoof, Peter Christen, Kees Mandemakers, and Marijn Schraagen. 2015. Population Reconstruction. Springer.
[4] Peter Christen. 2006. A Comparison of Personal Name Matching: Techniques and Practical Issues. In ICDM Workshop on Mining Complex Data. Hong Kong.
[5] Peter Christen. 2008. Febrl: An open source data cleaning, deduplication and record linkage system with a graphical user interface. In ACM SIGKDD.
[6] Peter Christen. 2012. Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer.
[7] Peter Christen. 2016. Application of Advanced Record Linkage Techniques for Complex Population Reconstruction. arXiv preprint arXiv:1612.04286 (2016).
[8] Victor Christen, Anika Grof, Jeffrey Fisher, Qing Wang, Peter Christen, and Erhard Rahm. 2017. Temporal group linkage and evolution analysis for census data. In EDBT, Venice, Italy, 620–631.
[9] Chris Dibben, Lee Williamson, and Zengyi Huang. 2012. Digitising Scotland. http://gtr.rcuk.ac.uk/projects/ref=ES/K00574X/2
[10] Lisa Y. Dillon. 1996. Integrating nineteenth-century Canadian and American census data sets. Computers and the Humanities 30, 5 (1996), 381–392.
[11] Xin Luna Dong and Divesh Srivastava. 2015. Big data integration. Synthesis Lectures on Data Management 7, 1 (2015), 1–198.
[12] Xin Luna Dong and Wang-Chiew Tan. 2015. A time machine for information: Looking back to look forward. Proceedings of the VLDB Endowment 8, 12 (2015).
[13] Zhichun Fu, Mac Boot, Peter Christen, and Jun Zhou. 2014. Automatic Record Linkage of Individuals and Households in Historical Census Data. International Journal of Humanities and Arts Computing (2014).
[14] Zhichun Fu, Peter Christen, and Jun Zhou. 2014. A Graph Matching Method for Historical Census Household Linkage. In PAKDD. Tainan, Taiwan.
[15] Zhichun Fu, Jun Zhou, Peter Christen, and Mac Boot. 2012. Multiple Instance Learning for Group Record Linkage. In PAKDD. Kuala Lumpur.
[16] Emily Grundy and Cecilia Tomassini. 2005. Fertility history and health in later life: a record linkage study in England and Wales. Social Science and Medicine 61, 1 (2005), 217–228.
[17] David Hand and Peter Christen. 2018. A note on using the F-measure for evaluating record linkage algorithms. Statistics and Computing 28, 3 (2018), 539–547.
Figure 4: Precision-recall results for the temporal star clustering approach described in Section 3.3 using the three discussed overlap resolving methods, and without (top row) and with (bottom row) temporal constraints. Each plot shows results for the six similarity graphs described in Section 4 (with / without weighted similarities and different attributes compared).

Figure 5: Precision-recall results for the greedy temporal clustering approach described in Section 3.4 using the three discussed selection methods, and without (top row) and with (bottom row) temporal constraints.

[18] Katie Harron, Harvey Goldstein, and Chris Dibben. 2015. Methodological Developments in Data Linkage. John Wiley & Sons.
[19] Oktie Hassanazadeh, Fei Chiang, Hyun Chul Lee, and Renée J Miller. 2009. Framework for evaluating clustering algorithms in duplicate detection. Proceedings of the VLDB Endowment 2, 1 (2009), 1282–1293.
[20] Hye-Chung Kum, Ashok Krishnamurthy, Ashwin Machanavajjhala, and Stanley C. Ahalt. 2014. Social Genome: Putting Big Data to Work for Population Informatics. IEEE Computer 47, 1 (2014).
[21] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. Mining of Massive Datasets. Cambridge University Press.
[22] Felix Naumann and Melanie Herschel. 2010. An Introduction to Duplicate Detection. Morgan and Claypool Publishers.
[23] Gill Newton. 2011. Recent developments in making family reconstitutions. Local Population Studies 87, 1 (2011), 84–89.
[24] Byung-Won On, Nick Koudas, Dongwon Lee, and Divesh Srivastava. 2007. Group Linkage. In IEEE ICDE. Istanbul.
[25] Alice Reid, Ros Davies, and Eilidh Garrett. 2002. Nineteenth-Century Scottish Demography from Linked Censuses and Civil Registers. History and Computing 14, 1-2 (2002).
[26] Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2017. Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In Advances in Databases and Information Systems. Springer, 278–293.
[27] Edward Wrigley and Roger Schofield. 1973. Nominal record linkage by computer and the logic of family reconstitution. Identifying people in the past (1973), 64–101.