Unsupervised Graph-based Entity Resolution for Accurate and Efficient Family Pedigree Search

Nishadi Kirielle, Charini Nanayakkara, Peter Christen
nishadi.kirielle@anu.edu.au
The Australian National University
Canberra, ACT 2600, Australia

Chris Dibben, Lee Williamson, Eilidh Garrett
chris.dibben@ed.ac.uk
University of Edinburgh
Edinburgh EH8 9XP, UK

Claire Manson
claire.manson@phs.scot
Public Health Scotland
Edinburgh EH12 9EB, UK

ABSTRACT

Tremendous progress has been made in medical research in recent years to detect, treat, and prevent a variety of commonly occurring cancers. There is now a shift towards better identifying and understanding cancers and other diseases that occur very rarely in a population. One approach to investigate such rare diseases is to look at a patient's family history through genealogical as well as genetic research. The Genetics Genealogy Team, part of Public Health Scotland, has been providing a unique service exploring family pedigrees (family trees) using birth, death, and marriage certificates going back to 1855. Thus far searches have been manually exploring each digitised record individually which can be time consuming and labour intensive. Here we present a prototype application for automated family pedigree search that is based on unsupervised graph-based entity resolution techniques combined with approximate query matching and ranking methods to efficiently and accurately extract and visualise family pedigrees from searched birth or death certificates. We describe the steps of our application, and how we anonymise sensitive personal data to allow it being used for training and educational purposes. Our prototype application and an anonymised data set are available at: https://dmm.anu.edu.au/SNAPS/

1 INTRODUCTION

Entity resolution (ER), also known as data or record linkage [11], is the process of identifying and linking records that refer to the same entities within one or across several databases. If applied on a single database the task is known as duplicate detection [46]. ER has applications in many domains, ranging from health and social science research to national censuses, crime and fraud detection, public health surveillance, and national security [14, 39, 53].

The main challenge of ER is that generally no unique entity identifiers (such as social security or patient numbers) are available in the databases to be linked. Therefore, attributes that partially identify entities, known as quasi-identifiers (QIDs) [14], need to be compared to link records with similar QID values that provide evidence that two records may refer to the same entity. For databases containing records about people, the QIDs used for ER commonly include personal names, addresses, dates of birth, and so on [52]. However, it is challenging to resolve such databases because QID values can change over time, and typographical errors as well as spelling variations occur commonly with personal data [11].

To make ER scalable to large databases, blocking, indexing, and filtering techniques [54] can be applied where similar records are grouped into blocks and only records in the same block are compared. The compared record pairs are then classified into matches (two records referring to the same entity) and non-matches (two records referring to two different entities) using a decision model [11]. Such decision models generally consider the similarities calculated between QID values, and possibly also the relationship information available between records [6, 19]. These decision models can be as simple as a single similarity threshold, be based on probabilistic models [23], or they can be sophisticated machine learning classifiers such as deep learning or graph-based clustering approaches [4, 31, 44].

The majority of ER approaches assume the batch linkage of static databases [11]. Only limited research has explored how query records containing details of an entity can be linked in (near) real-time with a database that consists of records about known entities [7, 13, 56]. In such a context, the challenges include fast similarity calculations between a query record and selected candidate records in a database, as well as the accurate ranking of the most likely matching candidate records (rather than classifying record pairs into matches and non-matches). In this paper we describe such a query-based ER application aimed at generating family pedigrees (family trees) for patients who have familial cancer or other inherited genetic conditions [8].

The Genetics Genealogy Team of Public Health Scotland provides a unique service for patients with a potential familial cancer or other inherited genetic condition by supplying accurate and comprehensive family history research for patients who have been referred to a clinical genetics service[1]. The determination and management of risk is dependent on the quality of the family information genetics clinicians have to work with. The team explores the statutory records (births, death, and marriage certificates) held by the National Records of Scotland since 1855, with the aim to build a family pedigree for a patient which they combine with data from the NHS (National Health Service) Scotland registers. This will provide high quality data to enable geneticists to conduct a more complete risk assessment for a patient and their families [8]. Risk assessment for patients may result in gene testing, prophylactic surgery, screening appropriate to the determined risk, or reassurance and discharge from the care of clinical genetics services, using the UK's NHS resources in the most appropriate manner.

Currently the Genetics Genealogy Team query the Scottish statutory records via the Scotland's People web interface[2], retrieve the corresponding certificates and manually compile a family pedigree by linking individuals from across certificates [8, 34]. This is a time consuming and labour intensive process.

Our work is part of the Digitising Scotland (DS) project[3], which has recently completed the transcription of all Scottish birth, death, and marriage certificates from 1855 to 1973, in total around

[1] See: https://www.isdscotland.org/Health-Topics/Cancer/Genetics-Genealogy/
[2] See: https://www.scotlandspeople.gov.uk/
[3] See: https://digitisingscotland.ac.uk/

24 million certificates. Once linked, the complete DS database will allow the investigation of issues such as inter-generational social mobility or the changing demography across the period of industrialisation, among many other research questions. Note that no ground truth links between certificates are available for this database. Compared to existing linked multi-generational population databases that cover up-to four generations [28], the DS database will provide detailed information for up-to six generations, assuming a generation length of around 30 years.

Our aim is to employ our application on this DS database of the full population of Scotland, and provide an efficient and accurate service for the Genetics Genealogy Team to investigate the inheritability and genetic patterns of diseases for patients with familial cancer or other inherited genetic conditions.

In our work, in an offline phase we employ unsupervised graph-based ER techniques to first generate an entity graph from which we then create a pedigree graph. The main difference between the entity graph and the pedigree graph is that the latter includes relationships between different entities over multiple generations. In the offline phase we address several challenges in ER that are fundamental to linking records about people: changing QID (attribute) values, different relationships, ambiguities, partially matching groups, and wrong links. We elaborate further on these challenges in Section 2, and in Section 4 we highlight the novel approaches we develop to solve these challenges.

In the context of our application, we can only apply unsupervised learning techniques due to the lack of ground truth data. Manually obtaining true matches and non-matches for training of supervised techniques such as recently proposed deep learning methods [36, 43, 44, 65] would be a time consuming and expensive process that would require extensive domain knowledge about historical Scottish vital records. This process of manually generating ground truth data is known to often result in limited, biased, and incomplete ground truth data [3, 58].

In the online querying phase, for a given query record we first identify the most similar individuals in the pedigree graph generated from birth, death, and marriage certificates in the offline phase. For an individual selected by the user we then build their family pedigree by linking the record of this individual with those of their family members across several generations.

As we outline in Figure 1, our application to efficiently and accurately generate family pedigrees from search queries (referred to as *SNAPS* which stands for *ScotlaNd fAmily Pedigree Search*) consists of five steps. In Sections 4 to 6 we describe the main offline steps (to link records, generate the pedigree graph, and build the keyword and string similarity indices). In Sections 7 and 8 we then discuss the processing of queries, as well as the generation and visualisation of family pedigrees (trees). The linking of records consists of several steps, where dependencies, temporal constraints, missing and erroneous values, as well as ambiguities are considered in an unsupervised iterative process. The query processing facilitates both exact and approximate string matching [11, 47] to account for the uncertainty of users with regard to spelling variations of names and locations [11, 37].

Privacy and confidentiality restrictions have so far prevented us from remotely accessing the full DS database. We therefore evaluate our prototype on two Scottish data sets from the Isle of Skye and Kilmarnock containing birth, death, and marriage certificates from 1861 to 1901 [58]. These data sets have been extensively curated manually by domain experts, and partial ground truth data are available which allows us to also evaluate the linkage quality of our prototype.

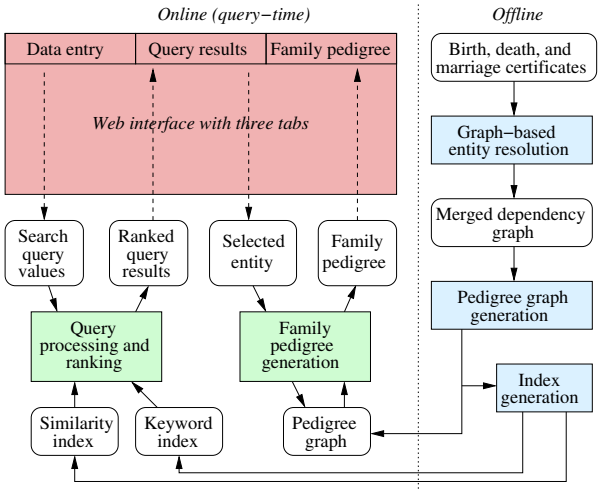

Figure 1: *SNAPS* architecture where the offline steps, to generate the entity and pedigree graphs, and the keyword and similarity indices, are shown in blue (right side), while the online query processing and ranking, and the family pedigree generation steps, are shown in green (left side).

An important aspect to facilitate testing, training, as well as public demonstrations of an application such as ours is to allow open access. Because *SNAPS* will be applied on a large database that contains real birth, death, and marriage certificates of the Scottish population, in Section 9 we describe an approach to anonymise a graph that represents such a population.

2 DATA CHARACTERISTICS AND ENTITY RESOLUTION CHALLENGES

An important aspect to discuss in ER for family pedigree search is the characteristics of the data sets we use in our application. In this section we discuss the characteristics of person data using examples from the Isle of Skye (IOS) and Kilmarnock (KIL) data sets and the DS database. The challenges for resolving such data are however generic to many demographic data sets including statutory records and census snapshots.

The IOS and KIL data sets are both samples of the full DS database and contain birth, death, and marriage certificates of the populations of the Isle of Skye and the town of Kilmarnock, respectively, over the period from 1861 to 1901. Both have been extensively curated and semi-manually linked (using database queries to extract possible links) by experts in the domain of linking such historical data [58]. Their semi-automatic approach was heavily biased towards certain types of links, such as links between birth parents, as their interests were in identifying children by the same mother to facilitate the analysis of child mortality [59]. This resulted in incomplete and biased ground truth data, an issue that is also known with other (historical) data sets about people [3]. However, this manual curation and linkage is not scalable to the full DS database which contains around 24 million certificates, requiring us to employ unsupervised methods for linking the DS database.

Existing graph-based ER techniques [6, 19, 35, 40] have difficulties achieving high linkage quality on person data of changing QID values, different relationships found at different points in time, the ambiguity of QID values, partially matching record groups, and wrong links. Temporal record linkage techniques [33,

Table 1: Missing value counts and QID value frequencies (minimum, average, and maximum) of deceased people in the IOS and KIL data sets, and the full DS database.

Data set (Entities)	QID attribute	Missing values	QID value frequencies Min	Avr	Max
IOS	First name	426	1	21.8	1,089
12,285	Surname	3	1	27.7	1,027
	Address	143	1	12.3	227
	Occupation	7,018	1	9.0	730
KIL	First name	241	1	5.8	1,837
23,715	Surname	3	1	9.0	520
	Address	5,873	1	13.9	806
	Occupation	16,846	1	4.4	425
DS	First name	56,387	1	13.8	520,685
8,289,592	Surname	7,299	1	50.9	112,673
	Address	10,730	1	3.3	71,456
	Occupation	4,795,995	1	9.0	78,823

41], on the other hand, only consider QID values changing over time but not these other challenges associated with linking person data. We now discuss these challenges in more detail.

When resolving entities in typical ER scenarios, such as linking publication records from two bibliographic databases, the QID values of a publication (like its title, authors, venue, and year) are static but potentially have some spelling variations or abbreviations [6, 19, 35, 40]. However, when linking records about people their QID values tend to change over time. For example, people can move around and therefore their addresses change. Similarly, surnames (mostly of women) can change after marriage. Along with missing values, as we show in Table 1, such changing QID values make ER of person data challenging.

Another challenge specific to person data is the different roles and relationships that we encounter in the different types of certificates, and where as a result an individual can have different roles and relationships at different points in time. For example, assume we want to link a mother in a birth certificate to her own birth certificate (where she has the role of the baby). In the initial birth certificate, the mother has a relationship to her spouse and baby, but in her own birth certificate she has a relationship to her parents. Therefore, these two certificates have no relationships in common. The different relationships and roles encountered in person data and how to employ them in an unsupervised way is a challenge that we address with our techniques.

Disambiguation of different entities having common QID values is another challenge in resolving person data. From Table 1 we can see that person records in the IOS and KIL data sets and the DS database have a high average frequency in most shown QID attributes. To elaborate more on the ambiguity of QID values found in person records, in Figure 2 we show the frequency distributions of the 100 most common values of selected QIDs for deceased persons in the IOS and KIL data sets. These distributions are very skewed, where the most common first name and surname each occur in over 8% of all records in the IOS data set.

Finally, we discuss the partial match group problem. When, for example, we have two birth certificates of siblings, we should link their parents since they represent the same two individuals. However, we should not link the two siblings as they refer to two different individuals. This is challenging because most of the QID values of these siblings will likely be the same, such as their surname and street address. We consider this as the partial match group problem because only some records in a group should be linked and resolved as referring to the same entity.

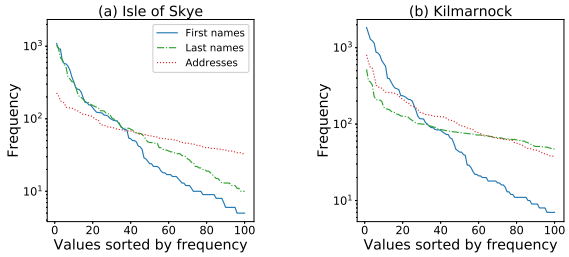

Figure 2: Frequency distribution of the 100 most common first names, surnames, and addresses of deceased people in the Isle of Skye (IOS) and Kilmarnock (KIL) data sets.

3 PROBLEM DEFINITION AND APPLICATION OVERVIEW

We now define the problem of our graph-based ER application for family pedigree search. Let $\mathbf{R}$ be a set of records from birth (B), death (D), and marriage (M) certificates, where each record $r \in \mathbf{R}$ has a role, such as a baby (Bb) and its mother (Bm) and father (Bf) on a birth certificate, or a deceased person (Dd) and their mother (Dm) and father (Df), and possibly their spouse (Ds), on a death certificate. We refer to a cluster of records $\mathbf{R}_o \subset \mathbf{R}$ as an entity $o \in \mathbf{O}$ that represents a real-world individual.

Problem definition: Given a set of records, $\mathbf{R}$, and a query record, q, containing a mandatory first and surname, and optionally a gender, year, and a location, the family pedigree search problem is to find the set $\mathbf{O}_q \subset \mathbf{O}$ of entities in $\mathbf{O}$ that have the highest similarities with q along with their family pedigrees $\mathbf{P}_q$.

Figure 1 shows the architecture of *SNAPS* which has an online and an offline component, consisting of the following main steps which we describe in detail in Sections 4 to 8.

(1) Graph-based entity resolution: This key step in *SNAPS* is aimed at linking records $r \in \mathbf{R}$, where each r corresponds to a single occurrence of an individual on a certificate, such as a baby, a bride, or a deceased person. We aim to associate the corresponding entity, $o \in \mathbf{O}$ (that represents a real-world individual) to the correct set of linked records (a record cluster) $\mathbf{R}_o \subset \mathbf{R}$. For this, we represent $\mathbf{R}$ as a dependency graph [19], a directed graph $\mathbf{G}_D = (\mathbf{N}, \mathbf{E})$, that consists of a set of nodes, $\mathbf{N} = \mathbf{N}_A \cup \mathbf{N}_R$, that represent either pairs of QID (attribute) values (atomic nodes, $\mathbf{N}_A$), or pairs of records that possibly refer to the same entity (relational nodes, $\mathbf{N}_R$); and a set of edges, $\mathbf{E}$, that represent the relationships between these nodes where an edge direction indicates a dependency. To link records and associate the linked record clusters to the corresponding entities, we iteratively merge the nodes in the dependency graph $\mathbf{G}_D$.

(2) Pedigree graph generation: We then generate a pedigree graph, $\mathbf{G}_P$, to represent the relationships between the entities associated with the record clusters in $\mathbf{G}_D$. The nodes in this graph are the entities, $o \in \mathbf{O}$, while the edges represent relationships between entities, such as *motherOf* (*Mof*), *fatherOf* (*Fof*), *spouseOf* (*Sof*), and *childOf* (*Cof*).

(3) Index generation: To improve the efficiency of query processing and ranking, we generate two index structures. The keyword index, $\mathbf{K}$, maps QID values, such as first names and surnames, to entities $o \in \mathbf{O}$ to facilitate fast query processing.

To speed-up the matching of QID values in a query record q with values in the keyword index $\mathbf{K}$, and to allow for efficient approximate string matching, we furthermore pre-calculate and store the similarities between pairs of strings [13].

(4) Query processing and ranking: In the online phase, a query record q is first matched with QID values in the keyword index, $\mathbf{K}$. For each entity $o \in \mathbf{O}$ that has a string match on at least one of first name or surname (and possible other matching QID values), we then calculate an overall match score and present the top ranked entities in $\mathbf{O}$ to the user.

(5) Family pedigree extraction and visualisation: When a user selects an individual through the web interface, this step extracts the pedigree, p, of this entity from $\mathbf{G}_P$, and generates a family tree visualisation for that pedigree.

4 GRAPH-BASED ENTITY RESOLUTION

The core of *SNAPS* is the unsupervised graph-based ER step that will link records appearing on birth, death, and marriage certificates. We cannot employ supervised learning techniques due to the lack of ground truth data available, and the costs and challenges involved in manually obtaining true matches and non-matches in the context of (historical) personal data [58]. Supervised ER techniques commonly also require a large amount of training data to obtain good results [4, 20]. We therefore propose an unsupervised graph-based ER solution that addresses the challenges we discussed in Section 2, such as changing QID values, different relationships encountered at different points in time, ambiguities of QID values, and the partial match group problem.

We now present the main steps of *SNAPS*: dependency graph generation, bootstrapping, and merging.

4.1 Dependency Graph Generation

Because we are modelling records in a dependency graph, $\mathbf{G}_D$, as defined in Section 3, if all possible record pairs and QID value pairs are added into this graph then it can get very large. We therefore conduct blocking to reduce the comparison space and obtain record pairs that are potential matches. We employ a locality sensitive hashing based blocking technique in *SNAPS* that maps similar QID value pairs to the same hash value to group likely matches [54]. Then we consider record pairs in each block to generate $\mathbf{G}_D$ in two phases.

First, only QID value pairs that have a (string) similarity [11] of at least a threshold t_a are added to $\mathbf{G}_D$ along with their similarities as atomic nodes, $\mathbf{N}_A$. The similarities between QID values in atomic nodes are assumed to be between 0 (completely different values) and 1 (same values), where we calculate these similarities using approximate string comparison functions, such as Jaro-Winkler or edit distance [11], as appropriate to the values in a QID attribute. Next, relational nodes, $\mathbf{N}_R$ are added based on two filtering steps. We first filter record pairs of impossible role types, such as pairs with different genders, and then we filter pairs by constraints as we discuss further in Section 4.2.2.

For example, in Figure 3, r_1, r_2, and r_3 are a Bb, Bm, and Bf extracted from a birth certificate, while r_4, r_5, r_6, and r_7 are a Dd, Dm, Df, and Ds extracted from a death certificate. These records are represented in a dependency graph, $\mathbf{G}_D$. One possibility is that the birth baby r_1 dies and becomes the deceased person r_4 in the death certificate. In that scenario, the nodes (r_1, r_4), (r_2, r_5), and (r_3, r_6) will be added to the graph $\mathbf{G}_D$ as relational nodes with relationship edges between these nodes. Likewise, we add all other possible nodes and their relationships to $\mathbf{G}_D$.

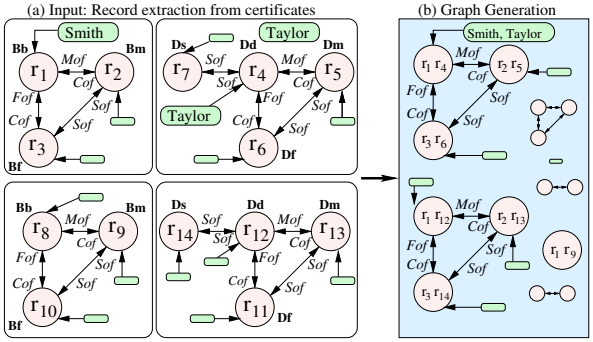

Figure 3: Dependency graph generation, where we show atomic nodes in green and relational nodes in pink. The input are the records extracted from two birth and two death certificates, showing the relationships *motherOf* (*Mof*), *fatherOf* (*Fof*), *spouseOf* (*Sof*), and *childOf* (*Cof*).

4.2 Bootstrapping and Merging

Bootstrapping and merging are the key steps where we link records iteratively. Prior to describing these steps, we outline the key techniques that we employ in these steps below: global propagation of QID (attribute) values and constraints, leveraging ambiguity of QID values, adaptive leveraging of relationship structure, and dynamic refining of record clusters.

4.2.1 Global Propagation of QID Values (PROP-A). As we discussed in Section 2, the first challenge is changing QID values where values such as names and addresses can change over time. When an attribute value changes over time, this change makes it difficult to decide if two records refer to the same entity. To solve this problem, we check the QID values of the entities associated with records to make those link decision. We consider this propagation of QID values in the ER process as positive evidence.

Assume the records r_1, r_4, and r_9 in Figure 3 (a) refer to the same entity, o_1, and this entity has changed her surname after marriage. Therefore, her baby record $r_1(Bb)$ has her maiden surname, *Smith*, while her death record, $r_4(Dd)$ and her birth mother record, $r_9(Bm)$, have her married surname. Assume (r_1, r_9) is already merged as in Figure 4 (b) and associated with the entity o_1, which has all QID values of r_1 and r_9. Then, when we consider the node (r_1, r_4), because r_1 now has the associated entity o_1, we compare all QID values of r_4 with the corresponding values of o_1 to find the best matching atomic nodes with highly similar values. As the surname of r_1 is *Smith* and of r_4 is *Taylor*, the node (r_1, r_4) is already associated with the atomic node (*Smith, Taylor*). When we compare the surname of r_4, *Taylor*, with the surnames of o_1, and assuming $sim(Tayler, Taylor) > sim(Smith, Taylor)$ based on a string similarity functions [11], we remove the edge from the atomic node (*Smith, Taylor*) and add a new edge from the node (*Tayler, Taylor*). In this way, even if an individual changes their name or address, we can still identify them because of the propagation of QID values.

4.2.2 Global Propagation of Constraints (PROP-C). The second challenge is the different relationships that we encounter in records at different points in time. Since these relationships are between different entities, we cannot directly compare the corresponding records when making link decisions. However, in *SNAPS* we utilise such different relationships as negative evidence for any subsequent links by modelling the characteristics

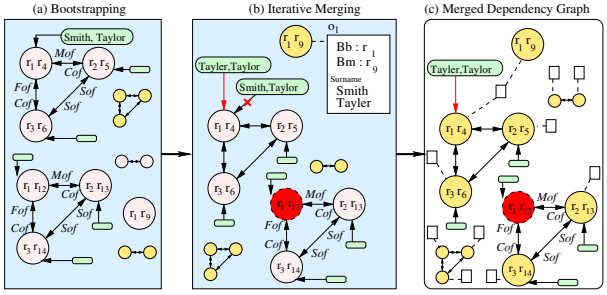

Figure 4: (a) Bootstrapping where groups of nodes that have a high average similarity (greater than a predefined threshold) are merged. (b) Merging step where nodes are merged applying techniques such as propagation of QID value changes. For example, node (r_1, r_4) is updated with the surname atomic node (*Tayler,Taylor*).

of relationships as constraints. For example, the time difference between a birth baby (*Bb*) becoming a birth mother (*Bm*) should be (biologically) at least 15 and at most around 55 years [58]. Similarly, we can also add such constraints for other role pairs based on domain knowledge [12]. As these constraints are related to temporal aspects, we refer to them as *temporal constraints*.

Furthermore, there can be constraints that are based on the properties of certain relationships. For example, in Figure 4, since $r_1(Bb)$ is being linked with $r_4(Dd)$, r_1 cannot be linked to any other deceased person such as $r_{12}(Dd)$ (and vice versa) because a person can have only one birth and only one death record. We refer to such constraints as *link constraints* as they are related to the links between entity roles. These link constraints can be one-to-one and one-to-many based on the pairs of entity roles they are applied on. Because these constraints are likely domain dependent, they need to be manually specified by domain experts or learned from training data to be used with *SNAPS*.

This technique of propagating QID values and constraints has first been proposed by Dong et al. [19]. The novelty in our propagation method is that we make a global propagation of changing QID values whereas Dong et al. used an exhaustive search to merge relational nodes in the dependency graph $\mathbf{G}_D$.

4.2.3 Leverage Ambiguity of QID Values (AMB). The next challenge we discussed in Section 2 is the ambiguity of QID values encountered in person records, where many entities potentially share the same QID values such as common surnames or city names [11]. To address this challenge we incorporate ambiguity into the similarity calculations between two records. We specifically calculate a similarity score, s, for each node in the graph $\mathbf{G}_D$ consisting of an atomic similarity, s_a and a disambiguation similarity, s_d, defined as:

$$s_a(r_i, r_j) = \frac{w_M \cdot s_M(r_i, r_j) + w_C \cdot s_C(r_i, r_j) + w_E \cdot s_E(r_i, r_j)}{w_M + w_C + w_E}, \quad (1)$$

$$s_d(r_i, r_j) = \frac{\log_2(|\mathbf{O}|/(r_i.f + r_j.f))}{\log_2 |\mathbf{O}|}, \quad (2)$$

$$s(r_i, r_j) = \gamma \cdot s_a(r_i, r_j) + (1 - \gamma) \cdot s_d(r_i, r_j), \quad (3)$$

where $0 \leq \gamma \leq 1$ is the weight distribution for the two similarity components, s_a and s_d, as we describe next.

To calculate s_a, we consider all incoming atomic nodes of a relational node. Each atomic node has a score that represents the similarity of two QID values. The contribution of different QID value similarities will affect differently towards the calculation of s_a. For example, in person records, QID values such as first names are more important because they are more complete and stable over time, whereas QID values such as occupation or address can be missing and can change over time [33].

To this end, we categorise attributes into *Must*, *Core*, and *Extra* attributes, based on their importance in the ER process determined using domain knowledge or data characteristics, such as completeness [14, 50, 58]. For two records to be classified similar, they need to have highly similar values in the *Must* attributes (such as first name), but they can have a comparatively lower similarity in *Core* attributes (like surname). *Extra* attributes (such as occupation) provide further evidence of similarities between records. We calculate an atomic similarity as shown in Equation (1), where s_M, s_C, and s_E represent the average of atomic node similarities of the *Must*, *Core*, and *Extra* attribute categories, while w_M, w_C, and w_E represent their corresponding weights.

Disambiguation similarity, s_d, accounts for the ambiguity of QID (attribute) values. If the pair of records in a node has QID values that occur frequently in the data set, then a high atomic similarity, s_a, of the node is not significant compared to a pair of records having less frequent (or even unique) QID values. For example, if *Smith* occurs commonly in a data set and *Taylor* occurs only a few times, then two records having the name *Taylor* have a higher likelihood of referring to the same individual compared to two records having the same name *Smith*.

As link decisions are dependent on each other, we need to prioritise unique record pairs such that they are processed before ambiguous pairs. As this is similar to the concept of inverse document frequency used in information retrieval, we use a normalised score of inverse document frequency [60] as the disambiguation similarity s_d. Let $r_i.f$ and $r_j.f$ be the two frequencies of a combination of several QID values of two records in a node. If the number of unique records in the data set is $|\mathbf{O}|$, we define s_d as per Equation (2).

Assume that node (r_1, r_4) in Figure 4 has three incoming atomic nodes, a surname node $(Tayler, Taylor)$, a first name node $(Mary, Mary)$, and a city node $(Klmor, Kilmore)$ with node similarities of 0.9, 1.0, and 0.9, respectively. Assuming we set w_M, w_C, and w_E as 0.5, 0.3, and 0.2, respectively (based on domain knowledge) and consider first name $(Mary, Mary)$ as a must attribute, surname $(Tayler, Taylor)$ as a core attribute, and city $(Klmor, Kilmore)$ as an extra attribute. Then, we can calculate $s_a(r_1, r_4)$ as $\frac{0.5 \cdot 1.0 + 0.3 \cdot 0.9 + 0.2 \cdot 0.9}{0.5 + 0.3 + 0.2} = 0.95$ using Equation (1). Similarly, assuming $r_1.f = 45$, $r_4.f = 12$, and $|O| = 100$, using Equation (2) we can calculate $s_d(r_1, r_4)$ as $\frac{\log_2(100/(45+12))}{\log_2(100)} = 0.12$.

Bhattacharya and Getoor [6] have explored ambiguity of static attribute values in relational clustering for collective ER. However, they have not investigated how to incorporate disambiguation while propagating link decisions, or when attribute values can change over time.

4.2.4 Adaptive Leveraging of Relationship Structure (REL). To leverage relationship structures in the ER process, recent approaches have proposed different methods such as the use of average similarity of the relationally connected groups [49] or using a separate similarity component with an individual weight given to the similarity of relationally connected nodes [6, 19]. Both of these methods have limitations in resolving the partial match group problem we discussed in Section 2. For example, assume a group of two siblings and their parents. We need to merge the two parent nodes but not the sibling node. The sibling node will have a lower similarity as it refers to two different individuals. If we consider any of the methods from the literature,

then this sibling node will reduce the chances of the two parent nodes getting merged since the lower similarity of the sibling node reduces the overall similarity.

Therefore, in *SNAPS* we employ a novel iterative approach to exploit the relational structure of records. In merging nodes, we consider the average similarity of connected groups of nodes. Therefore, when we consider merging the node (r_1, r_4) in Figure 4, we consider the average similarity of the whole group including (r_2, r_5) and (r_3, r_6). If both neighbouring nodes have high similarities then this provides strong evidence that (r_1, r_4) represents the same entity.

The novelty in this technique is in the iterative merging process. For example, in Figure 4 assume that r_1 and r_{12} are two siblings. Therefore, the node, (r_1, r_{12}) has a lower similarity because it represents two entities. Now when we consider the group of nodes, (r_1, r_{12}), (r_2, r_{13}), and (r_3, r_{14}), in the first iteration these nodes are not merged due to the low average similarity (because the two siblings are different entities). Therefore, in the next iteration we remove the node with the lowest similarity, (r_1, r_{12}), which is the sibling node. The remaining parent nodes are then merged because they have a high average similarity.

4.2.5 Dynamic Refining of Record Clusters (REF). The propagation of link decisions can lead to poor linkage quality if two records that refer to two different entities are linked together. To remove such wrong links, we employ a novel refining step after each of the bootstrapping and merging steps. We create an individual graph for each entity that captures the records of an entity and their links. We apply the graph measure based error identification proposed by Randall et al. [57] on these graphs with the hypothesis that loosely connected groups of records (such as chains) are more likely to contain errors compared to densely connected groups (such as cliques). Unmerging of likely wrong links allows correct records to be linked in the next iteration.

We use the graph measures of *bridges* and *density* to identify loosely connected record clusters. A bridge is an edge that will disconnect the graph if removed; and density, d, is measured by the number of edges out of the total number of possible edges in a graph [57], calculated as $d = 2|E'|/(|N'| \cdot (|N'| - 1))$, where E' and N' are the edges and nodes of the graph associated with the entity. For a graph having at least three records, we calculate the density and if it is less than a threshold, t_d, we remove the node with the lowest degree. For a node group with more than t_n records, we split the record cluster by any existing bridges.

4.2.6 Bootstrapping and Merging Steps. We now describe how to employ these techniques in our bootstrapping and merging steps. The bootstrapping step is basically to merge the highly similar relational nodes. Therefore, we merge only nodes in groups (leaving the singletons), where the average atomic similarities of all nodes in a group must be at least the bootstrap threshold which we set to $t_b = 0.95$ (based on initial experiments) to achieve highly similar bootstrap links, as illustrated in Figure 4 (a). We only consider groups having high average similarities at this stage rather than individuals because groups can provide more relationship evidence than individuals.

The merging step proceeds by maintaining a priority queue of node groups. The queue is initialised by all relational node groups in $\mathbf{G}_D$, giving precedence to larger groups and then to the groups that have a high average similarity of nodes. In every iteration, we perform merging of the top node group in the queue. For each node in that group, we check if a node is valid to be merged based on temporal and link constraints. This is where we utilise the technique of global propagation of constraints (*PROP-C*) to validate record pairs by applying constraints based on previous links. The previous links may or may not have associated an entity $o \in \mathbf{O}$ to records in a node. If the records in a node are associated with two different entities, we validate the node by applying constraints on every possible record pair between the entities to merge those two entities. Otherwise, we apply constraints between the original records.

For each node that satisfies the constraints, we propagate the QID value changes of the node using the technique *PROP-A* and then calculate its new similarity based on the technique *AMB*. We then calculate the average similarity of the node group. If it is lower than a predefined threshold, t_m, we remove the node with the lowest similarity and recalculate the average similarity. Similarly, if there is any node that violates any constraints, then this node is also removed from the node group based on the *REL* technique. We do this iteratively until either we find a node group that satisfies the constraints with an average similarity of at least t_m and merge it, or until the node group becomes a pair. We continue the merging process until all the node groups in the queue are processed. After each of the bootstrapping and merging steps we perform dynamic refining of record clusters (*REF*) to remove any wrong links in the record clusters.

Algorithm 1: ***Pedigree Graph Generation***

Input:
- $\mathbf{G}_D$: Merged dependency graph

Output:
- $\mathbf{G}_P$: Pedigree graph

1: $\mathbf{G}_P = Graph()$ // Initialise $\mathbf{G}_P$ to an empty graph
2: **for** $n \in \mathbf{N}_R$ **do:** // Iteration to add nodes
3: **if** $n.isMerged$ **then:** // If the node is merged
4: $o = \mathbf{G}_d.getEntity(n)$ // Get entity associated with the node
5: **if** $o \notin \mathbf{G}_P.nodes$ **then:** // If entity o is not in $\mathbf{G}_P$
6: $\mathbf{G}_P.addNode(o)$ // Add node for this entity
7: **for** $n \in \mathbf{N}_R$ **do:** // Iteration to add edges with relationships
8: **if** $n.isMerged$ **then:** // If the node is merged
9: $o = \mathbf{G}_d.getEntity(n)$ // Get entity associated with the node
10: **for** $\rho \in \{mother, father, spouse, child\}$ **do:**
11: $\mathbf{N}^n_\rho = \mathbf{G}_d.getNeigbourNode(n, \rho)$ // Neighbours with relationship ρ
12: **for** $n_\rho \in \mathbf{N}^n_\rho$ **do:** // Iteration through neighbours
13: **if** $n_\rho.isMerged$ **then:** // If the node is merged
14: $o_\rho = \mathbf{G}_d.getEntity(n_\rho)$ // Get entity associated with the node
15: $\mathbf{G}_P.addEdge(o, o_\rho, relationship = \rho)$ // Add edge
16: return $\mathbf{G}_P$

5 PEDIGREE GRAPH GENERATION

Our final task is to retrieve family pedigrees for each individual entity. In particular, we generate a pedigree graph, $\mathbf{G}_P$, utilising the merged nodes and associated entities from $\mathbf{G}_D$.

The pedigree graph generation, shown in Algorithm 1, first iterates through the nodes in $\mathbf{G}_D$ and selects merged nodes in lines 2 and 3. As these merged nodes are associated with an entity $o \in \mathbf{O}$, we add the node to $\mathbf{G}_P$ if it is not already in $\mathbf{G}_P$ (lines 4 to 6). As each entity o is associated with either a single or multiple nodes in $\mathbf{G}_D$, o by now has a set of records from $\mathbf{R}$ where each record refers to a different role in a certificate. Therefore, we also add the QID values of the associated records of an entity, such as its first names, surnames, and locations, to the nodes in $\mathbf{G}_P$.

The edges in $\mathbf{G}_P$ reflect the relationships among the entities $o \in \mathbf{O}$. Therefore, we again iterate through the nodes of $\mathbf{G}_D$ to find the neighbouring relationships of an entity and add edges to $\mathbf{G}_P$ in lines 7 to 15. For a node in $\mathbf{G}_D$, we get the associated entity in line 9, and check if neighbouring nodes also have associated entities in lines 10 to 14. If there exists any neighbouring node

that has an associated entity, we add an edge between those two entities in $\mathbf{G}_P$ with their relationship ρ (line 15). In the pedigree graph we generate, we consider the relationships of *mother*, *father*, *spouse*, and *child*. The final pedigree graph, $\mathbf{G}_P$, is then used in Section 8 to extract family pedigrees (family trees).

6 KEYWORD AND SIMILARITY INDEX GENERATION

To speed-up query processing and ranking, from the pedigree graph we generate a keyword index and a string similarity index. The keyword index, $\mathbf{K}$, captures the mapping between QID (attribute) values and entities, $o \in \mathbf{O}$. For QIDs such as first name and surname, gender, birth/death year, and location, this index holds the corresponding entity identifiers that map to the nodes in the pedigree graph, $\mathbf{G}_P$. The keyword index is used in query processing and ranking to find the entities that have the corresponding query strings that were provided by the user.

To speed-up the retrieval of approximate matches, we employ a similarity-aware index, $\mathbf{S}$ [13]. In the offline phase we pre-calculate the similarities between all pairs of string QID values (names and locations) in the keyword index, $\mathbf{K}$, that share at least one bigram (two-character substring [11]). For every string value in $\mathbf{K}$, we keep all other values in $\mathbf{K}$ that have a normalised string similarity of at least s_t, with $0 < s_t < 1$, using the Jaro-Winkler approximate string comparison function [11]. Comparing a string with itself would result in a similarity of 1, while comparing two strings that have no letter in common would result in a similarity of 0. We set $s_t = 0.5$ because this will retrieve approximate matches that have adequate similarity, while at the same time limiting the size of the generated similarity-aware index $\mathbf{S}$, and therefore the number of generated candidate matches [13].

7 QUERY PROCESSING AND RANKING

As shown in Figure 1, the first step of the online component of *SNAPS* is to process user queries and rank the retrieved entities from the pedigree graph containing QID values based on their overall similarities to the given query values. A user query as entered on the web interface shown in Figure 5 will contain a first name and surname (mandatory), the type of records (birth or death) to be searched, and optionally a gender, year (of birth or death), and a location (such as a parish or district name). These are the details of the individual for which the user wants to retrieve a family pedigree from $\mathbf{G}_P$.

In common with any ER method [11, 21], uncertainty in the actual spelling of names, or the years and locations of the birth or death of a person, means that a user query can contain inaccurate QID values with misspellings in strings or wrong year values. Only retrieving entities that have exact matching values could miss the true matching person the user is searching for.

We employ approximate string matching functions [11] to allow for variations in names and locations. We generate an accumulator $\mathbf{M}$ [5], where we first retrieve from the keyword and similarity indices, $\mathbf{K}$ and $\mathbf{S}$, the identifiers of all entities from the pedigree graph, $\mathbf{G}_P$, that have a match on first name and/or surname. We consider both exact and approximate string matches, and we sum the match scores of the two name QIDs for these entities [13], as we describe below. We require matching names because otherwise the query result set can contain entities that have an exact match on year, gender, and location, but their names are very different (which would not be useful).

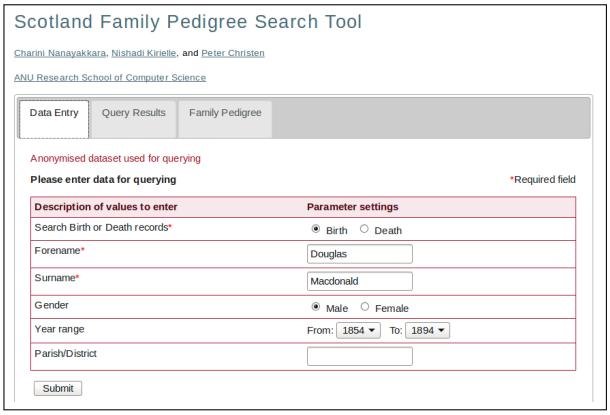

Figure 5: Data entry tab with example query values.

If a QID value in a query does not occur in $\mathbf{K}$, then we retrieve all values in $\mathbf{K}$ that share at least one bigram with the query value, and calculate the similarities between the query value and the retrieved values. We keep pairs with a similarity of at least s_t and add them to $\mathbf{S}$ to speed-up future queries of the same value.

We then refine the retrieved candidate matches in the accumulator $\mathbf{M}$ (a set of entity identifiers from the keyword index $\mathbf{K}$ that have exact or approximate name matches) with their corresponding matching gender, year, and location (if such query values were provided by the user). If these query values are empty we assume the user is willing to consider any entity in the result set irrespective of their QID values (for example, entities with any location). If a query value is provided for year, gender, and/or location, then we retrieve the entity identifiers from $\mathbf{K}$ that have this value, and increase the match scores of all corresponding entities in $\mathbf{M}$ [5]. We however do not add new entities to the accumulator as these would not have matching name values.

At the end of this process, each entity in the accumulator $\mathbf{M}$ has a set of (exactly or approximately) matched QID values, with a corresponding match score for each value provided in the query. For a given entity $o \in \mathbf{M}$, its match score, s_r, will be the weighted sum of similarities over its QID values that were provided in the query, $s_r = \sum_{a \in \mathbf{A}_q} w_a \cdot sim(q_a, o_a)$. Here, q_a and o_a are the query and entity value in attribute a, respectively, $\mathbf{A}_q$ is the set of QID attributes (from first name, surname, year, gender, and location) for which a value was provided through the query interface, and $sim()$ is a string similarity function.

The individual attribute match scores can be assigned weights, w_a, indicating their importance. For example, first name and surname can be assigned a higher weight than location because name values that match provide more evidence that an entity is relevant to a given query compared to the location of a birth or death record (which might not be exactly known by the user). While we currently set these match weights manually, in future work we aim to learn optimal match weights [23] based on ground truth data.

We then rank the entities $o \in \mathbf{M}$ based on their overall match scores, and select the top m entities with the highest scores. These m entities are then passed back to the web interface and presented to the user as a ranked list. We normalise the overall match scores into a percentage, with 100% indicating an entity from the pedigree graph, $\mathbf{G}_P$, matches exactly on all QID values provided by the user in their query.

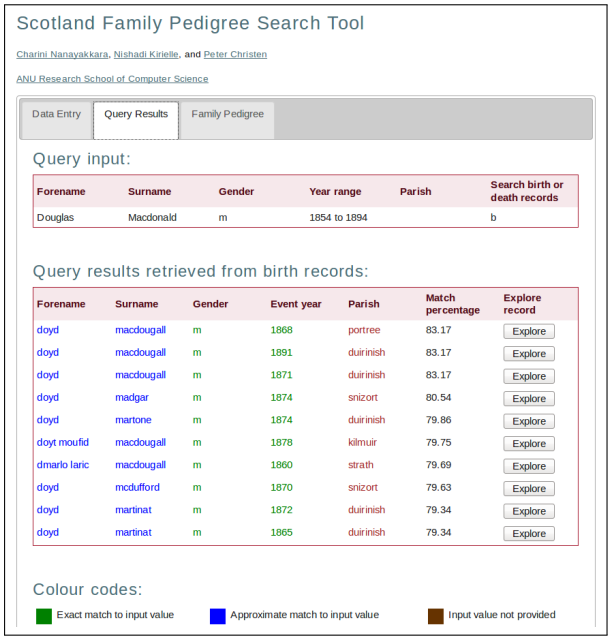

Figure 6: Screenshot of *SNAPS* showing ranked results for the example query 'Douglas Macdonald' shown in Figure 5.

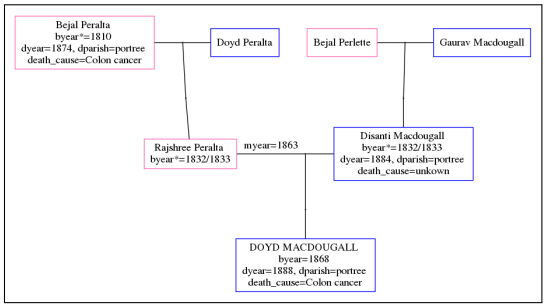

Figure 7: Graphical family pedigree for the selected (top ranked) record of 'Doyd Macdougall' from Figure 6.

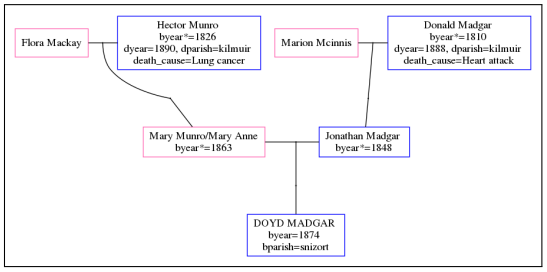

Figure 8: A family pedigree of a baby, 'Doyd Madgar'.

We show QID values in different colours to highlight exact and approximate matches, as shown in Figure 6. This will allow the user to make an informed selection of an entity, or modify their search query.

8 FAMILY PEDIGREE TREE EXTRACTION AND VISUALISATION

As the user obtains a ranked list of matched records (each corresponding to an entity in the pedigree graph G_P) from the query processing and ranking step, as shown in Figure 6 they can select to explore one of the listed entities. Exploring will extract the family pedigree, p, for the selected entity from G_P which was generated in the offline component of *SNAPS*.

In order to extract the family pedigree for a particular entity, we first retrieve the entity from G_P (its node), and then, based on the number of generations g required, we extract the neighbouring nodes up-to g hops away. We currently set $g = 2$, however larger values are possible. The nodes located 1 hop away are the entities one generation before or after (parents and children), whereas nodes located 2 hops away are the entities two generations away (grand parents and grand children). The extracted family pedigree is provided to the user on the web interface both in textual form, as well as a graphical family tree[4]. The family pedigrees shown in Figures 7 and 8 are hierarchical trees where higher levels indicate older generations, and where gender is shown by different colours.

9 GRAPH DATA ANONYMISATION

The statutory records used by the Genetics Genealogy Team of Public Health Scotland for determining the risk of inherited genetic conditions are real-world personal data. If exposed to the public these may reveal sensitive personal details and therefore pose a threat to the privacy of individuals [14]. We therefore cannot make the original Scottish data publicly available via our *SNAPS* web interface as this would violate the data protection regulations in Scotland. Developing anonymisation techniques for protecting sensitive information in data, such that sensitive data sets are rendered safe for conducting research and building applications such as our web interface, is therefore vital for the advancement of science. While many anonymisation approaches have been proposed [18], the majority of them compromise the human interpretability and patterns in a data set in order to preserve its privacy [64, 66]. However, in applications such as ours, where family pedigrees are generated, it is important to preserve the structure and characteristics of the original data set in the anonymised version as well, such as the preservation of string similarities across names.

Since family pedigrees are graph data representations, we use a graph anonymisation technique [45] to anonymise the sensitive Scottish data set such that the human interpretability and the similarities between QID (attribute) values in the sensitive original data set are preserved.

This graph anonymisation technique initially maps QID values of the sensitive graph data set to QID values from a public data source using a cluster-based mapping approach. Following this approach, we separately cluster female first names, male first names, and surnames in the sensitive and public data sets, such that highly similar names appear in the same cluster and dissimilar names in different clusters. Subsequently, each sensitive name value cluster is mapped to the best matching public name value cluster, where a best match is determined by how similar the intra-cluster similarity values are across clusters. Once cluster mapping is completed, it is possible to determine which public name value is to be used as a replacement for each sensitive name value. We used name QID values from a publicly available US voter database (see: https://dl.ncsbe.gov) for anonymisation. Furthermore, we shift all date values by a global offset to hide the actual years of birth and death to ensure anonymity [45].

[4]See: https://github.com/adrienverge/familytreemaker

Table 2: Characteristics of the data sets used in the experimental evaluation.

Data set	Role pair	Interpretation (links between)	Number of records		Record pairs	True matches
			Role-1	Role-2		
Isle of Skye	Bp-Bp	Birth parents in birth certificates	34,272	34,272	436,518	83,132
(IOS) [58]	Bp-Dp	Parents in birth and death certificates	34,272	23,938	628,141	38,662
Kilmarnock	Bp-Bp	Birth parents in birth certificates	74,948	74,948	1,571,991	135,346
(KIL) [58]	Bp-Dp	Parents in birth and death certificates	74,948	45,186	2,357,625	80,819

Some causes of death can be highly sensitive and potentially identify individuals, for example if locations or causes of accidents, or criminal aspects of a death are described. To anonymise such sensitive causes of death, we employ an approach inspired by k-anonymity [61], where we first identify all frequent causes of death strings that occur at least $k > 1$ times. For each cause of death string that is rare and occurs less than k times we then find the most similar string using the Jaccard coefficient based approximate string similarity [11], and we replace the rare cause of death string with its most similar frequent string.

In order to prevent odd causes of death, such as men dying of *ovarian cancer* or infants of *old age*, we apply this anonymisation approach gender specific as well as stratified on age categories. As appropriate for our historical data, we specify *young* for ages up-to 20 years, *middle* for ages 20 to 40, and *old* for 40 years and over. We set $k = 10$ as this provided us with a reasonably anonymised data set while still having around 100 frequent death causes (and over 2,000 rare death causes). If no frequent similar death cause can be found based on the age and gender restrictions we set a specific rare death cause to *not known*.

The *SNAPS* web interface made available to the general public for educational and training purposes does not facilitate searching for real-world Scottish people due to the application of the graph anonymisation methods we describe here. Rather, as we previously explained, it retrieves people with names from the US voter database, and their corresponding pedigrees. A family pedigree search tool based on an anonymised data set that preserves the structure of the original sensitive Scottish data is useful in training new users, for educational demonstrations, and for assessing the efficiency and effectiveness of *SNAPS*. Such an implementation also enhances the transparency of what institutions such as Public Health Scotland do with personal data, and is instrumental in obtaining the trust and approval of a wider audience for the invaluable work they conduct.

10 EXPERIMENTAL EVALUATION

In this section we conduct experiments to evaluate the linkage quality and efficiency of the *SNAPS* application.

Data Sets: We evaluated *SNAPS* on the two real data sets from the Isle of Skye (IOS) and the town of Kilmarnock (KIL) [58] as we described in Section 2. Both contain birth, death, and marriage certificates from 1861 to 1901 as characterised in Table 2. From each certificate, we extracted records $r \in \mathbf{R}$ that correspond to individuals, as we described in Section 4. Similar to other historical data sets [2, 25], these two data sets have a small number of unique name values, as well as numerous transcription errors and name variations, that make the problem challenging [12].

Demographers with expertise in linking such data have curated and linked both the IOS and KIL data sets [58]. Their semi-automatic approach is biased towards certain types of links, such as *Bp-Bp* (links between birth parents across two birth certificates) and *Bp-Dp* (links between birth parents and death parents), as their research interests were in identifying siblings of the same mother and exploring child mortality. Therefore, we show results of *Bp-Bp* for which we have directly curated ground truth links, along with results for *Bp-Dp* which we inferred through indirect *Bb-Dd* true links.

For scalability experiments we use the publicly available Brabant Historical Information Center (BHIC) data set [10] because our other data sets are quite small. This data set contains civil certificates from North Brabant, a province of the Netherlands, in the period from 1759 to 1969. However, we cannot show the linkage quality of the BHIC data set as there is no ground truth data available.

Implementation and Parameter Settings: We implemented *SNAPS* in Python 2.7 and all offline components were executed on a server running Ubuntu 18.04 with 64-bit Intel Xeon 2.10 GHz CPUs and 512 GBytes of memory, while the *SNAPS* web interface was deployed using Apache 2.4.38 Debian server and PHP 7.3.11. To facilitate repeatability, the code and the anonymised data sets are available from: https://dmm.anu.edu.au/SNAPS/.

In the graph-based ER step described in Section 4 and for all baselines, we employed indexing based on locality sensitive hashing [54] to block records and find likely matches. Then, in the record pair comparison step, we used similarity functions such as Jaro-Winkler for names and the Jaccard coefficient for other textual strings [11] to compare QID values between records. For numerical comparisons we used the maximum absolute difference [11], while for comparing addresses in the IOS data set we geocoded addresses [37] and calculated similarities based on the distances between two locations. Due to the absence or low quality of addresses we did however not consider geocoding for the KIL and BHIC data sets.

Based on a parameter sensitivity analysis, we set the default merging threshold to $t_m = 0.85$, the atomic node similarity threshold to $t_a = 0.9$, the weighting distribution in the similarity score to $\gamma = 0.6$, and for graph measures we set the thresholds $t_n = 15$ (bridges) and $t_d = 0.3$ (density). We provide further details of our parameter sensitivity analysis on the *SNAPS* web site.

Baselines: Because we do not have any end-to-end family pedigree search system to compare *SNAPS* with, we evaluated its graph-based ER approach with three other ER baselines.

(1) *Attr-Sim* provides basic pairwise similarity based linking to obtain a baseline similar to traditional record linkage [11].

(2) *Dep-Graph* is an implementation similar to the collective ER approach proposed by Dong et al. [19] that propagates link decisions in the ER process, where we apply the same set of temporal and link constraints as we employed in *SNAPS*.

(3) *Rel-Cluster* is a similar implementation to the method proposed by Bhattacharya and Getoor [6] that employs ambiguity of

Table 3: Ablation analysis for *SNAPS* that shows how each key technique in the application affects linkage quality.

Data Set	Role Pair		SNAPS	without PROP-A and PROP-C	without AMB	without REL	without REF
IOS	Bp-Bp	P	98.73	86.79	99.22	**99.88**	98.02
		R	94.70	**95.20**	93.56	61.58	94.87
		F*	**93.56**	83.15	92.89	61.53	93.08
IOS	Bp-Dp	P	86.44	72.56	**89.72**	0.00	85.28
		R	92.87	**93.24**	88.62	0.00	93.14
		F*	**81.06**	68.93	80.45	0.00	80.24

Table 4: Precision (P), Recall (R), and F*-measure results of *SNAPS* compared to the baselines (averages ± standard deviations).

Data Set (Role Pair)		*SNAPS*	Attr Sim	Dep-Graph	Rel-Cluster	Magellan
IOS (Bp-Bp)	P	**98.73**	63.67	90.87	93.59	77.9 ± 33.4
	R	**94.70**	88.41	65.26	63.72	72.9 ± 35.1
	F*	**93.56**	58.76	61.25	61.06	60.4 ± 38.6
IOS (Bp-Dp)	P	**86.44**	43.05	0.00	80.91	67.8 ± 37.9
	R	**92.87**	72.32	0.00	49.19	62.2 ± 41.4
	F*	**81.06**	36.96	0.00	44.07	46.1 ± 40.4
KIL (Bp-Bp)	P	**97.81**	30.26	54.81	71.81	69.6 ± 40.1
	R	**89.52**	89.13	74.93	71.92	62.7 ± 46.7
	F*	**87.76**	29.18	46.32	56.09	51.6 ± 45.9
KIL (Bp-Dp)	P	**74.36**	11.05	28.95	30.35	63.9 ± 36.1
	R	89.57	**90.49**	70.69	43.18	61.8 ± 44.1
	F*	**68.44**	10.93	25.85	21.69	45.6 ± 39.4

QID values in the ER process. We used the same set of temporal and link constraints as with *SNAPS* for this baseline.

(4) *Magellan* is a state-of-the-art supervised ER system [17], where we selected four classifiers (a SVM, a random forest, a logistic regression, and a decision tree) and averaged their linkage quality results, where these four achieved the best performance in our experiments among the classifiers available in Magellan.

Linkage Quality: To assess the quality of links in family pedigrees, we use precision, $P = TP/(TP + FP)$, and recall, $R = TP/(TP+FN)$, where TP is the number of true matches classified as matches, FP is the number of true non-matches classified as matches, and FN is the number of true matches classified as non-matches [11]. We do not use the F-measure as recent research has found that it is not suitable for measuring linkage quality in ER [29] because the relative importance given to precision and recall in the F-measure depends on the number of classified matches. Instead we use an alternative for F-measure, the F^*-measure [30], calculated as $F^* = TP/(TP + FP + FN)$, which is more interpretable and is a monotonic transformation of the F-measure.

Table 4 shows precision, recall, and F*-measure results of *SNAPS* compared to the four baselines detailed above. As can be seen, *SNAPS* is able to outperform all baselines. For both the IOS and KIL data sets, *SNAPS* obtains high precision, recall, and F* results for the role pair *Bp-Bp*. For *Bp-Dp* it has a small drop in results which is because we have an incomplete (inferred or biased) set of ground truth links for the *Bp-Dp* role pair [58].

The *Attr-Sim* baseline shows poor linkage quality for all of the data sets. This is a good indication that traditional pairwise linkage is not sufficient to address the challenges associated with resolving records with person data. *Dep-Graph* [19] and *Rel-Cluster* [6] are two collective ER baselines. We can see that both of these approaches show poor linkage quality results compared to *SNAPS* even though they exploit relationship information. The *Dep-Graph* baseline propagates QID value changes and constraints in the process of resolving entities. However, in a context where ambiguities occur *Dep-Graph* fails because it does not perform disambiguation. Similarly, it does not deal with the problems of partial match groups and incorrect links that *SNAPS* addresses. This is also a reason behind the lower results for the *Rel-Cluster* baseline. While *Rel-Cluster* addresses the problem of ambiguity it does not deal with changing QID values, partial match groups, or incorrect links.

We present the results for *Magellan* as averages with standard deviations since we used four different classifiers and multiple settings to generate the training and testing data sets. Since we have different role pairs in both the IOS and KIL data sets, we trained *Magellan* in two different ways. First, we trained it only on record pairs of the specific role pair that is being tested, and second we trained it on the full data set. We considered the second approach because in most practical scenarios it is likely to train a classifier on record pairs of all role pair types due to the availability of incomplete ground truth data. From Table 4 we can see that *Magellan* has a higher standard deviation in these two settings. It obtains better results compared to *SNAPS* when trained only on the data from the specific role pair type, however in practical scenarios not enough training data might be available for each pair of role types. On the other hand, when trained on records of all role pair types then *Magellan* achieves poorer linkage quality. These results illustrate the sensitivity of supervised learning methods with regard to the type of training data used, while the unsupervised approach we employ in *SNAPS* consistently obtains better linkage results.

Table 3 provides an ablation analysis for *SNAPS* that shows how important each novel key technique is towards obtaining high quality ER results. We include results only for the IOS data set due to the limited space available. We show results with one novel key technique employed in *SNAPS* (discussed in Section 4.2) removed at a time. Since both *PROP-A* and *PROP-C* propagate link decisions, we consider them as a single component in this analysis. Therefore we show results without *PROP-A* and *PROP-C*, without *AMB*, without *REL*, and without *REF*, separately.

When we removed *PROP-A* and *PROP-C* we neither propagate positive nor negative evidence in the linkage process, and we can see that the F*-measure results drop up-to 12%. This drop occurs because we do not consider any QID value changes and constraints. This indicates that the propagation of link decisions is necessary to obtain high quality linkage results for person data where value changes do occur and constraints can be applied.

To remove the effect of ambiguity (*AMB*) from *SNAPS* we then removed the disambiguation similarity from the similarity calculation and calculated overall similarities only based on QID

Table 5: Runtime results (in seconds) for offline component of *SNAPS* and baselines.

Data Set	$\|\mathbf{n_A}\|$	$\|\mathbf{n_R}\|$	SNAPS	Attr-Sim	Dep-Graph	Rel-Cluster	Magellan
IOS	74,851	2,992,834	372	**50**	176	358	10,059
KIL	1,565,730	11,190,176	1,945	**178**	1,207	7,663	9,632

Table 6: Runtimes of the offline component of *SNAPS* for different graph sizes of the BHIC data set generated for different time periods. Linkage time is the total of bootstrapping and merging steps.

Time Period	Number of Nodes	Number of Edges	Generate N_A time (s)	Generate N_R time (s)	Bootstrap time (s)	Iterative Merging time (s)	Linkage time (ms) per node	Linkage time (ms) per edge
1900 - 1935	22,928,967	41,121,771	20,642	1,438	896	23,155	1.0	0.6
1890 - 1935	42,398,382	80,524,946	28,881	2,172	1,685	113,143	2.7	1.4
1880 - 1935	68,739,033	134,057,215	36,033	3,910	3,013	299,123	4.4	2.3
1870 - 1935	100,907,697	199,588,456	39,113	6,062	5,423	660,896	6.6	3.3

similarity by setting $\gamma = 1$ in Equation (3). In Table 3 we can see that recall dropped up-to 4% when disambiguation similarity was removed, which is because ambiguous record pairs with high QID similarity are linked wrongly, thereby preventing the correct ones from being linked with enforced constraints.

Then we removed the adaptive leveraging of relationship structure (*REL*). It is interesting to see that the *Bp-Dp* role pair has zero results for all linkage quality measures. This has happened because of the existence of partial match groups when we consider the birth parents to death parents role pair. At the bootstrapping step none of the groups having the *Bp-Dp* role pair have been linked due to the partial match groups. That is why in the merging process none of the correct ones have been linked.

Finally, we removed dynamic refining (*REF*) of entity clusters from *SNAPS* resulting in precision to drop for both role pairs. The drop in the obtained results is small as we only have small clusters in these data sets. With larger data sets that contain bigger clusters the improvement with *REF* will be larger because the record clusters will be larger.

Scalability: First, we evaluate the runtime of the offline component of *SNAPS* compared to the baselines in Table 5. *Attr-Sim* has the best runtimes for both data sets because it simply links records without considering any relationships. The next best runtimes are from *Dep-Graph* [19]. *SNAPS* takes more time compared to *Dep-Graph* because it addresses all challenges specified in Section 2, whereas *Dep-Graph* addresses only the problems of changing QID values and different relationships. *Rel-Cluster* has longer runtimes compared to both *SNAPS* and *Dep-Graph* because of the iterative clustering method employed. The worst performing baseline is *Magellan* (these runtimes are averages for the four supervised classifiers and two different settings we described above) as it consumes much time for training the supervised classification models. Overall, the runtimes of *SNAPS* are comparatively better than the other baselines given it addresses all challenges specific to personal data.

Next, we evaluate the scalability of the offline component of *SNAPS* by comparing the runtimes on different sized data sets in Table 6. For that we vary the time periods of records considered for generating the graph with the BHIC data set. These runtimes indicate that the merging step accounts for the largest component of the overall runtime because it is the most time consuming step that involves most of the key techniques described in Section 4. To measure the scalability we use total linkage time. Considering the values of linkage times per node and per edge, we can see that our proposed framework has a near linear scalability with both, which indicates that *SNAPS* can scale to large graphs.

Table 7: Minimum, average, median, and maximum time in seconds for querying and extracting family pedigrees.

Task	Minimum	Average	Median	Maximum
Querying	1.32	1.34	1.33	1.40
Pedigree extraction	0.66	0.74	0.74	0.92

Table 7 shows the minimum, average, median, and maximum number of seconds taken for querying and for extracting family pedigrees. As shown in the table, both querying and pedigree extraction can be done in well under two seconds, whereas doing the same task manually would have taken several days of laborious manual work. This attests to the efficiency with which family information can be retrieved with *SNAPS*.

11 RELATED WORK

We now describe related work, with a focus on ER of familial and historical data, query-time ER, and graph-based ER techniques.

Family pedigrees are increasingly being used in various biomedical research fields, ranging from psychology to sleep research [32, 42]. Various ER solutions have been proposed for resolving familial networks. Kouki et al. [40] proposed a collective ER approach for building familial networks based on probabilistic soft logic. Furthermore, Antonie et al. [2] and Folkman et al. [24] proposed supervised ER systems specific for genealogical studies. However, the limited ground truth data available in most genealogical studies [22, 58] prevents supervised learning systems to be applied, while manually generating ground truth data is time and labour intensive [11], and generally a large number of true matches and non-matches are required to obtain high quality ER results [4, 20].

Query time ER has been discussed in the literature where the focus for execution time is considered similarly important to that of accuracy. Bhattacharya and Getoor [7] pioneered query time collective ER by identifying and resolving only the records that are most useful for a query. While this approach provided a promising solution, because it only considered a small neighbourhood of a query record, it lacks the relational information propagation as well as the exploitation of the ambiguity we incorporate in our application. Christen et al. [13] and Ramadan et al. [56] proposed indexing techniques to match query records

with entities from a real-world database in real-time. In our application we employ the similarity-aware index proposed in [13] to speed-up the retrieval of approximate matches in the query processing step.

Graph-based ER approaches are an improvement over pairwise classification approaches that do not exploit relational information [26] in the ER process. Kalashnikov et al. [35] proposed a random walk based reference disambiguation approach to identify the entity of each record, while Dong et al. [19] employed a dependency graph-based approach to propagate link decisions in the ER process which is the closest to our approach in *SNAPS*. Bhattacharya and Getoor [6] used relational information between different types of entities by employing an iterative cluster merging process using a relationship graph. However, unlike *SNAPS*, none of these approaches perform propagation of link decisions to account for dynamically changing attribute values in a context with ambiguous records, relationships with different constraints that can change over time, and partial match groups.

Recently, more focus has been given to supervised ER [38, 51] since these approaches provide promising results when sufficient ground truth data are available. *Magellan* is one of those frameworks that supports end-to-end ER [38]. More recent work includes deep learning approaches [9, 44, 48] to resolve entities. The main drawback of these approaches is the difficulty to obtain ground truth data. For example, in our *SNAPS* application we cannot perform manual linkage to obtain ground truth data for the full DS database we described in Section 2 due to privacy and confidentiality regulations of person data [14]. Similar challenges arise when applying semi-supervised ER techniques such as active learning [15, 16, 36, 55] that query external sources to resolve challenging training cases, or crowd-based approaches [1, 27, 63] that employ hybrid machine and human-based systems for resolving entities.

To the best of our knowledge, none of the existing solutions provide an end-to-end system for family pedigree search that includes an online query-time interface and an offline unsupervised ER component which addresses the challenges of ambiguity, temporal constraints, and relationship aspects to link records.

12 CONCLUSIONS AND FUTURE WORK

We have presented the *SNAPS* application that combines an unsupervised graph-based entity resolution (ER) approach with a query method that allows for approximate string matching. Our application is aimed at providing support for the Genetics Genealogy Team of Public Health Scotland, who currently manually query Scottish birth, death, and marriage certificates to compile family pedigrees (family trees) for patients who have familial cancer or other inherited genetic conditions. Our application will automate much of this time consuming and labour intensive process and provide high quality family pedigrees in a matter of a few seconds. Future expansions can include assistance for general practitioners to efficiently obtain the family health histories of their patients.

While aimed at this specific health application in Scotland, the underlying graph-based ER techniques and the query processing and ranking methods are general and can be applied on any data sets that contain birth, death, and marriage certificates.

Privacy regulations currently prevent remote access to the Digitising Scotland database covering the full population of Scotland. Therefore, to assess the quality of our ER and query approaches, we have evaluated *SNAPS* on two smaller historical data sets from the Isle of Skye and Kilmarnock, respectively, for which partial ground truth are available [58]. We also used the historical BHIC data set [10] for scalability experiments since this is a considerably larger data set compared to the two small Scottish data sets. Our unsupervised ER approach is able to obtain precision up-to 98.7%, recall up-to 94.7%, and F^*-measure results up-to 93.6%, respectively, while query processing and extracting family pedigrees is accomplished on average in less than 1.4 and 0.8 seconds, respectively.

We also described an approach to anonymise a graph data set of birth, death, and marriage certificates [45]. This allows us to provide a publicly available version of our application that can be used for training and educational purposes on data that have similar characteristics as the sensitive real data they are based on. The anonymisation approach replaces first names and surnames in such ways that the similarities between groups of names are maintained, while all years (such as of births, deaths, or marriages) are shifted by a certain (kept secret) offset to maintain the temporal distances between vital events. Rare causes of death are replaced by a frequent cause that is most similar.

As future work, we aim to improve the scalability of our application with the aim to make it usable with a population-scale database of millions of records, and to incorporate geographical distances into the query process (to allow users to limit searches to certain geographical regions) by geocoding the addresses available on certificates [37]. We also plan to investigate how census data can be incorporated into our ER techniques to improve linkage quality.

A broader user study, including on larger databases, is planned for later in 2022 to improve both the web interface as well as the generation of family pedigrees. We aim to incorporate interactive access to the actual original certificates (digitised images) held by the National Records of Scotland, and incorporate feedback from domain experts on correctly and wrongly generated family trees. Such feedback can then be employed within an active learning based framework to improve the quality of generated links [15, 55, 62].

ACKNOWLEDGEMENTS

We like to thank Thilina Ranbaduge for help in developing the web interface, Alice Reid and Ros Davis (University of Cambridge) for their work on the Isle of Skye and Kilmarnock data sets, and Kellas Campbell (University of Edinburgh) for providing the statistics of the Digitising Scotland database. This work was partially supported by ESRC grants ES/K00574X/2 Digitising Scotland and ES/L007487/1 Administrative Data Research Centre Scotland.

REFERENCES

[1] Asma Abboura, Soror Sahrl, Mourad Ouziri, and Salima Benbernou. 2015. CrowdMD: Crowdsourcing-based Approach for Deduplication. In *IEEE BigData*. IEEE, Santa Clara, USA, 2621–2627.
[2] 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, 1 (2014), 129–146.
[3] Martha J Bailey, Connor Cole, Morgan Henderson, and Catherine Massey. 2020. How well do automated linking methods perform? Lessons from US historical data. *Journal of Economic Literature* 58, 4 (2020), 997–1044.
[4] Nils Barlaug and Jon Atle Gulla. 2021. Neural Networks for Entity Matching: A Survey. *ACM TKDD* 15, 3 (2021), 1–37.
[5] Ricardo J. Bayardo, Yiming Ma, and Ramakrishnan Srikant. 2007. Scaling up all pairs similarity search. In *WWW*. ACM, Banff, Canada, 131–140.
[6] Indrajit Bhattacharya and Lise Getoor. 2007. Collective Entity Resolution in Relational Data. *ACM TKDD* 1, 1 (2007), 5–es.
[7] Indrajit Bhattacharya and Lise Getoor. 2007. Query-time entity resolution. *JAIR* 30 (2007), 621–657.

[8] David H Brewster, Alison Fordyce, and Roger J Black. 2004. Impact of a cancer registry-based genealogy service to support clinical genetics services. *Familial Cancer* 3, 2 (2004), 139–141.

[9] Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures-a step forward in data integration. In *EDBT*. Copenhagen, Denmark.

[10] Brabant Historical Information Center. 2021. Genealogie. Retrieved June 29, 2021 from https://opendata.picturae.com/organization/bhic

[11] Peter Christen. 2012. *Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer, Heidelberg.

[12] Peter Christen. 2016. Application of Advanced Record Linkage Techniques for Complex Population Reconstruction. *arXiv preprint arXiv:1612.04286* (2016).

[13] Peter Christen, Ross Gayler, and David Hawking. 2009. Similarity-Aware Indexing for Real-Time Entity Resolution. In *CIKM*. ACM, Hong Kong, 1565–1568.

[14] Peter Christen, Thilina Ranbaduge, and Rainer Schnell. 2020. *Linking Sensitive Data.* Springer, Heidelberg.

[15] Peter Christen, Dinusha Vatsalan, and Qing Wang. 2015. Efficient entity resolution with adaptive and interactive training data selection. In *ICDM*. IEEE, Atlantic City, 727–732.

[16] Victor Christen, Peter Christen, and Erhard Rahm. 2019. Informativeness-Based Active Learning for Entity Resolution. In *Workshop on Data Integration and Applications, held at PKDD/ECML*. Springer, Würzburg, 125–141.

[17] AnHai Doan, Pradap Konda, Paul Suganthan GC, Yash Govind, Derek Paulsen, Kaushik Chandrasekhar, et al. 2020. Magellan: toward building ecosystems of entity matching solutions. *Commun. ACM* 63, 8 (2020), 83–91.

[18] Josep Domingo-Ferrer, David Sánchez, and Jordi Soria-Comas. 2016. *Database Anonymization: Privacy Models, Data Utility, and Microaggregation-based Inter-model Connections.* Morgan and Claypool Publishers.

[19] Xin L. Dong, Alon Halevy, and Jayant Madhavan. 2005. Reference reconciliation in complex information spaces. In *SIGMOD*. ACM, Baltimore, 85–96.

[20] Xin L. Dong and Theodoros Rekatsinas. 2018. Data integration and machine learning: A natural synergy. *VLDB Endowment* 11, 12 (2018), 2094–2097.

[21] Xin L. Dong and Divesh Srivastava. 2015. *Big Data Integration.* Morgan and Claypool Publishers.

[22] Sören Edvinsson and Elisabeth Engberg. 2020. A Database for the Future: Major Contributions from 47 Years of Database Development and Research at the Demographic Data Base. *Historical Life Course Studies* (2020).

[23] Ivan P. Fellegi and Alan B. Sunter. 1969. A Theory for Record Linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.

[24] Tyler Folkman, Rey Furner, and Drew Pearson. 2018. GenERes: A Genealogical Entity Resolution System. In *DINA workshop held at ICDM*. IEEE, Singapore, 495–501.

[25] Zhichun Fu, Peter Christen, and Jun Zhou. 2014. A graph matching method for historical census household linkage. In *PAKDD*. Springer, Tainan, 485–496.

[26] Lise Getoor and Ashwin Machanavajjhala. 2013. Entity resolution for Big data. In *SIGKDD*. ACM, Chicago, 1527–1527.

[27] Yash Govind, Erik Paulson, Palaniappan Nagarajan, Paul Suganthan G. C., AnHai Doan, Youngchoon Park, Glenn M. Fung, Devin Conathan, Marshall Carter, and Mingju Sun. 2018. Cloudmatcher: A Hands-off Cloud/Crowd Service for Entity Matching. *VLDB Endowment* 11, 12 (2018), 2042–2045.

[28] Naomi C Hamm, Amani F Hamad, Elizabeth Wall-Wieler, Leslie L Roos, Oleguer Plana-Ripoll, and Lisa M Lix. 2021. Multigenerational Health Research using Population-Based Linked Databases: An International Review. *International Journal of Population Data Science* 6, 1 (2021).

[29] David J. Hand and Peter Christen. 2018. A note on using the F-measure for evaluating record linkage algorithms. *Stats and Comp* 28, 3 (2018), 539–547.

[30] David J. Hand, Peter Christen, and Nishadi Kirielle. 2021. F*: An Interpretable Transformation of the F-measure. *Machine Learning* 110, 3 (2021), 451–456.

[31] Oktie Hassanzadeh, Fei Chiang, Hyun Chul Lee, and Renée J Miller. 2009. Framework for evaluating clustering algorithms in duplicate detection. *VLDB Endowment* 2, 1 (2009), 1282–1293.

[32] William D. Hill, Ruben C Arslan, Charley Xia, Michelle Luciano, Carmen Amador, et al. 2018. Genomic analysis of family data reveals additional genetic effects on intelligence and personality. *Molecular Psychiatry* 23, 12 (2018), 2347–2362.

[33] Yichen Hu, Qing Wang, Dinusha Vatsalan, and Peter Christen. 2017. Improving temporal record linkage using regression classification. In *PAKDD*. Springer, Jeju, 561–573.

[34] ISD Scottish Genetics Genealogy Service. 2018. Annual Report 2018. https://www.isdscotland.org/Health-Topics/Cancer/Genetics-Genealogy/

[35] Dmitri V. Kalashnikov and Sharad Mehrotra. 2006. Domain-independent data cleaning via analysis of entity-relationship graph. *TODS* 31, 2 (2006), 716–767.

[36] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource Deep Entity Resolution with Transfer and Active Learning. In *Annual Meeting of the ACL*. ACL, Florence, 5851–5861.

[37] Nishadi Kirielle, Peter Christen, and Thilina Ranbaduge. 2019. Outlier Detection Based Accurate Geocoding of Historical Addresses. In *AusDM*. Springer, Adelaide, 41–53.

[38] Pradap Konda, Sanjib Das, Paul Suganthan GC, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, et al. 2016. Magellan: Toward building entity matching management systems. *VLDB Endowment* 9, 12 (2016), 1197–1208.

[39] Pradap Konda, Sanjay Subramanian Seshadri, Elan Segarra, Brent Hueth, and AnHai Doan. 2019. Executing Entity Matching End to End: A Case Study. In *EDBT*. Lisbon, 489–500.

[40] Pigi Kouki, Jay Pujara, Christopher Marcum, Laura Koehly, and Lise Getoor. 2019. Collective entity resolution in multi-relational familial networks. *KAIS* 61, 3 (2019), 1547–1581.

[41] Pei Li, Xin L. Dong, Andrea Maurino, and Divesh Srivastava. 2011. Linking temporal records. *VLDB Endowment* 4, 11 (2011), 956–967.

[42] AK Licis, DM Desruisseau, KA Yamada, SP Duntley, and CA Gurnett. 2011. Novel genetic findings in an extended family pedigree with sleepwalking. *Neurology* 76, 1 (2011), 49–52.

[43] Michael Loster, Ioannis Koumarelas, and Felix Naumann. 2021. Knowledge Transfer for Entity Resolution with Siamese Neural Networks. *JDIQ* 13, 1 (2021), 1–25.

[44] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, et al. 2018. Deep learning for entity matching: A design space exploration. In *SIGMOD*. ACM, Houston, 19–34.

[45] Charini Nanayakkara, Peter Christen, and Thilina Ranbaduge. 2020. An Anonymiser Tool for Sensitive Graph Data. In *CIKM EYRE workshop*. Galway.

[46] Felix Naumann and Melanie Herschel. 2010. *An Introduction to Duplicate Detection.* Morgan and Claypool Publishers.

[47] Gonzalo Navarro. 2001. A guided tour to approximate string matching. *Comput. Surveys* 33, 1 (2001), 31–88.

[48] Hao Nie, Xianpei Han, Ben He, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 2019. Deep Sequence-to-Sequence Entity Matching for Heterogeneous Entity Resolution. In *CIKM*. ACM, Beijing, China, 629–638.

[49] Byung-Won On, N. Koudas, Dongwon Lee, and D. Srivastava. 2007. Group Linkage. In *IEEE ICDE*. IEEE, Istanbul, 496–505.

[50] Toan C. Ong, Michael V. Mannino, Lisa M. Schilling, and Michael G. Kahn. 2014. Improving Record Linkage Performance in the Presence of Missing Linkage Data. *Journal of Biomedical Informatics* 52 (2014), 43–54.

[51] Matteo Paganelli, Francesco Del Buono, Pevarello Marco, Francesco Guerra, and Maurizio Vincini. 2021. Automated Machine Learning for Entity Matching Tasks. In *EDBT*. Nicosia, 325–330.

[52] Fabian Panse, André Düjon, Wolfram Wingerath, and Benjamin Wollmer. 2021. Generating Realistic Test Datasets for Duplicate Detection at Scale Using Historical Voter Data. In *EDBT*. 570–581.

[53] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. 2021. The Four Generations of Entity Resolution. *Synthesis Lectures on Data Management* 16, 2 (2021), 1–170.

[54] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and Filtering Techniques for Entity Resolution: A Survey. *Comput. Surveys* 53, 2 (2020), 1–42.

[55] Kun Qian, Lucian Popa, and Prithviraj Sen. 2017. Active learning for large-scale entity resolution. In *CIKM*. ACM, Singapore, 1379–1388.

[56] Banda Ramadan, Peter Christen, Huizhi Liang, and Ross W Gayler. 2015. Dynamic sorted neighborhood indexing for real-time entity resolution. *JDIQ* 6, 4 (2015), 1–29.

[57] Sean M Randall, James H Boyd, Anna M Ferrante, Jacqueline K Bauer, and James B Semmens. 2014. Use of graph theory measures to identify errors in record linkage. *Computer Methods and Programs in Biomedicine* 115, 2 (2014), 55–63.

[58] Alice Reid, Ros Davies, and Eilidh Garrett. 2002. Nineteenth-century Scottish demography from linked censuses and civil registers: A 'sets of related individuals' approach. *History and Computing* 14, 1–2 (2002), 61–86.

[59] Alice Reid, EM Garrett, Chris Dibben, and Lee Williamson. 2016. Gender specific mortality trends over the epidemiological transition: a view from the British mainland 1850-2000. (2016), 73–88.

[60] Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation* 60, 5 (2004), 503–520.

[61] Pierangela Samarati. 2001. Protecting respondents identities in microdata release. *IEEE TKDE* 13, 6 (2001), 1010–1027.

[62] Yufei Tao. 2018. Entity matching with active monotone classification. In *SIGMOD-SIGACT-SIGAI PODS*. ACM, Houston, 49–62.

[63] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *VLDB Endowment* 5, 11 (2012), 1483–1494.

[64] Li-E. Wang and Xianxian Li. 2018. A graph-based multifold model for anonymizing data with attributes of multiple types. *CaS* 72 (2018), 122–135.

[65] Chen Zhao and Yeye He. 2019. Auto-EM: End-to-end Fuzzy Entity-Matching using Pre-trained Deep Models and Transfer Learning. In *WWW*. ACM, San Francisco, USA, 2413–2424.

[66] Bin Zhou, Jian Pei, and WoShun Luk. 2008. A Brief Survey on Anonymization Techniques for Privacy Preserving Publishing of Social Network Data. *SIGKDD Explor. Newsl.* 10, 2 (Dec. 2008), 12–22.