Effective Record Linkage Techniques for Complex Population Data

Charini Nanayakkara

A thesis submitted for the degree of
Doctor of Philosophy in Computer Science
The Australian National University

April 2022

Except where otherwise indicated, this thesis is my own original work.

Charini Nanayakkara
28 April 2022

I dedicate this thesis to the four pillars in my life, my loving parents, husband and my only sibling.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisory panel for the unwavering support they have offered me throughout my doctoral studies. Having an amazing individual as Professor Peter Christen (The Australian National University, Canberra, Australia) as my primary supervisor has immensely contributed to the successful completion of this thesis and related publications. With more than twenty years of experience in the field of academia, his guidance and support have always been admired by all his students. Above all, I am truly grateful to him for genuinely caring for his students' well-being and their future, and for constantly making time for each and every one of us such that we never deviated from track. I would like to acknowledge the contribution of my co-supervisor Dr. Thilina Ranbaduge (Data 61 – CSIRO, Canberra, Australia) for his valuable support and comments, and for being an inspiration to all of us by completing a PhD with several A-grade publications, while raising a young child. Many thanks goes to Dr. Eilidh Garrett (University of Edinburgh, UK) for her insights on population data as an experienced demographer.

I express my heartfelt gratitude to the two people who gave me life and nurtured me, my beloved parents Jayanthi and Tissa Nanayakkara. The many sacrifices they made in life for caring for my aunt in ill health, and for my grandmother who was suffering from Alzheimer's disease taught me what it truly means to be resilient despite all odds. The strength they found to do so much for others even when the world did not necessarily applaud their sacrifices was astounding, and still does not fail to inspire me. Actions speak louder than words, and I am indebted to my parents for guiding me through example and not merely by advice. Simply observing my parents helped me to find the courage to refrain from wrongdoings, and steer in the correct path. All the good that I have done and the achievements I have made are a tribute to my parents; for having raised two girls while being caregivers, despite them not being very wealthy. They will always be remembered as the two people who laid the foundation to all my good traits and why I strive to be a person of value to society.

I am truly grateful to my loving husband Yasith Kanchana Lokuge for his unwavering love, support and encouragement throughout my life as a graduate student and beyond. He sacrificed his well-paid, hard-earned job as a Software Engineer in Sri Lanka to help me pursue my dreams despite knowing that it is extremely difficult for immigrants to secure jobs in Canberra, Australia. Undeterred by all these challenges, Yasith was perseverant in searching for opportunities in the industry which eventually led to him securing a position in an Australian software development company even prior to us arriving in Canberra. I am awed by his kind heart and

the love he showers upon me every single day. I take this opportunity to thank my loving sister Dulanji Nanayakkara who has always believed in my potential to reach great heights, and for constantly being my cheerleader. My parents, husband and sister are my greatest pillars of strength, and I am truly grateful to them for being who they are.

My heartfelt gratitude goes out to my friends who were with me through thick and thin during my childhood and doctoral studies. I appreciate each one of them for being exemplary individuals whom I could learn something from, and for their friendship. My wonderful, generous relatives, and my in-laws who have helped me in numerous ways are acknowledged with much love for their support, blessings and encouragement in this journey.

I express my gratitude to Dr. Amitha Caldera and Dr. Shiromi Arunathilake from the University of Colombo School of Computing, Sri Lanka, and Mr. Viraj Brian Wijesuriya who is reading for his PhD at the University of Oxford, for their support with obtaining a scholarship at the ANU. Their immense support has helped me secure scholarships for which there is high demand and competition among students from around the globe.

I am truly grateful to the School of Computing at the Australian National University (ANU) for offering me two scholarships for international students (a fee waiver and a stipend). Hailing from a developing country such as Sri Lanka, and not being extremely wealthy as to be able to afford a foreign education, pursuing doctoral studies would have never become a reality for me without such scholarships. I express my sincere gratitude to the non-academic staff at the ANU as well, for their largely unapplauded invaluable services, without which the academic work would not have progressed smoothly. The workshops and support provided by the staff of the ANU Academic Skills is greatly valued and acknowledged since those (especially programs such as the Thesis Boot Camp led by Victoria and Inger) have been immensely helpful in organising and improving my work as a graduate student.

Last, but not least, I would like to express my heartfelt gratitude to my Alma maters from my motherland, the University of Colombo School of Computing and Musaeus College, Colombo, Sri Lanka, for moulding me into the person I am today. The education and values I have learnt from these institutions have helped me become a better individual and succeed in life. I pay tribute to every teacher and lecturer who has taught me up to now, and has been influential in my life. I sincerely thank the people in my beautiful country, Sri Lanka, for having paid for my education with their taxes, and to the amazing free education system in my motherland. Free education in Sri Lanka, as initiated by Dr. C.W.W Kannangara, paves the path for Sri Lankan children to receive quality education and reach great heights, regardless of their background and wealth, and I would forever owe my country for this reason. I express my immense gratitude and dedicate this thesis to all who have helped me in the slightest manner during my doctoral studies, and in all aspects of life.

Abstract

Real-world data sets are generally of limited value when analysed on their own, whereas the true potential of data can be exploited only when two or more data sets are linked to analyse patterns across records. A classic example is the need for merging medical records with travel data for effective surveillance and management of pandemics such as COVID-19 by tracing points of contacts of infected individuals. Therefore, Record Linkage (RL), which is the process of identifying records that refer to the same entity, is an area of data science that is of paramount importance in the quest for making informed decisions based on the plethora of information available in the modern world.

Two of the primary concerns of RL are obtaining linkage results of high quality, and maximising efficiency. Furthermore, the lack of ground-truth data in the form of known matches and non-matches, and the privacy concerns involved in linking sensitive data have hindered the application of RL in real-world projects. In traditional RL, methods such as blocking and indexing are generally applied to improve efficiency by reducing the number of record pairs that need to be compared. Once the record pairs retained from blocking are compared, certain classification methods are employed to separate matches from non-matches. Thus, the general RL process comprises of blocking, comparison, classification, and finally evaluation to assess how well a linkage program has performed.

In this thesis we initially provide a holistic understanding of the background of RL, and then conduct an extensive literature review of the state-of-the-art techniques applied in RL to identify current research gaps. Next, we present our initial contribution of incorporating data characteristics, such as temporal and geographic information with unsupervised clustering, which achieves significant improvements in precision (more than 16%), at the cost of minor reduction in recall (less than 2.5%) when they are applied on real-world data sets compared to using regular unsupervised clustering.

We then present a novel active learning-based method to filter record pairs subsequent to the record pair comparison step to improve the efficiency of the RL process. Furthermore, we develop a novel active learning-based classification technique for RL which allows to obtain high quality linkage results with limited ground-truth data. Even though semi-supervised learning techniques such as active learning methods have already been proposed in the context of RL, this is a relatively novel paradigm which is worthy of further exploration. We experimentally show more than 35% improvement in clustering efficiency with the application of our proposed filtering approach; and linkage quality on par with or exceeding existing active learning-based classification methods, compared to our active learning-based classification technique.

Existing RL evaluation measures such as precision and recall evaluate the classification outcome of record pairs, which can cause ambiguity when applied in the group RL context. We therefore propose a more robust RL evaluation measure which evaluates linkage quality based on how individual records have been assigned to clusters rather than considering record pairs. Next, we propose a novel graph anonymisation technique that extends the literature by introducing methods of anonymising data to be linked in a human interpretable manner, without compromising structure and interpretability of the data as with existing state-of-the-art anonymisation approaches. We experimentally show how the similarity distributions are maintained in anonymised and original sensitive data sets when our anonymisation technique is applied, which attests to its ability to maintain the structure of the original data. We finally conduct an empirical evaluation of our proposed techniques and show how they outperform existing RL methods.

List of Contributions

Major Contributions

1. **Charini Nanayakkara**, Peter Christen, and Thilina Ranbaduge. *Temporal Graph-based Clustering for Historical Record Linkage*. International Workshop on Mining and Learning with Graphs (MLG) Workshop, held at ACM SIGKDD, London (2018).

2. **Charini Nanayakkara**, Peter Christen, and Thilina Ranbaduge. *Robust Temporal Graph Clustering for Group Record Linkage*. Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Macau (2019).

3. **Charini Nanayakkara**, Peter Christen, and Thilina Ranbaduge and Eilidh Garrett. *Evaluation Measure for Group-based Record Linkage*. International Journal of Population Data Science, 4(1) (2020).

4. **Charini Nanayakkara**, Peter Christen, and Thilina Ranbaduge. *An Anonymiser Tool for Sensitive Graph Data*. International Workshop on Entity Retrieval and Learning (EYRE) co-located with CIKM, Galway, Ireland (2020).

5. **Charini Nanayakkara**, Peter Christen, and Thilina Ranbaduge. *Active Learning Based Similarity Filtering for Efficient and Effective Record Linkage*. Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Delhi (2021).

Minor Contributions

1. **Charini Nanayakkara**, Peter Christen, and Thilina Ranbaduge. *Historical Record Linkage* (some of the clustering algorithms related to Chapter 4). June 2020, https://dmm.anu.edu.au/HISTRL/.

2. **Charini Nanayakkara**, Peter Christen, and Thilina Ranbaduge. *DOYEN - **D**ata Generator and An**o**n**y**miser for S**en**sitive Graph Data*. December 2020, https://dmm.anu.edu.au/doyen/.

3. **Charini Nanayakkara**, Nishadi Kirielle, and Peter Christen. *SNAPS - **S**cotla**n**d F**a**mily **P**edigree **S**earch Tool*. June 2021, https://dmm.anu.edu.au/SNAPS/.

4. Nishadi Kirielle, **Charini Nanayakkara**, Peter Christen, Chris Dibben, Lee Williamson, Eilidh Garrett, and Clair Manson. *Unsupervised Graph-based Entity Resolution for Accurate and Efficient Family Pedigree Search*. International Conference on Extending Database Technology (EDBT), Edinburgh, UK (2022).

Contents

List of Figures

List of Tables

Notations and Terminology

Notations

$\mathbf{A}$, a	List of attributes, an attribute
$\mathbf{C}$, $\mathbf{c}$	List of clusters, a cluster of records
C	A classifier
$\mathbf{D}$	A data set
$\mathbf{E}$	A set of edges in a graph data structure
$\mathbf{G}$	Graph data structure where $\mathbf{G} = (\mathbf{V}, \mathbf{E})$
$\mathbf{M}$	A set of classified matches
$\mathbf{N}$	A set of classified non-matches
$P(.)$	A probability function
p	A probability value
$r, r.id, r.t$	A record, a record identifier and a record time-stamp
$\mathbf{S}$	A list of similarity functions
$\mathbf{s}$	A similarity vector
$S_a(.,.)$	A similarity function used to calculate the similarity between record pairs on attribute a, where $S_a \in \mathbf{S}$
s	An overall similarity value
$\mathbf{T}$	List of temporal constraints
$\mathbf{V}$	A set of vertices (nodes) in a graph data structure
$\mathbf{w}$	List of weights
β	A budget value
Δt	A time difference
δ	A threshold value

Terminology

AUPRC	Area Under the Precision-Recall Curve
COVID-19	Corona Virus Disease
DL	Deep Learning
DOYEN	Data generator and anOnYmiser for sENsitive graph data [124]
F	F-measure [85]
F^*	F-star measure [83]
FN	False Negatives
FP	False Positives
IoS	Isle of Skye birth data set
IPUMS	Integrated Public Use Microdata Series [152]
Kilm	Kilmarnock birth data set
LIFE-M	Longitudinal, Intergenerational Family Electronic Micro-Database Project [7]
LSH	Locality Sensitive Hashing
ML	Machine Learning
MPC	Minnesota Population Centre
NCVR	North Carolina Voter Registration data set
NT	Non-Temporal
P	Precision [85]
PPRL	Privacy-Preserving Record Linkage
R	Recall [85]
RL	Record Linkage
T	Temporal
TN	True Negatives
TP	True Positives
UK	Synthetic birth data set from the United Kingdom

Certificate	A document which contains information about a life event such as a birth, death or marriage, and contains personal details of a single or several related entities.
Record	A collection of attribute values describing a single entity, where one or more records can be retrieved from a certificate.
Data set	A collection of records.

Chapter 1

Introduction

In this chapter, we provide an introduction to Record Linkage (RL) and highlight our contributions to this field of study. We initially provide an overview of RL and its numerous application areas in Section 1.1. In Section 1.2 we then introduce the research problems, whereas in Section 1.3 we highlight our aims and objectives by providing feasible solutions to these problems. In Section 1.4 we describe our contributions, whereas in Section 1.5 we define the scope of our thesis. Finally, in Sections 1.6 and 1.7, we describe the methodology we followed in conducting our research, and provide an outline of how the rest of this thesis is organised.

1.1 Overview of Record Linkage

Record linkage (RL), also known as entity resolution, data matching, or object identification, is the process of identifying sets or groups of records which refer to the same entity, within one or across two or more data sets [35]. The plethora of information available in real-world data sets can be effectively manipulated subsequent to the application of RL methods. For instance, RL techniques are often used to deduplicate repeated records in data sets, since such repetition is detrimental to data set quality, storage, and data analysis tasks. Linking records across data sets is also useful to obtain a holistic view of data. For example, the overall health of a patient can be analysed by linking his/her records residing in different hospital data sets. A non-exhaustive list of areas where RL methods are applied is as follows.

1. **The Health Sector**: One of the primary areas of RL application is the health domain [59, 131] because it is impossible to infer certain medical conditions and patterns of deceases from a single health data set alone. The value of applying RL techniques to contain pandemic situations by integrating surveillance data was profoundly realised during the current COVID-19 outbreak [118]. Records from different health data sets may need to be linked in order to identify interdependencies among certain illnesses, or to identify medication that can result in certain side effects. Furthermore, medical records may be linked with census and other vital records (such as birth and death) to find and trace roots of hereditary illnesses [163], as done in the early *Oxford Record Linkage Study* which used RL techniques to link and analyse birth, death, and hospital data [74].

2. **Fraud Detection**: RL is an integral part of modern fraud detection tools and in identifying identity theft [29], where two or more data sets are linked to identify unusual patterns. Verification of identity is crucial in modern times where travel across countries is approved based on electronically provided information by individuals. By applying RL techniques, it is possible to validate such personal information by linking them to several external data sets containing accurate information. A successful application of RL for fraud detection was mentioned in [93], where a data set of airplane pilots licensed by the US Federal Aviation Administration (FAA) was linked with a data set consisting of individuals receiving disability payments from the Social Security Administration. Forty pilots with their record appearing in both data sets were arrested for either deceiving the FAA or for unlawfully receiving benefits, whereas the licences of fourteen pilots were revoked.

3. **Genealogical and Demographic Studies**: Among the numerous application areas of RL, genealogical and demographic studies conducted based on historical data, such as censuses and vital records (birth, marriage and death certificates), are of considerable importance [149]. Since longitudinal data such as vital records generally consist of several records representing the same individual (for example, an individual can appear as a baby in a birth certificate and later as the bride in a marriage certificate), it is possible to link these records to construct an entire population. Such a population data set can pave the way to a multitude of studies in the health and social sciences that currently are not feasible on individual data sets [17, 106].

   Historical data sets not only provide information about individuals but also about groups of individuals, such as families and households. Application of RL techniques for identifying groups of entities, known as *group linkage* [134], has received considerable attention in recent years, due to the numerous complex studies that linked groups of records can facilitate [43, 68]. The identification of relationships between individuals can enrich 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 [17].

4. **Online Stores**: RL techniques are widely applied to organise products in online stores [54]. Due to the possibility of the same product being referred to by various names, and different products being identified by the same name, RL methods need to be applied to distinguish among unique products. Ambiguity among product names is further emphasised when individual traders are allowed to advertise and sell their items via online marketplaces. The freedom to name products as traders wish may even be exploited to increase the chances of potential buyers viewing a product, by deceitfully assigning a name that is similar to a commodity in high demand [146]. The reputation of online stores would be at stake unless RL techniques are applied to identify and prevent such malicious acts.

Even though the importance of RL is indisputable, the inherent challenges in RL [35, 40, 56], including the difficulty to scale linking larger data sets, privacy concerns, and the lack of ground-truth data for evaluation, have greatly hindered the development of commercial applications in this field of study. These challenges largely increase when linking population data, due to low data quality (transcription errors and misspellings of hand-written forms), sensitivity of personal data, missing data, and lack of ground-truth (which is difficult and expensive to obtain). Therefore, the vast majority of research in the area of RL has concentrated on either exploiting the structure in such data sets (such as households and families) and developed group linkage methods [43, 68, 70, 134] or collective techniques [15, 37, 147]. In the following section, we identify the challenges in RL which we aim to address in our thesis.

1.2 Research Problems

Linking records within or across data sets in the absence of a unique identifier is a non-trivial task, especially due to the large number of record pairs that need to be compared to identify records that refer to the same entity [35]. As we discuss in Section 2.3, the number of naïve record pair comparisons grows quadratically with the size of the data sets being linked, thus resulting in the inability to conduct RL in feasible time. For example, the total number of record pair comparisons across two data sets with a thousand records in each is a million. Thus, a massive number of record pair comparisons would need to be conducted across real-world data sets which often contain many millions of records. As we emphasise in Section 1.5, we will be focusing on population data throughout our thesis. The scalability of RL approaches applied on population data sets is a significant concern, since such data sets usually contain millions of records.

As we show in Section 2.5, population data is generally of low quality and extremely skewed with regard to frequency distribution of values (attribute values such as names are usually shared by many records belonging to different entities), which adversely affects the capability of linking these data using existing RL techniques. Furthermore, longitudinal population data such as censuses and civil registries are highly dependent on time (for example, the death record of a person can occur only after his/her birth record, or the biological time intervals within which a mother can or cannot deliver two children), an aspect which is incorporated only in a limited number of customised rule-based languages for RL [92], but not in general RL techniques.

We have identified the following research problems pertaining to linking population data, which we aim to address in our thesis.

1. **Data Set Sizes**: As we previously highlighted, it is infeasible to conduct RL on data sets containing a large number of records with naïve pairwise comparisons of records. Therefore, techniques such as blocking or indexing [35, 138] are often applied as a step in RL to reduce the comparison space. Blocking and

indexing are techniques where records that are likely to refer to the same entity are grouped into the same blocks, and records that are likely to refer to different entities are assigned to different blocks. This approach reduces the number of record pair comparisons, thereby improving the efficiency of the overall record linkage process. Additional meta-blocking [61] methods are sometimes applied to further reduce redundant and superfluous record pair comparisons [138].

However, existing blocking, indexing, and meta-blocking techniques, when not customised to the linkage problem at hand, are often inadequate for achieving a desirable run-time for the overall RL process due to a considerable number of record pair comparisons resulting from blocking, which mostly consist of true non-matching pairs [35, 138]. Therefore, more complex filtering techniques need to be developed to conduct RL efficiently.

2. **Data Quality**: A lack of data quality is a problem with the majority of real-world data sets, whereas population data sets are often more susceptible to this issue than other types of data. The quality of population data can be lacking due to having different variations for attribute values, such as for names and addresses. The quality of data is often compromised at transcription as well, such as when digitising hand-written birth, death, and marriage certificates [35]. Furthermore, we often encounter highly skewed name distributions in populations where many people share similar names, and few unique names appear. Name and address values can also change over time due to marriage or change of residence. Such imbalanced attribute value distributions, spelling variations, use of name abbreviations, potential attribute value changes, and errors introduced during data entry and transcription, can result in low data quality. Therefore, methods which are capable of achieving high linkage quality in the presence of low data quality need to be developed.

3. **Data Privacy**: When applying RL on population data, it is important to consider privacy when publishing linkage results [40]. Privacy-Preserving Record Linkage (PPRL) is a research area which is beyond our scope as we highlight in Section 1.5. However, we need to be mindful of the information we publish when making linkage results available to a research audience. Therefore, it is important to develop new anonymisation methods that can ensure the privacy of data sets utilised for conducting linkage experiments, while ensuring that data set anonymisation does not compromise the efficiency and effectiveness of the RL methods applied.

4. **Lack of Ground-truth Data**: Ground-truth data in the form of known matches (record pairs that refer to the same entity) and non-matches (record pairs that refer to different entities) are often difficult to obtain for RL projects [35]. This is due to the infeasibility of manually classifying the potentially large number of record pairs across data sets as true matches and non-matches. Even though both supervised and unsupervised techniques are used for linking records [35], applying supervised classification approaches for RL can be difficult because

ground-truth data is required for training supervised algorithms. Hence, more advanced unsupervised and semi-supervised algorithms need to be developed for linkage tasks.

5. **Evaluating Linkage Results**: The performance of linkage techniques are often assessed using precision, recall, and the F-measure [35] as we elaborate in Chapter 2. These measures were originally developed to assess the quality of classification methods in machine learning. In their recent research work, Hand and Christen [85] showed why it may be inappropriate to use the F-measure to compare linkage techniques despite its wide usage in the area of RL. It is therefore important to further explore alternative methods to facilitate effective and comparable evaluation of different RL techniques.

1.3 Aim and Objectives

The aim of our thesis is to address and propose feasible solutions for the research problems we have highlighted in Section 1.2 within the scope we define in Section 1.5. More precisely, *our aim is to develop efficient and effective algorithms for RL, while ensuring that the privacy of the resulting linked data set is preserved, and that linkage quality can be assessed in a robust manner*. In order to achieve our research aim, we address the following objectives in our thesis.

1. **Improve Record Linkage Efficiency with Record Pair Filtering**: Conducting RL in an efficient manner is a significant challenge due to the large sizes of modern data sets. As we highlighted in Section 1.2, a large number of record pairs are often retained even after applying blocking, indexing, and meta-blocking techniques to reduce record pair comparisons. Applying complex supervised and unsupervised techniques on a large number of compared record pairs considerably reduces the efficiency of the RL process [58]. Our objective is to use a record pair filtering mechanism to improve the efficiency of the RL process.

2. **Enhance Linkage Quality**: As emphasised in Section 1.2, the lack of data quality, as commonly encountered in real-world data sets, can be detrimental to linkage quality. The lack of data quality in real-world data sets significantly hinders the performance of unsupervised and semi-supervised RL techniques, since these approaches mostly rely on attribute values (which may be erroneous if data is dirty) to determine the match and non-match status of the compared record pairs. Therefore, our objective is to develop methods of improving unsupervised RL techniques, such that high linkage quality can be achieved even when data quality is lacking.

3. **Develop Techniques to Ensure Data Privacy in Record Linkage**: RL is often performed on data sets that contain sensitive information such as personal data (for example, names, addresses, and medical conditions of individuals). Due to this reason, a research area known as Privacy-Preserving Record Linkage

(PPRL) [40] is dedicated to exploring how RL can be conducted in a privacy-preserving manner. We do not consider PPRL techniques in our work here because they hinder the human interpretability of linked data, for example, with the use of binary encryption for ensuring data privacy [156]. Instead, our objective is to propose human understandable, anonymised data sets by exploring alternative mechanisms of data anonymisation as applicable to RL.

4. **Develop Unsupervised and Semi-supervised Record Linkage Techniques**: More often than not, existing supervised RL approaches are not applicable in the real-world due to a lack of ground-truth data, as we have highlighted in Section 1.2. Our objective is to address this issue by developing unsupervised and semi-supervised methods, where the former requires no ground-truth data, and the latter utilises only limited ground-truth data for training a classifier.

5. **Develop Novel Robust Techniques for Evaluating Linkage Results**: As we elaborated in Section 1.2, precision, recall, and the F-measure are commonly used to assess RL techniques even though these measures were originally developed to evaluate information retrieval and machine learning methods. Furthermore, these measures are only applicable for assessing pairwise linkage results, but not clusters or groups of linked records. Our objective is to explore the issues with using these measures for linkage assessment of groups of records (by retrieving the record pairs within groups), and to develop techniques that are specifically suited for evaluating group RL approaches.

1.4 Contributions

We have herewith summarised the contributions we are making in this thesis to the domain of RL by achieving our research objectives highlighted in Section 1.3.

1. **Robust Unsupervised Classification for Record Linkage Utilising Data Characteristics**: While both supervised and unsupervised classification methods are applied in RL, for the majority of real-world applications the pragmatic approach is to use unsupervised techniques due to the difficulty of obtaining ground-truth data for supervised classification. This is due to the large sizes of the data sets to be linked making it impossible for domain experts to manually examine an adequate number of record pairs to determine whether they are true matches or not, to conduct training in supervised RL. Therefore, as presented in Chapters 4 and 5, we have developed novel unsupervised non rule-based classification approaches for RL which do not require ground-truth data for training. Furthermore, these techniques aim to improve the linkage quality by incorporating data characteristics such as temporal and spatial information, and corresponding probability distributions in a population.

2. **Graph Filtering for Record Linkage**: As we described in Section 1.2, the RL process can be inefficient even after blocking and indexing techniques are applied to reduce the number of record pair comparisons, where compared record

pairs are often represented as graphs in RL applications [35]. To the best of our knowledge, no work has been conducted so far to determine how the sizes of such graphs can be reduced by selectively discarding the many non-matching record pairs such that the efficiency of the subsequent classification phase is improved. We introduce a novel filtering technique which utilises patterns along data dimensions (such as temporal or spatial) combined with active learning [169] in Chapter 6, to effectively filter out record pairs that are likely non-matches.

3. **Active Learning Based Record Linkage With Filtering**: As we discussed in Section 1.2, the lack of availability of ground-truth data prohibits the use of supervised approaches in real-world RL applications. Active learning is an emerging area of research, which explores how supervised methods can be applied when only limited ground-truth data is available. In the RL context, however, it is important to develop an active learning strategy that can deal with high class imbalance and the lack of data quality that is often encountered in RL tasks. In Chapter 7 we develop a novel active learning based RL technique which mitigates the class imbalance issue and produces high quality linkage results, while conducting RL in an efficient manner by iteratively filtering out high confidence matches and non-matches.

4. **A Novel Evaluation Method for Record Linkage**: As we discussed in Section 1.2, the evaluation methods currently applied for assessing RL techniques were originally developed for information retrieval or classification methods, and are not always suitable for determining the quality of linkage results. In this work, we show how these evaluation methods can produce ambiguous results when assessing group RL techniques, and introduce a novel evaluation technique as we discuss in Chapter 8.

5. **An Anonymisation Method for Graph Data**: We introduce a novel anonymisation method for graph data which is specifically useful for anonymising the graph representations of data sets to be integrated using RL techniques. While there are several existing anonymisation techniques for graph data [51, 173, 188], they are not applicable in the RL context because they cannot retain the graph similarity structure as required for executing linkage algorithms, and/or the anonymisation renders the graph non-human interpretable. Our method mitigates both these issues, and is therefore ideal for anonymising graph representations of sensitive data. Furthermore, our anonymisation method, which we present in Chapter 9, allows research work conducted in the area of RL to be made accessible to a larger audience without compromising data privacy.

6. **Experimental Evaluation of the Proposed Methods**: We conduct a comprehensive experimental evaluation of our proposed techniques to assess their quality and scalability, and compare these techniques with existing state-of-the-art solutions for RL. We present the results of this empirical evaluation in Chapter 10.

1.5 Research Limitations

In this section, we describe the scope of our research by describing which areas in RL we aim to address.

1. **Population Data**: The novel methods introduced in our research are applicable to population data which is also known as personal data (data about people). We do not develop linkage methods considering other types of data such as for consumer product data. RL is commonly applied on personal data [98, 114, 115], and therefore it is important to give priority to the characteristics of such data sets (such as data quality and privacy issues) when developing new linkage models. Furthermore, we consider data sets with different data dimensions such as time and space. These are, for example, data sets that contain dates of birth (temporal aspect), whereas data sets with addresses or other locations have a spatial (geographic) dimension. The presence of such characteristics is important, since we utilise those in the methods we introduce in our thesis. However, the active learning based RL strategy we present in Chapter 7, the evaluation method we describe in Chapter 8, and the graph data anonymisation technique we propose in Chapter 9, are all applicable to non-personal data as well, as they are independent of the type of data being linked.

2. **Privacy-Preserving Record Linkage**: Our study does not consider Privacy-Preserving Record Linkage (PPRL) [40]. Even though we propose a graph anonymisation technique in Chapter 9 to help researchers publish linked sensitive data sets to a wider audience, we do not propose PPRL techniques where the data is completely obscured from the individuals who conduct a linkage task. In PPRL, data is generally encrypted into binary or other forms, whereas the anonymised data resulting from our approach are interpretable to humans.

3. **Parallel and Distributed Computing**: We do not focus on parallelisation of our algorithms to improve scalability, since this is another research space [39, 99]. Rather, we improve the efficiency of our algorithms at the conceptual and implementation levels by identifying likely true matches and true non-matches early on in the linkage process and removing them from further processing.

4. **Ground-truth Data**: Our quality evaluation is based on the assumption that the ground-truth data sets accumulated by domain experts reflect the true state of the world. That is, we assume that records linked by a domain expert represent the correct real-world entities, and that records not linked by a domain expert represent different real-world entities. The reason for this assumption is due to the possibility for domain experts to make some errors in the manual linkage process. However, since the ground-truth provided by domain experts is the closest we have to the gold standard, this is what we use to assess linkage quality.

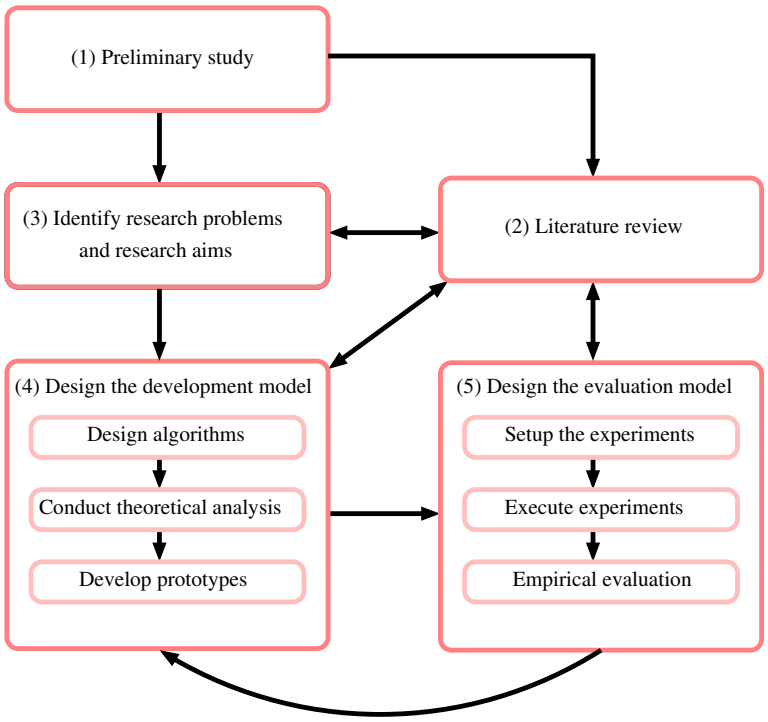

Figure 1.1: The research methodology followed in this thesis.

5. **Single-Source Record Linkage**: In almost all the RL methods we have proposed in our thesis, we have only conducted single-source RL (de-duplication of records within a single data set), except in Chapter 7 where we conduct linkage across two or more data sets as well. However, the methods we have proposed are applicable for multi-source RL as well, given constraints as suitable to the context are applied (such as one-to-one constraints where one record from a data set can only be linked to one record from the other data set).

1.6 Research Methodology

The methodology we follow to achieve our research objectives is illustrated in Figure 1.1, as we further discuss in this section.

1. **Conducting the preliminary study**: Initially we conducted a preliminary study to obtain the background knowledge related to RL and regarding general machine learning and data mining techniques (such as classification), clustering approaches, evaluation methods, blocking methods, and so on. Obtaining a general understanding of related techniques helped us determine limitations in existing RL methods and identify the research problems.

2. **Conducting the literature review**: We studied the existing literature on RL to obtain knowledge about the state-of-the-art RL techniques and their limitations.

This was an ongoing study throughout our research, which helped refine our research problems, the designing of prototypes, and their evaluation.

3. **Define the problem statement and research aims**: Subsequent to identifying the limitations of existing RL techniques, we formulated our research problems and aims so as to address these limitations.

4. **Development model**: In the development model we designed algorithms to address our research aims, and assessed them conceptually with regard to quality and scalability. We then implemented prototypes of these proposed algorithms using the Python programming language[1] due to its simplicity, readability and the availability of many libraries for statistical, data mining, and machine learning tasks.

5. **Evaluation model**: In the evaluation model, we first setup the experiments by deciding on suitable baselines, parameter settings, and data sets. The experiments were then executed to compare our own RL techniques with suitable baseline methods. Subsequently, the obtained results were evaluated for quality and scalability using appropriate evaluation measures [35, 49, 85] to ascertain whether our solutions mitigate the problems associated with baseline approaches.

1.7 Thesis Outline

While we provided an overview of Record Linkage (RL) and highlighted our research contributions in this introduction chapter, the rest of our thesis is structured as follows. In the next chapter, we provide background information about the RL process and the data sets we have used for empirical evaluation. In Chapter 3 we then conduct a literature review of existing RL techniques. In Chapter 4, we introduce three unsupervised classification approaches for record linkage which use data characteristics such as time and space related information to enhance linkage quality. We attempt to further improve the effectiveness of these algorithms by incorporating population distribution patterns as we discuss in Chapter 5. We introduce a novel filtering approach for RL in Chapter 6, where we elaborate on the efficiency improvement gained when applying this approach. Subsequently, in Chapter 7, we present an active learning based RL with filtering technique that employs a new active learning strategy to effectively and efficiently link records. In Chapter 8, we show how existing evaluation methods for group RL can produce ambiguous linkage quality results, and introduce a novel evaluation approach. In Chapter 9 we present our proposed anonymisation method for sensitive graph data, which is helpful in making RL related research conducted on sensitive data sets available to a wider audience. Next, in Chapter 10 we conduct a comprehensive empirical evaluation of our proposed methods using a new data set. Finally, in Chapter 11 we conclude our thesis

[1] https://www.python.org/

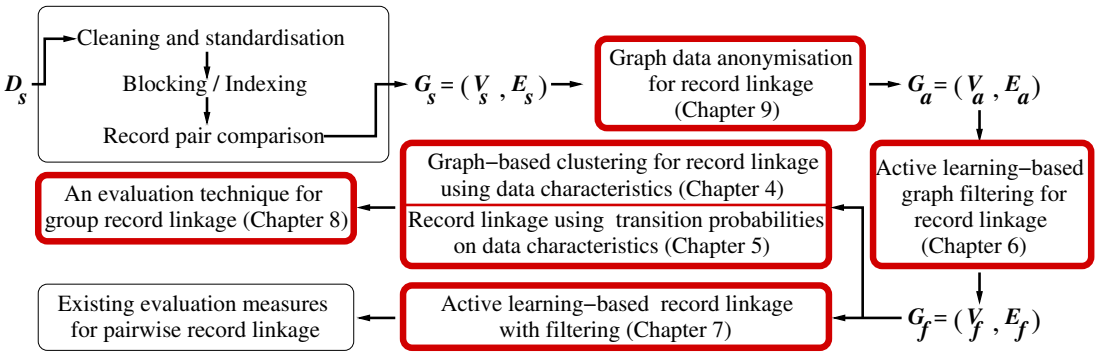

Figure 1.2: Flow diagram indicating how the techniques we propose in this thesis (highlighted in red) can be integrated to conduct end-to-end record linkage.

by highlighting our contributions, how they have addressed the research questions, and what future work could be explored in this area.

As a guideline for the reader, in Figure 1.2 we have highlighted how the chapters where we discuss our contributions (Chapters 4 to 9) connect together. We discuss the cohesiveness of our work as shown in this diagram in detail in Chapter 10.

Chapter 2

Background

In this chapter, we provide background information which is required to understand the area of Record Linkage (RL), and the following chapters in this thesis. In Section 2.1 we provide a concise description of the history of RL, followed by a description of the use of RL for linking complex population data in Section 2.2. In Section 2.3 we elaborate on the primary steps involved in the process of RL. We discuss the evaluation measures which are commonly used to assess the quality and scalability of RL techniques in Section 2.4. Subsequently, in Sections 2.5 and 2.6, we describe the data sets and the corresponding pairwise similarity graphs on which the experimental evaluations throughout this thesis will be based. Finally, in Section 2.7, we conclude this chapter with a summarisation of its content.

2.1 A Brief History of Record Linkage

Statisticians and health researchers have shown interest in the RL domain even before the invention of modern computers [35]. The term *Record Linkage* was first introduced by Dunn in 1946 [59] where he interpreted RL as the process of compiling a book of life for individual persons. These books would begin with the birth record of a person and end with their death record, whereas in between one would encounter records of marriage, censuses, health, and so on. If such a book existed for each individual in a population, it would allow much valuable information to be obtained regarding several generations and facilitate many different social and health studies.

In the 1950s and early 1960s, Howard Newcombe et al. [129, 130] proposed a method to automate the process of RL using computers. The concept of *probabilistic record linkage*, which was initially proposed by Newcombe, was formally established by the two statisticians Ivan Fellegi and Alan Sunter in 1969 [64]. To this date, the probabilistic RL approach and its extensions [177] are the core of many RL systems.

Over the past few decades, interest and research related to RL has increased rapidly, due to technology, social networks, and enhanced database storage allowing the generation and storage of a massive amount of data [82]. In order to link / de-duplicate databases containing millions of records, it is important to develop advanced RL techniques, by employing methods such as sophisticated machine learning (ML), clustering, and graph-based techniques, as we describe in Chapter 3.

2.2 Linkage of Complex Population Data

Many RL tasks correspond to the linkage of population data, which contain information about individuals, such as their names, addresses, ages, and so on. Population data is often complex due to the variations encountered in attribute values such as names (which may have spelling variations), skewed frequency distributions of attribute values such as certain cities appearing more frequently than others in a data set, and their large sizes [35]. Furthermore, most population data sets contain errors introduced during data entry and transcription, thus rendering the linkage of population data sets a difficult task [35].

A noteworthy area of RL is the application of linkage techniques on population data from historical censuses and vital records (birth, death, and marriage certificates), to construct longitudinal data sets about a population. Such linked historical populations are extensively used for studies in the social sciences and genealogy [3, 7, 152]. The problem of historical population linkage has been studied in the past four decades by researchers working in different domains [179].

In 1980, the US Census Bureau conducted a linkage on census data from several US states, using a linkage system developed based on the seminal work of Fellegi and Sunter [64]. Following the same principles, in 1996 Dillon [52] investigated an approach to link census records from the US and Canada to generate a longitudinal database to examine changes in household structures. More recent work related to historical population linkage was done by Antonie et al. [4] who proposed a novel RL technique to automatically assign unique household identifiers to records in the 1891 Canadian census. Fu et al. [70] introduced a novel historical household linkage method which utilises the structural relationships (such as relationships and generation difference) between household members in the linkage process. A temporal approach to linking census data was proposed by Christen et al. [43] which considers the relationships between household members, and the possibility for certain attribute values (such as names and addresses) to change over time.

The Integrated Public Use Microdata Series[1] (IPUMS) is a large project initiated by the Minnesota Population Centre (MPC) which aims to curate and ultimately link large demographic data collections [152]. The Longitudinal, Intergenerational Family Electronic Micro-Database Project[2] (LIFE-M) is another example of transforming historical records into a multi-generational longitudinal database [7], where US vital records are combined with census information.

The Digitising Scotland project[3] 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. Such a linked database will allow researchers in various domains to conduct studies that are currently impossible to do.

[1] https://www.ipums.org/
[2] https://life-m.org/
[3] https://digitisingscotland.ac.uk/

Another area where population RL is extensively applied is the health sector. An invaluable application of RL in recent times was for the containment of the COVID-19 pandemic, which was very effectively used by the Taiwanese health care system to successfully combat COVID-19 [112]. Nguyen et al. [132] discusses about a recent application of RL on patient data for public health surveillance, whereas a renowned, practical example of patient data linkage was conducted in Western Australia which considerably helped to improve health policies in the state [101]. Recent research conducted by Hamm et al. [81] further emphasises the importance of population RL for multi-generational heath research such that early diagnosis and intervention of hereditary illnesses are made possible.

Ensuring national security, and fighting crime prevention in the modern world also relies on sophisticated population RL techniques [35]. Terrorists and criminals often take extreme caution to avoid leaving traces of their activities such as online transactions. Due to this reason, they often impersonate innocent individuals, or use forged information whenever they interact with the rest of the world. Therefore, in order to fight crime and terrorism, population data sets can be linked to identify outliers [113], which could represent deliberately altered identities or potential identity thefts.

Bibliographic data sets such as the ACM Digital Library[4], IEEE Xplore[5], DBLP[6], and Google Scholar[7] contain information about researchers such as their names, publications, and affiliations. Therefore, bibliographic data can be considered as population data, on which RL techniques are extensively applied to support academic related decision making and analysis, for example determining the impact of researchers, and allocate funds to research projects [35].

As highlighted in Section 1.5, the scope of this thesis is to propose RL techniques for integrating population data. We present the population data sets we use for conducting the experimental evaluation of our proposed methods in Section 2.5.

2.3 The Main Steps of the Record Linkage Process

The RL process comprises primarily of five steps, and an optional clerical review step as shown in Fig. 2.1. This section is dedicated to describing these steps of RL [35].

2.3.1 Data Pre-processing

Data pre-processing for RL can include data cleaning and standardisation of the data to be linked, which is crucial for successful RL [35]. While the importance of data pre-processing is indisputable, one should also take care not to introduce new errors to the data while conducting pre-processing [73]. The primary steps involved in data pre-processing are:

[4]https://dl.acm.org/
[5]https://ieeexplore.ieee.org/
[6]https://dblp.org/
[7]https://scholar.google.com/

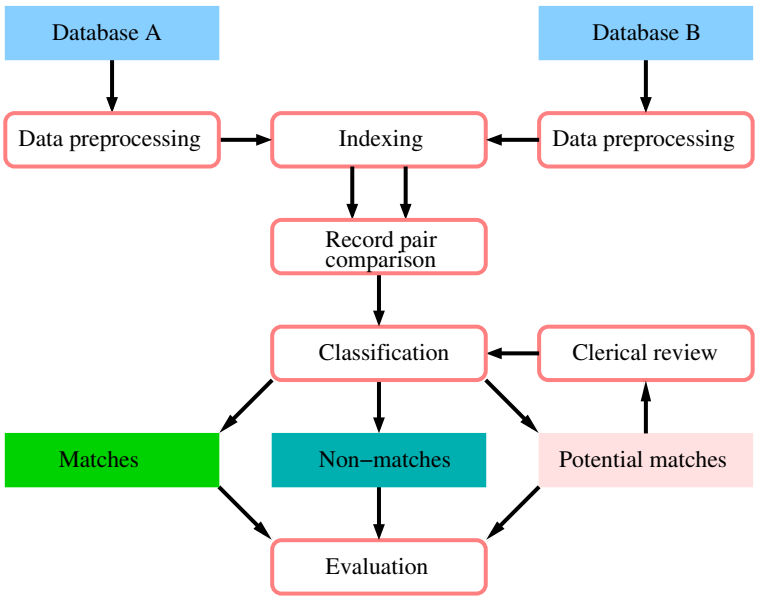

Figure 2.1: The primary steps involved in the process of record linkage (reproduced from [35]).

- Standardising the structure and content of data when conducting RL across different databases. Generally, different organisations have different database structures, even if the data describe the same objects. Therefore, converting all records into the same structure is an important step in data pre-processing. Schema matching, the task of identifying which attributes across databases contain the same type of information, is part of this standardisation step [13, 38]. It may even be necessary to segment certain attributes for convenience of record comparison [93]. For example, the address attribute might be represented as street number, street name, street type, suburb, postcode, and state.

- Removing unwanted characters and words. Stop words [121] and characters such as commas, periods, and hashes can be removed from the data set since they generally are not useful for RL techniques, and can even degrade the linkage performance [35].

- Expand abbreviations, correct misspellings and/or apply phonetic encoding. For example, in the Isle of Skye data set which we discuss in Section 2.5.1, occupation values are often abbreviated (e.g. 'domestic servant' is recorded as 'ds'). Therefore, expanding abbreviations, correcting misspellings, and applying phonetic encoding [35] is very important to improve the quality and consistency of attribute values of records to be linked.

- Verify the correctness of attribute values. This pre-processing step can be applied if valid attribute values are known. For example, a phone number can be verified if its expected number of digits and possible prefixes are known. An

address can be verified if an external database containing all valid addresses of a given region or country is available [35].

2.3.2 Indexing/Blocking

Comparison of each pair of records in a large database (or across two databases) is computationally expensive. The computational effort grows quadratically as the number of records in the databases increases. The techniques used to reduce the number of pairwise comparisons are commonly referred to as indexing [35], whereas the traditional indexing approach is widely known as blocking [12]. The aim of indexing or blocking in RL is to eliminate as many record pair comparisons as possible that correspond to true non-matches [35]. In order to achieve this task, records are assigned into one or several blocks, such that records which are likely to refer to the same entity (correspond to true matches) are assigned to the same block, and records that are likely to correspond to different entities are assigned to different blocks [35]. A block is generally represented by a 'blocking key', which is the common value obtained for all records contained in that block by applying a certain blocking function.

A simple example for a blocking key would be using the prefix (such as the first three letters) of the first or last name attribute in a certain data set. All records that share the same prefix of the name attribute would belong to a single block, and pairwise similarities would be calculated within each block. Phonetic encoding algorithms, such as Soundex [32], are also used as blocking techniques in some instances, whereas there exist a number of other complex indexing methods such as Locality Sensitive Hashing (LSH), sorted neighbourhood indexing and q-gram based indexing [35, 138]. We now briefly explain the Soundex and LSH algorithms, since we use them to conduct blocking on the population data sets we present in Section 2.5.

- **Soundex**: In the Soundex approach, which was developed by Russel and Odell in 1918 [133], strings are encoded based on how they are pronounced in American English such that strings with similar pronunciation are converted into the same Soundex code. This method is particularly helpful when blocking records using person names, since certain names can be spelt in different ways. For example, applying the Soundex algorithm on the names 'Gail', and 'Gayle' results in the same Soundex code 'g400'.

- **Locality Sensitive Hashing (LSH)**: LSH is a technique that maximises the likelihood of hashing similar items into the same 'bucket', and therefore it can be used as a blocking approach in RL [107]. We employ a specific implementation of LSH known as min-hashing based LSH [107]. The similarity of a min-hash signature pair is an approximation to the Jaccard similarity of two values [107]. The min-hash signature generated for each record is split into 'bands' using a user defined number of min-hash bands and a band size. Record pairs are assigned to the same block for comparison if they have the same value in at least some of these 'bands'.

There is, however, a trade-off in applying blocking techniques since the efficiency improvement often comes at the cost of losing some true matching record pairs [184]. A good blocking approach would discard as many non-matching record pairs as possible, while retaining almost all true matches. Scalability measures can be used to assess blocking techniques in this regard, as we further discuss in Section 2.4.2.

2.3.3 Record Pair Comparison and Pairwise Similarity Graph Generation

Subsequent to indexing records, the records which were indexed into the same block are compared in more detail in order to determine whether they refer to the same entity or not. Generally, several attribute values of a record pair are compared to calculate their similarity. Since a true matching record pair would not necessarily have exact matches across all compared attributes [35], it is important to reflect the extent to which records are similar. Therefore, appropriate approximate similarity functions, such as Jaro-Winkler [95] and date similarity functions, are used to compare attribute values. These values are then normalised to reflect a similarity between 1.0 and 0.0, where 1.0 means an exact match and 0.0 means total dissimilarity [34]. Next, the normalised attribute value similarities are combined to reflect the overall similarity of the record pair, with or without using weights for each attribute (which reflects the importance of an attribute in identifying matching records). The similarity functions we used to compare records in the data sets used for experimental purposes in this thesis (as described in Section 2.5) are as follows [35].

- **Exact**: The simple exact comparison function returns a similarity value of 1.0 for exactly matching attribute value pairs, and 0.0 otherwise.

- **Edit distance based string similarity (Edit)**: In this method, the string similarity calculation is based on the minimum number of edit operations (character insertions, deletions, and substitutions) which are required to convert one string into another [35]. The similarity is 1.0 if no edits are required, whereas the similarity is 0.0 if all characters need to be converted.

- **Jaccard coefficient based similarity (Jacc)**: In the Jaccard comparison method, strings are initially split into q-grams or tokens (we have used bi-grams in our experiments where for example the string 'pete' has the bi-grams 'pe', 'et', and 'te'), and the similarity is calculated as the proportion of common unique q-grams, compared to the number of all unique q-grams in the two strings being compared. For a pair of identical strings, a Jaccard similarity of 1.0 is obtained, whereas the similarity is 0.0 if the string pair has no common q-grams.

- **Dice coefficient based similarity (Dice)**: The dice coefficient based comparison function is similar to the Jaccard method. However, rather than comparing the number of common unique q-grams against the number of all unique q-grams, in the Dice method the number of common unique q-grams is compared against the average number of q-grams across the compared pair of strings.

- **Jaro-Winkler string comparison (JW)**: This is an approximate string comparison function which was specifically developed for name comparison by Matthew Jaro and William Winkler from the US Census Bureau [95, 177], by taking numerous name related heuristics into account. For example, the US Census Bureau [182] has identified that people tend to make fewer errors at the beginning of names when they write or type them compared to the remainder of name strings, and therefore higher weight (or more importance) is given to the first few characters in name comparison with the Jaro-Winkler method.

- **Maximum absolute difference (MAD)**: This is a method used for approximate comparison of numerical attribute value pairs [35]. If, for two numerical values n_1 and n_2, the absolute difference is less than a user defined threshold δ_d, then the pairwise similarity is calculated as a linear extrapolation between 1.0 and 0.0 as $1.0 - (|n_1 - n_2|/\delta_d)$. Otherwise, if $|n_1 - n_2| \geq \delta_d$, the similarity for the numerical value pair is 0.0.

While there are advanced methods of conducting record pair comparison in the literature [97], we adopt the widely applied weighted attribute value similarity technique, which can be formally defined as follows. Note that even though we mostly consider single-source RL in this thesis, in the following definitions we have considered the multi-source (two-source) situations as well for these definitions to be generally applicable.

Definition 1 (Record Pair Comparison) *Let $\mathbf{D}_A$ and $\mathbf{D}_B$ be two data sets to be linked. A blocking method as discussed in Section 2.3.2 has generated a set of candidate record pairs (r_i, r_j), with $r_i \in \mathbf{D}_A$ and $r_j \in \mathbf{D}_B$. For a de-duplication problem on a single data set $\mathbf{D}$, we can assume $r_i \in \mathbf{D}, r_j \in \mathbf{D}, i \neq j$. We denote the list of attributes such as names and addresses corresponding to records in a data set by $\mathbf{A}$, where a single attribute is denoted with $a \in \mathbf{A}$. We consider a list of weights $\mathbf{w}$ ($|\mathbf{w}| = |\mathbf{A}|$) where each weight corresponding to an attribute $a \in \mathbf{A}$ is represented as $w_a \in \mathbf{w}$ ($0.0 \leq w_a \leq 1.0$). A higher weight w_a reflects higher importance of attribute $a \in \mathbf{A}$ in the linkage process.*

Let $\mathbf{S}$ be a list of similarity functions, and $S_a \in \mathbf{S}$ be the similarity function used to compare a record pair on attribute $a \in \mathbf{A}$. The pairwise similarity of a record pair (r_i, r_j) on a single attribute $a \in \mathbf{A}$ can be denoted as $S_a(r_i, r_j)$. We can now represent each compared record pair by a similarity vector, $\mathbf{s}_{i,j}$, where $|\mathbf{s}_{i,j}| = |\mathbf{A}|$. Furthermore, assuming all similarities are in $[0, 1]$ (with a similarity of 0 for totally different values and 1 for an exact match), an overall normalised similarity for the record pair (r_i, r_j) can be calculated as:

$$s_{i,j} = \frac{\sum_{a \in \mathbf{A}} S_a(r_i, r_j) \cdot w_a}{\sum_{a \in \mathbf{A}} w_a}. \tag{2.1}$$

A graph representation of compared records makes it possible to apply graph-based classification and clustering techniques to identify the final linked records [148, 154]. Therefore, a pairwise similarity graph is often generated in the comparison

phase, where nodes in the graph represent records, and an edge between two nodes represents a record pair and the corresponding pairwise similarity [148]. Other metadata corresponding to a record pair, such as relationships among individuals and distances between records in relation to time and space, can also be associated with edges in a graph, as we discuss in later chapters. In Section 2.6 we describe the pairwise similarity graphs generated for the data sets we used in our experiments. A pairwise similarity graph can be formally defined as follows.

Definition 2 (Pairwise Similarity Graph) *The set of compared records can be represented as an undirected similarity graph,* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$*, where each node (vertex) in* $\mathbf{V}$ *represents a record,* r_i *or* r_j*, and an edge* $(r_i, r_j) \in \mathbf{E}$ *connects two records* r_i *and* r_j *if their overall similarity* $s_{i,j}$ *(as provided in Equation 2.1) is at least a certain similarity threshold* δ_s *(*$s_{i,j} \geq \delta_s$*), with* $0 \leq \delta_s \leq 1$*. An edge may be represented by the overall similarity* $s_{i,j}$ *or a similarity vector* $\mathbf{s}_{i,j}$ *as we described in Definition 1.*

2.3.4 Match and Non-match Classification

After obtaining the similarity of compared record pairs, these record pairs are classified as matches, non-matches, and sometimes into a third class of potential matches. The potential matches class consists of the ambiguous links to be reviewed by a domain expert [93]. The classification could either be based on threshold similarities which classify record pairs (r_i, r_j) based on their overall similarities $s_{i,j}$, or use supervised, unsupervised, or semi-supervised machine learning (ML) techniques which are generally based on the attribute level similarity vector, $\mathbf{s}_{i,j}$, for classification [35].

Supervised ML techniques are extensively applied for addressing many data related problems in the modern world, including the linkage of records [69, 103, 120]. Even though the application of supervised ML techniques for RL has been shown to be highly effective [55], a major limitation of this approach is the requirement of training (ground-truth) data which is unavailable for most RL projects [35]. Therefore, unsupervised methods such as clustering, and semi-supervised methods such as active learning, need to be employed for real-world RL applications [33, 142, 145, 154].

In recent years, collective classification techniques such as graph-based clustering [86, 148, 153] have been developed, since the traditional classification methods suffer from the inability to consider the potential dependencies among record pairs (such as the inability to consider the transitive closure [35]). Clustering is the process of grouping similar records, where each group is assumed to represent a single entity. Obtaining groups with high intra-cluster similarity (objects within a group are highly similar with one another) and low inter-cluster similarity (objects in different groups are highly dissimilar with one another) is the aim of clustering approaches [82]. A pairwise similarity graph needs to be generated at the RL comparison phase for clustering to be conducted, as we described in Section 2.3.3. The clusters resulting from the application of graph clustering techniques can then be analysed to identify whether the records within a cluster represent the same entity. Star clustering, center clustering, and merge clustering are examples of graph clustering techniques [58, 86, 153].

Recently, in contrast to traditional pairwise record linkage, *group linkage* [134] has received significant attention because of its applicability in linking groups of individuals, such as individuals in families or households [43, 71]. The aim of group linkage is to identify groups of records that match across data sets. For example, a family or household would consist of records of several people, where each record represents a single entity. Group linkage can be applied when we want to identify whether the same family or household appears in two data sets [43].

Given that the majority of our contributions as listed in Section 1.4 aim to improve the classification step in the RL process, we extensively discuss and provide information regarding existing classification approaches in Chapter 3.

2.3.5 Evaluation

Subsequent to classifying record pairs or clusters into matches and non-matches, the quality and completeness of a RL method needs to be assessed. Ideally, quality can be assessed against a ground-truth data set, where domain experts have manually classified record pairs as matches or non-matches. However, where ground-truth data is not available, quality evaluation is based on measures which utilise certain characteristics of the linkage results itself (such as the similarity of matches and the disparity of non-matches) [53].

Apart from the quality of RL techniques, the scalability of a linkage algorithm needs to be assessed as well. Since RL generally concerns large data sets and any RL algorithm needs to scale up for it to be of use in real-world applications, evaluating scalability is of considerable significance in RL. We next describe the measures used for RL evaluation in more detail.

2.4 Evaluation Measures

In this section, we describe the evaluation measures commonly used to assess linkage quality and scalability of RL techniques [35].

2.4.1 Linkage Quality Measures

When ground-truth data is available for a linked data set, it is possible to evaluate the quality of RL techniques using several measures. All the compared record pairs can be categorised into four classes as True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). The TP consist of record pairs which were classified as matches and are also matches in the ground-truth. The FP are the record pairs which were classified as matches but are non-matches in the ground-truth. The TN consist of record pairs which were classified as non-matches and are also non-matches in the ground-truth. The FN are the record pairs which were classified as non-matches but are matches in the ground-truth. The linkage quality measures can be defined as follows, using these four categories.

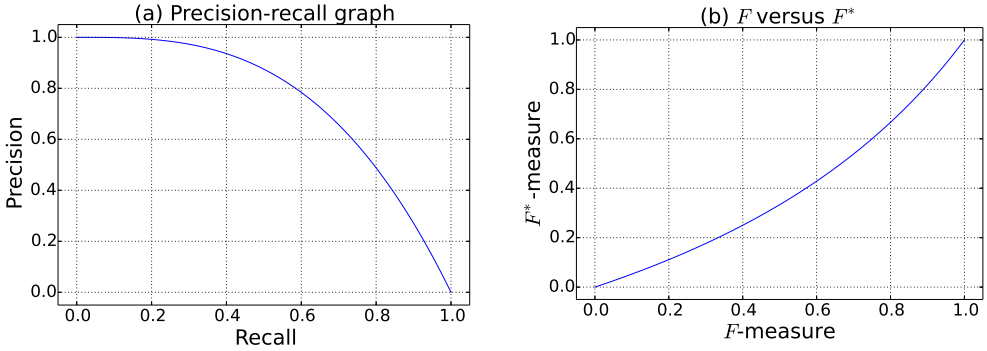

Figure 2.2: Plots showing (a) a precision-recall graph, and (b) the relationship between F and F^* measures.

- **Precision** [178] calculates the proportion of how many of the classified matches ($TP + FP$) are true matches (TP). It can be calculated as:

$$Precision\,(P) = \frac{TP}{TP + FP} \tag{2.2}$$

- **Recall** [178] calculates the proportion of true matches ($TP + FN$) that were classified correctly (TP). It can be calculated as:

$$Recall\,(R) = \frac{TP}{TP + FN} \tag{2.3}$$

- **F-measure** [127] calculates the harmonic mean between precision and recall. It can also be represented as a weighted mean of P and R using a weight of $\theta = (FN + TP)/(FN + FP + 2TP)$ [85]. The F-measure can be calculated as:

$$F\text{-measure}\,(F) = \frac{2 \cdot P \cdot R}{P + R} = \theta R + (1 - \theta)P \tag{2.4}$$

Apart from the above measures, it is also possible to use visualisations to assess the quality of a linkage technique. The *Precision-recall (PR) graph* [35] as shown in Figure 2.2 (a) is one such visualisation, where precision values are plotted against recall values for different classification parameter settings, such as different similarity thresholds used in pairwise classification. The *Area under the precision-recall curve* (AUPRC) corresponding to the PR graph is also used as a RL evaluation measure [23].

Even though the F-measure and the *F-measure graph* (which is used for comparing F-measure results obtained with different similarity thresholds) are commonly used to evaluate RL techniques, recent research has found that this measure is not suitable to comparatively assess RL approaches due to the relative importance (reflected by the weight θ as shown in Equation 2.4) assigned to precision and recall in the F-measure depending on the number of predicted matches [85]. As an alternative, we use the recently introduced F^*-measure, which mitigates some of the issues with

the F-measure [83], where F^* is the number of true matches identified against the number of matches which are either misclassified or are correctly classified matches. F^* is calculated as:

$$F^* = \frac{P \cdot R}{(P + R - P \cdot R)} = \frac{F}{(2 - F)} = \frac{TP}{(FN + FP + TP)} \tag{2.5}$$

Figure 2.2 (b) shows the monotonic relationship between the F-measure and the F^*-measure. This indicates that conclusions made based on F^* (such as the choice of classification methods and parameters) are identical to the conclusions made based on F-measure results.

All the linkage quality evaluation measures which we have discussed in this section assess the quality of RL techniques considering record pair (pairwise) classification as matches and non-matches. When group RL approaches are used, record pairs within groups (clusters) are considered in the quality evaluation. Even though these evaluation measures are adequate for assessing pairwise RL techniques, the suitability of these measures for evaluating group RL methods (as we discussed in Section 2.3.4) lacks investigation in the literature, as we discuss in Chapter 3. Therefore, in Chapter 8 we propose a novel evaluation measure for group RL.

2.4.2 Linkage Complexity and Scalability Measures

Scalability measures assess how scalable RL techniques are. Note that as described in Section 1.5 we do not consider parallel scalability in this section. we Run-time is one popular scalability measure which conveys the time taken by a RL method to execute all the steps from pre-processing until the generation of a linked data set. However, run-times are highly reliant on the specifications of the machine used for execution. Therefore, it is desirable to have measures which produce results that are independent of external factors. *Reduction ratio, pairs completeness* and *pairs quality* [35] are such scalability measures used in the domain of RL which utilise the number of candidate record pairs generated by an indexing or blocking method to assess scalability. The following notation is used to present these measures [35].

The total number of matched and non-matched record pairs are denoted with o_m and o_n respectively. These numbers refer to the total numbers of possible comparisons and not the comparisons after indexing. Therefore, when RL between two databases $\mathbf{D}_A$ and $\mathbf{D}_B$, containing $|\mathbf{D}_A|$ and $|\mathbf{D}_B|$ records respectively, is concerned, $o_m + o_n = |\mathbf{D}_A| \times |\mathbf{D}_B|$. If RL is conducted to de-duplicate a single database with $|\mathbf{D}|$ records, then we have $o_m + o_n = |\mathbf{D}|(|\mathbf{D}| - 1)/2$. Let us denote the number of true match and true non-match candidate record pairs generated by an indexing technique as i_m and i_n, respectively. Note that generally $i_m + i_n \ll o_m + o_n$.

- **Reduction ratio** calculates the relative reduction in the comparison space due to indexing, with respect to the total number of possible comparisons. It can be calculated as:

$$RR = 1 - \left(\frac{i_m + i_n}{o_m + o_n}\right) \tag{2.6}$$

- **Pairs completeness** calculates the proportion of how many of the true matching record pairs in the total comparison space are captured as true matches by an indexing method (corresponding to recall). It can be calculated as:

$$PC = \frac{i_m}{o_m} \tag{2.7}$$

- **Pairs quality** is the number of true matches generated by an indexing method, divided by the total number of candidate record pairs that were generated (corresponding to precision). It can be calculated as:

$$PQ = \frac{i_m}{i_m + i_n} \tag{2.8}$$

Note that ground-truth data must be available to calculate pairs completeness and pairs quality, since otherwise the true matching record pairs i_m generated by an indexing technique are not known.

2.5 Population Data Sets

In this section we describe the different data sets we use to experimentally evaluate the RL techniques we propose in this thesis. We use six data sets, among which three are birth data sets, providing information regarding the birth of individuals and their parents. Two are bibliographic data sets which contain details of researchers and their publications, and one is a voter data set containing personal information of voters (such as their names and addresses) from a state in the US.

Prior to discussing the individual data sets we have used in our experiments, we define and distinguish between the terms *certificate* and *record*, which we extensively use throughout this thesis. A *certificate* is the actual document which contains information regarding a single person, or several related individuals [149]. A *record*, however, contains information regarding a single entity, where the entity could be an individual, or a publication. For example, up to three records can be generated from a birth certificate corresponding to the baby, the mother and the father, if their details are provided. Likewise, up to six records can be generated from a marriage certificate (related to the bride, the groom, the bride's parents, and the groom's parents), and up to four records from a death certificate (the deceased, their parents, and possibly their spouse). A collection of records is referred to as a data set.

2.5.1 Birth Data Sets

We use the following three birth data sets for evaluating our proposed methods. Note that these birth data sets contain *birth records* which we have retrieved from the corresponding *birth certificates*. The main entity in a birth record is the baby whereas it includes the baby's parents' information as attributes describing the baby.

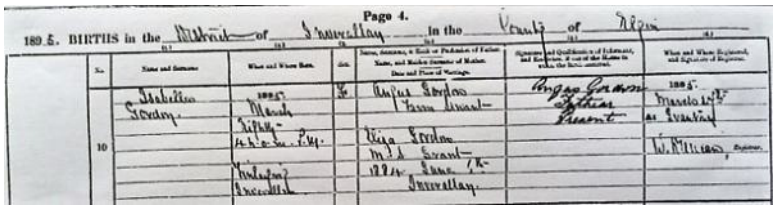

Figure 2.3: A snippet of a Scottish birth certificate from 1895. Image taken from http://www.scotlandsgenealogy.com/blog/scottish-birth-certificate/.

Table 2.1: The number of records in each data set, and the number of ground-truth links o_m corresponding to each data set, as determined by a domain expert.

Data set	Number of records	Number of ground-truth links (o_m)
IoS	17,613	41,247
Kilm	37,121	68,215
UK	14,027	22,019
NCVR	14,619,783	6,978,001
ACM	2,294	2,220 (DBLP-ACM)
DBLP	2,616	5,347 (DBLP-Scholar)
Google Scholar	64,263	

- **Isle of Skye (IoS) data set**: This is a real-world data set [149] covering the population of the Isle of Skye (an island in Scotland) over the period from 1861 to 1901, which we extensively use in our research. This data set consists of 17,613 birth records derived from the original hand-written birth certificates from Scotland, as shown in Table 2.1. Each record contains personal details about a baby and its parents, such as their names, address, parents' marriage date, parents' occupations, and the birth date of the child. A snippet of such an original Scottish birth certificate is shown in Figure 2.3, highlighting the lack of legibility of these original certificates that can potentially result in transcription errors when birth certificates are digitised, transcribed, and converted into birth records.

  As with other historical data [4, 70], this data set has a very small number of unique name values (2,055 first and only 547 last names). As Figure 2.4 (a) shows, the frequency distributions of names and addresses are also very skewed, with few attribute values occurring many times. The five most common first and last name values occur in between 30% and 40% of all records, as Table 2.2 illustrates. Many records have missing addresses or occupations, and for unmarried women the details of a baby's father are missing.

  This data set has been extensively curated and linked semi-manually by demographers who are experts in the domain of linking such historical data [149]. Their approach followed long established rules for family reconstruction [179], leading to a set of linked birth records. We thus have a set of manually generated links of births that allows us to assess the quality of different RL tech-

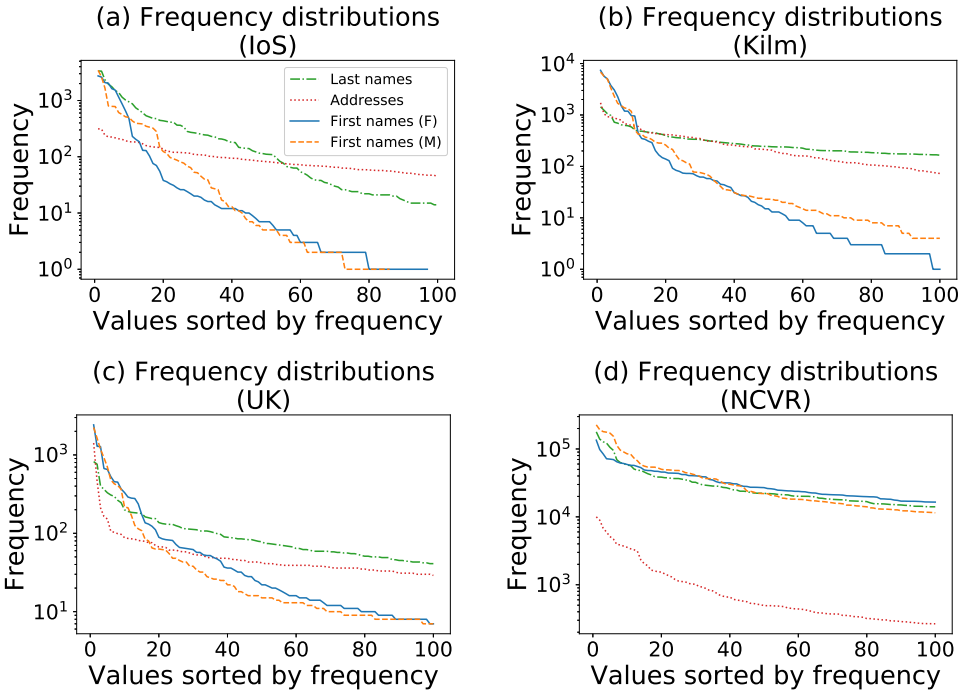

Figure 2.4: Sorted frequency distributions of male and female first names, last names, and addresses in the (a) IoS, (b) Kilm, (a) UK birth data sets, and (d) the NCVR data set. Notice the highly skewed frequency distributions especially for first names in the birth data sets, where a small number of attribute values occur many times.

niques. The number of ground-truth links o_m (pairs of records corresponding to births by the same mother, or siblings) is 41,247 for the IoS data set, as we show in Table 2.1. Since the IoS data set is a real-world data set which is challenging to be linked due to the presence of errors, missing values, and the skewness of its attribute value frequency distributions, we use this data set for evaluating all our proposed methods presented in Chapters 4 to 9.

- **Kilmarnock (Kilm) data set**: This is also a real-world data set covering the population of Kilmarnock, a town in West Scotland, over the period from 1860 to 1901. As shown in Table 2.1, this data set contains 37,121 birth records derived from the original birth certificates, where each record contains personal details about a baby and its parents, similar to the IoS data set [149]. With the manual linkage conducted on this data set by domain experts, 68,271 ground-truth record pairs corresponding to siblings have been identified, as shown in Table 2.1. The name and address attribute value frequency distributions are again very skewed, as illustrated in Figure 2.4 (b) and Table 2.2, where few attribute values occur many times.

  Similar to the IoS data set, linking the Kilmarnock data set is also challenging due to the presence of errors and missing values, and we therefore use

Table 2.2: The five most frequent values and their corresponding frequency counts for male and female first names, last names, and addresses in the birth data sets (IoS, Kilm, and UK) and the NCVR voter data set.

Data set	First name		Last name	Address
	Male	**Female**		
IoS	John (3,444)	Mary (2,740)	Mcdonald (3,349)	Breakish (314)
	Donald (2,628)	Catherine (2,606)	Mcleod (3,332)	Aird (310)
	Alexander (1,665)	Ann (2,084)	Mckinnon (2,332)	Roag (241)
	Malcolm (800)	Margaret (2,031)	Nicolson (1,954)	Edinbain (226)
	Neil (787)	Christina (1,626)	Mclean (1,758)	Bernisdale (224)
Kilm	John (6,809)	Janet (7,387)	Brown (1,445)	Robertson Pl (1,722)
	James (5,700)	Mary (5,675)	Wilson (1,237)	High St (1,058)
	William (4,991)	Margaret (5,036)	Thomson (1,059)	Low Glencairn St (925)
	Robert (3,342)	Elizabeth (3,746)	Smith (980)	Titchfield St (911)
	Thomas (2,324)	Agnes (3,072)	Campbell (723)	Fore St (890)
UK	John (2,239)	Mary (2,419)	Ashworth (822)	Burnley Road (1,416)
	James (1,636)	Elizabeth (1,288)	Taylor (762)	Bacup Road (464)
	William (1,181)	Sarah (1,279)	Lord (401)	Bury Road (211)
	Thomas (921)	Alice (670)	Haworth (364)	Church Street (164)
	George (702)	Margaret (653)	Pickup (329)	Haslingden Road (152)
NCVR	James (226,474)	Mary (134,660)	Smith (178,653)	1801 Fayetteville St (10,004)
	Michael (192,273)	Jennifer (98,201)	Williams (139,660)	9201 Uni City Blvd (8,954)
	William (180,012)	Elizabeth (84,655)	Johnson (127,625)	0 WSSU (6,819)
	John (176,597)	Patricia (71,707)	Jones (121,855)	1 Duke Uni West (5,953)
	Robert (171,170)	Linda (70,855)	Brown (105,539)	0 WFU (5,125)

this data set to conduct the overall comparative experimental evaluation of our developed techniques, which we present in Chapter 10.

- **Birth data set based on the 1901 UK census data set (UK)**: This is a synthetic birth data set which we generated using real census data from the district of Rawtenstall in North-East Lancashire in the United Kingdom (UK) from the year 1901 [43, 67]. This original 1901 UK census data set, which is the latest census data set used by Fu in her thesis [67], contains information about each individual in a household, such as their name, address, occupation, gender, age, and birth place. Furthermore, individuals from a single household are grouped together using a household identifier, whereas the household head and the relationships of the other household members to the head are indicated in this data set. This allows us to generate a synthetic birth data set and the corresponding ground-truth data. This synthetic UK birth data set therefore contains information about a baby and its parents, similar to the IoS and Kilmarnock data sets.

  To make this synthetic data set more challenging, we have corrupted the name and address attribute values in 20% of the families in this census data set. As shown in Table 2.1, this UK birth data set has 14,027 birth records where 22,019 record pairs correspond to pairs of siblings, or ground-truth links as

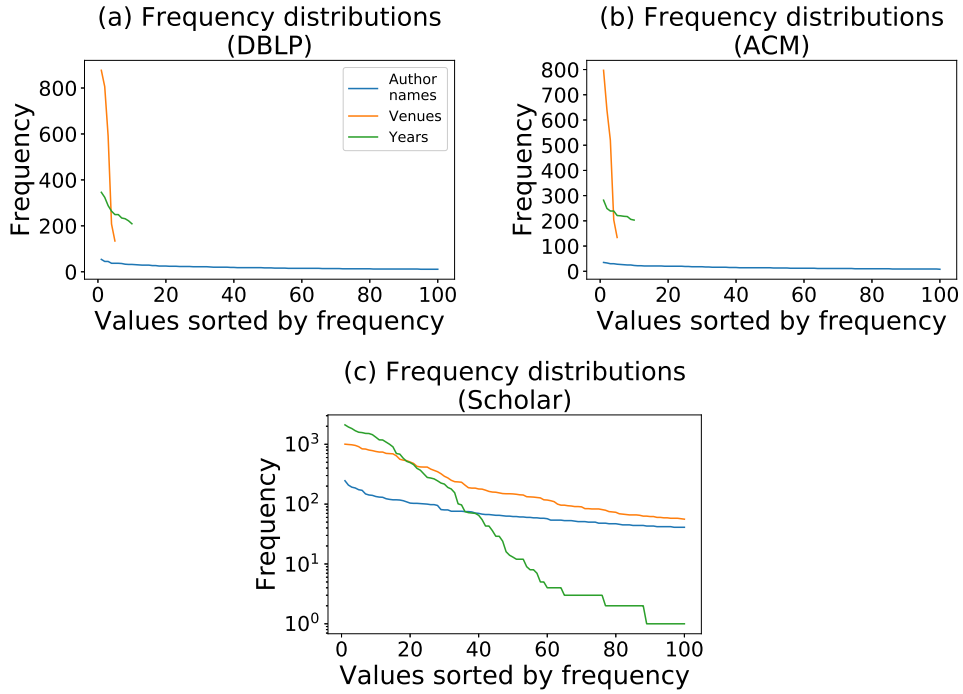

Figure 2.5: Frequency distribution of author names, venues, and years corresponding to publications in the bibliographic data sets (a) DBLP, (b) ACM, and (c) Scholar.

derived from the roles of individuals in each family in the census data set (son or daughter of the household head). Figure 2.4 (c) and Table 2.2 show the skewness of the attribute value frequency distributions in this data set. Similar to the IoS data set, we use this data set for evaluating all our proposed methods presented in Chapters 4 to 9.

2.5.2 The North Carolina Voter Registration Data Set (NCVR)

The NCVR data set contains personal information about US voters from the state of North Carolina[8], such as their names, addresses, gender, birth years, and birth places. The original NCVR data set contains information about voters collected for each election held within a year, for the past 15+ years[9]. We use a subset of the full NCVR data set by combining the voter data available from the years 2011 to 2020, where records corresponding to the same individual across the years can be identified with a unique voter identifier.

When combining these files, we retain the first occurrence of an individual, whereas a record of that person for each subsequent year is retained only if they have at least one attribute value changing (among a selected set of attributes) compared to

[8]https://dl.ncsbe.gov
[9]https://www.ncsbe.gov/results-data/voter-registration-data

their last retained record, in order to make the linkage task more challenging. However, instances where the first and last names together with the gender all change across the years are ignored (since those attribute values play a vital role in identifying an individual), whereas individuals with invalid values for attributes such as the registration year are also excluded. The reason why we follow an individual over time when combining the files is because we aim to link the NCVR snapshots over time in order to identify record pairs that correspond to the same individual.

The NCVR data set thus generated contains 14,619,783 records, where 6,978,001 record pairs correspond to the same individual (or ground-truth links) as shown in Table 2.1. The name and address values of individuals in the NCVR data set are somewhat skewed as shown in Figure 2.4 (d) and Table 2.2, but the skewness is relatively less compared to the skewness of the historical birth data sets. Even though this means that linking the NCVR data set should be less challenging, due to its large size, this data set is useful for assessing the methods which we propose in Chapters 6 and 7 to enhance the efficiency of the RL process.

2.5.3 The Bibliographic Data Sets

We consider three bibliographic data sets, DBLP, ACM, and Google Scholar (which we refer to as Scholar), which were originally provided by Köpcke et al. [104]. The aim of linking these data sets is to identify publications by the same authors across the DBLP and ACM, and the DBLP and Scholar data set pairs, respectively. Rather than using the original versions of these data sets, we use the ones provided by Mudgal et al. [120] who have organised these data sets for conducting supervised classification[10]. The two bibliographic data set pairs we use for experimental evaluation are as follows.

- **DBLP-ACM**: In this data set pair, the aim is to link records corresponding to the same publication across the DBLP and ACM bibliographic data sets. As shown in Table 2.1, the DBLP-ACM data set pair has 2,220 record pairs corresponding to the same publication, which are the ground-truth links.
- **DBLP-Scholar**: In this data set pair, the aim is to link records corresponding to the same publication across the DBLP and Google Scholar bibliographic data sets. As shown in Table 2.1, the DBLP-Scholar data set pair has 5,347 ground-truth links.

The frequency distributions for attribute values author names, venues, and years in these bibliographic data sets are shown in Figure 2.5. The venue and year values span across a small range for the DBLP and ACM data sets, because these contain publications from only five venues spanning across only ten years (from 1994 to 2003). Unlike for the birth data sets, the author names in these bibliographic data sets show a uniform distribution and therefore lack skewness, especially for the DBLP and ACM data sets. For these reasons, linking these bibliographic data sets

[10]See: https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

Table 2.3: Attributes in birth data sets used for calculating pairwise similarities (using similarity functions discussed in Section 2.3.3) to generate the similarity graphs, as discussed in Section 2.6. For the birth data sets we generate three similarity graphs $\mathbf{G}_A$, $\mathbf{G}_{NA}$, and $\mathbf{G}_N$, considering the attribute combinations *All*, *Parent names and addresses*, and *Parent names only*, respectively.

Attribute	**Function**	**IoS**			**Kilm**			**UK**		
		$\mathbf{G}_A$	$\mathbf{G}_{NA}$	$\mathbf{G}_N$	$\mathbf{G}_A$	$\mathbf{G}_{NA}$	$\mathbf{G}_N$	$\mathbf{G}_A$	$\mathbf{G}_{NA}$	$\mathbf{G}_N$
Father first name	JW	✓	✓	✓	✓	✓	✓	✓	✓	✓
Father last name	JW	✓	✓	✓	✓	✓	✓	✓	✓	✓
Mother first name	JW	✓	✓	✓	✓	✓	✓	✓	✓	✓
Mother last name	JW	✓	✓	✓	✓	✓	✓	✓	✓	✓
Parents marriage date	Edit	✓			✓					
Parents marriage place	JW	✓			✓					
Occupation father	Dice	✓			✓			✓		
Occupation mother	Dice	✓			✓			✓		
Address	Dice	✓	✓		✓	✓		✓	✓	
Source parish	JW	✓	✓		✓	✓		✓	✓	✓

Table 2.4: The blocking quality and the number of record pairs (edges) contained in each pairwise similarity graph generated for the birth data sets.

Data set	**Number of record pairs**			**Reduction ratio** (RR)	**Pairs completeness** (PC)	**Pairs quality** (PQ)
	$\mathbf{G}_A$	$\mathbf{G}_{NA}$	$\mathbf{G}_N$			
IoS	5,373,498	16,130,256	21,012,728	0.591	1.000	0.002
Kilm	25,045,978	50,802,438	72,468,163	0.769	0.992	0.001
UK	4,268,343	5,920,045	5,716,423	0.617	0.996	0.002

is relatively easier than linking birth data sets. However, we use these data sets for assessing the active learning based RL strategy we propose in Chapter 7, since most existing supervised and active learning based linkage methods have been assessed using these same data sets.

2.6 Generating Pairwise Similarity Graphs from Data Sets

In this section, we describe how we generate pairwise similarity graphs subsequent to the RL comparison step for the data sets presented in Section 2.5. The formal definitions of record pair comparison and similarity graph generation were provided in Definitions 1 and 2 respectively.

Prior to similarity graph generation, as discussed in Section 2.3.2 we applied LSH and Soundex blocking on the birth data sets and the NCVR data set respectively, to improve the efficiency of the record pair comparison step. Blocking was not applied on the bibliographic data set pairs since most true non-matches in them were already removed [120].

As we described in Section 2.3.3, we then generated a pairwise similarity graph $\mathbf{G}$ for each of the data sets to be linked, by comparing attribute values for pairs of

Table 2.5: Attributes in the NCVR and bibliographic data sets used for calculating pairwise similarities (using similarity functions discussed in Section 2.3.3) to generate the similarity graphs, as discussed in Section 2.6

Attribute	Function	NCVR	DBLP-ACM	DBLP-Scholar
First and Last names	JW	✓	✓	✓
Gender	Exact	✓		
Street and City	Dice	✓		
Zip-code	Jacc	✓		
Title	Jacc		✓	✓
Venue	Jacc		✓	✓
Year	MAD		✓	✓

Table 2.6: The blocking quality and the number of record pairs (edges) contained in each pairwise similarity graph generated for the NCVR and bibliographic data sets.

Data set	Number of record pairs	Reduction ratio (RR)	Pairs completeness (PC)	Pairs quality (PQ)
NCVR	34,589,478	0.999	0.821	0.0002
DBLP-ACM	12,363	-	-	-
DBLP-Scholar	28,707	-	-	-

records. For the birth data sets, we used a similarity threshold $\delta_s = 0.5$ to determine the record pairs to retain, since we identified that the majority of record pairs with an overall similarity below 0.5 ($s_{i,j} < 0.5$) were true non-matches. However, for the NCVR and bibliographic data sets we set $\delta_s = 0.0$ to include all compared record pairs. Furthermore, we set equal weights to all attributes used in the comparison in our experiments ($w_a = 1.0$ for $a \in A$) based on the assumption that all attributes are equally important in distinguishing among matching and non-matching record pairs.

- **Birth data sets**: We generated three types of pairwise similarity graphs for each birth data set by comparing different subsets of attributes, as shown in Table 2.3. The similarity graphs are *All* ($\mathbf{G}_A$), where all attribute values are compared, *Parent names and address* ($\mathbf{G}_{NA}$), where the parents' names and their addresses are compared, and *Parent names only* ($\mathbf{G}_N$), where only the parent names are compared. The number of edges (corresponding to compared record pairs) are shown in Table 2.4.

- **NCVR**: As shown in Table 2.5, we generated a single pairwise similarity graph for the NCVR data set by comparing the name and address values. This graph contains 34,589,478 edges, as shown in Table 2.6. We used a blocking strategy which resulted in a pairs completeness (PC) of only 0.821 for the NCVR data set because attempting to further improve it significantly increases the similarity graph size and further reduces pairs quality. Since such a low PC value can negatively impact the linkage precision of the RL classification step, we included all true matching record pairs in the NCVR pairwise similarity graph.

- **Bibliographic data sets**: As shown in Tables 2.6 and 2.5, we generated similarity graphs for the DBLP-ACM and DBLP-Scholar bibliographic data set pairs by comparing the title, venue, author names, and year attributes.

2.7 Summary

In this chapter, we have highlighted the background related to RL, discussing its history and the major steps involved in the RL process. We focused specifically on the background information corresponding to the linkage of complex population data which is the scope of our research. We provided detailed descriptions of the specific methods used in this thesis, corresponding to steps in the RL process. Furthermore, we described the data sets we use for experiments throughout this thesis, and the pairwise similarity graphs generated for each data set. In the next chapter, we will be discussing related research work conducted on RL as relevant to our research questions.

Chapter 3

Related Work

In this chapter, we discuss existing research conducted in the area of Record Linkage (RL) related to the research questions which we aim to address in this thesis, as we highlighted in Section 1.2. We categorise this chapter into six sections as aligned with our research questions. We first discuss state-of-the-art unsupervised, supervised, and semi-supervised classification techniques in Sections 3.1 to 3.3 which are applied in the classification step of the RL process. We mostly focus on unsupervised and semi-supervised classification approaches given that we aim to address the RL problem where none or only limited training data (ground-truth) is available. In Sections 3.4 and 3.5 we then discuss efficiency improvement methods and evaluation measures developed to assess RL techniques. Next, in Section 3.6, we discuss modern anonymisation techniques for graphs since anonymising sensitive data sets which are represented as graphs is one of our research problems. Finally in Section 3.7 we conclude this chapter with a summary of the presented literature.

3.1 Unsupervised Classification for Record Linkage

As discussed in Section 2.3.4, unsupervised RL methods such as clustering are often applied on a pairwise similarity graph to conduct the final classification of record pairs as matches and non-matches. The advantage of using unsupervised techniques for classification in RL is their applicability in real-world linkage projects where ground-truth data is not available. In this section, we present state-of-the-art unsupervised classification techniques for RL.

To summarise work in this area, Hassanzadeh et al. [86] developed a framework offering a range of clustering algorithms adapted for applications in the RL context. Saeedi et al. [153, 154] proposed several advanced clustering approaches for multi-source RL, whereas Draisbach et al. [58] proposed three clustering techniques for RL that consider only the structure of the pairwise similarity graph representing the data sets to be linked. Furthermore, the literature comprises advanced unsupervised group RL approaches as proposed by On et al. [134] and Antonie et al. [3], and rule based methods proposed by Kouki et al. [105]. These state-of-the-art unsupervised RL approaches take record pair similarities, their relationships, and many other factors into account in the record pair classification step as we discuss in detail below.

- Hassanzadeh et al. [86] introduced a framework named Stringer, which can comparatively evaluate the linkage quality of RL techniques. The following clustering methods that can be used for RL were empirically evaluated using 29 synthetic and real-world data sets. Apart from the following clustering methods, the transitive closure (also known as partitioning or connected components technique [86]), where a sub-graph in which any two vertices are connected to each other is identified as a cluster, was used as a baseline.
  - Centre Clustering [88]: In Centre Clustering, the record pairs (r_i, r_j) corresponding to the edges in the similarity graph are sorted in descending order according to their overall pairwise similarities $s_{i,j}$ (calculated as defined in Equation. 2.1), and processed iteratively. If both vertices (where a vertex represents a record) are unassigned to a cluster, one is assigned as a cluster centre and all other vertices that are connected to the centre record with high similarity are assigned to that cluster.
  - Merge Centre [87]: Similar to Centre Clustering, however, clusters are merged if a vertex has high similarity to the centres of other clusters.
  - Star Clustering [6]: In Star Clustering, the degree sequence (number of neighbours per vertex) is considered for selecting vertices with larger degrees as cluster centres. All vertices highly similar to a centre vertex are assigned to the corresponding star cluster, whereas a post-processing step resolves cluster overlaps.
  - Ricochet Family of Algorithms [176]: Wijaya and Bressan proposed four algorithms under this family, which are known as Sequential Rippling (SR), Balanced Sequential Rippling (BSR), Concurrent Rippling (CR), and Ordered Concurrent Rippling (OCR). SR is similar to star clustering, however, instead of considering the degree sequence, the average similarity of edges connected to each vertex is considered for selecting centres. In BSR, the centre vertices are selected such that the ratio between the average edge weight of a centre to the sum of its similarity to the centres of other existing clusters is maximised. In CR, all record pairs (edges) with an overall pairwise similarity above a given threshold δ ($s_{i,j} > \delta$) are sorted in descending order of their similarity, and the corresponding vertices which are not yet processed are assigned as cluster centres. If an edge connects a centre to a non-centre vertex, the non-centre vertex is added to the centre cluster. If an edge connects two centre vertices, the cluster corresponding to the centre vertex with the lower average similarity of adjacent edges is merged into the other cluster. OCR is similar to CR, however, instead of using a threshold δ to select the edges to be processed a simple sorting of edges by their similarity is used.
  - Correlation Clustering [8]: In Correlation Clustering, the edges in the input graph are marked as positive or negative, based on the pairwise similarity $s_{i,j}$ being above or below a similarity threshold δ. The aim of Correlation Clustering is to maximise the number of positive edges within a

cluster and the number of negative edges between clusters, which is an NP-hard problem. An approximate algorithm introduced by Bansal et al. [8] was used in the Stringer framework.

- Markov Clustering (MCL): A technique originally proposed by Stijn van Dongen [57]. MCL performs several random walks on a graph to identify areas with high edge density, which correspond to a cluster. Hassanzadeh et al. [86] were the first to introduce MCL for RL clustering, which was also shown to be among the most efficient and effective algorithms for the RL task based on their experimental evaluation.
- Cut Clustering: Cut Clustering uses the max flow-min cut theorem proven by Ford et al. [66] where the objective is to find partitions in the graph such that the sum of the pairwise similarities of removed edges is minimal. The Cut Clustering algorithm proposed by Flake et al. [65] was used in the Stringer framework.
- Articulation Point Clustering [9]: An articulation point in a graph is a vertex whose removal would result in disconnecting the graph. In this method, the articulation points are repeated in each sub-graph that is resulting from articulation point removal, and a post-processing step is applied to resolve cluster overlaps.

- Saeedi et al. [153] introduced a novel Apache Flink [28] based framework, known as FAMER, to perform multi-source RL. The FAMER framework initially generates a pairwise similarity graph for the data sets to be linked, as we described in Definition 2. Subsequently, clustering is conducted in a distributed manner on this graph. The clustering algorithms supported by FAMER are Connected Component Clustering, Centre Clustering, Merge Centre Clustering, Correlation Clustering, and Star Clustering, as defined by Hassanzadeh et al. [86] as we previously described. For Correlation Clustering the parallel approach proposed by Chierichetti et al. [31] (referred to as CCPivot) was used in this research. Saeedi et al. proposed and supported a second version of the Star algorithm in FAMER where the average similarity of adjacent edges of vertices were used to identify cluster centres (whereas in the original version of Star Clustering the degree of vertices were used).

  The different clustering algorithms in FAMER were experimentally evaluated using three data sets from three different domains, where the largest data set (based on the North Carolina Voter Registration data) contained records of 10 million entities. Based on a linkage quality evaluation, the most robust clusters were shown to be produced by Centre Clustering, the new version of Star clustering, and CCPivot. With respect to run-times, CCPivot was shown to be the worst due to excessive memory requirements. The Star clustering version proposed by Saeedi et al. [153] was shown as the best option among the evaluated clustering schemes for multi-source RL due to its capability of generating high quality clusters with the shortest run-times.

- Saeedi et al. [154] then proposed two novel approaches for RL of multiple data sources. The first approach is called Clustering based on Link Priority (CLIP), whereas the second is entitled Repair based on Link Priority (RLIP). In the CLIP clustering approach the edges $\mathbf{E}$ in a pairwise similarity graph generated for the data sets to be linked, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, are categorised as strong (if the edge $e_{a,b} \in \mathbf{E}$ connecting vertex $v_a \in \mathbf{V}$ and vertex $v_b \in \mathbf{V}$ has the highest pairwise similarity among all edges associated with both v_a and v_b), normal (if the similarity of edge $e_{a,b}$ is highest only for either v_a or v_b), or weak otherwise. The CLIP algorithm generates non-overlapping, source consistent clusters, meaning that only a single record from each data set is contained in each cluster. A graph containing only strong links is initially processed to find clusters, and subsequent to removing those clusters, the remaining graph is processed with strong and normal links.

  The RLIP algorithm repairs the clusters generated by other clustering algorithms (such as Star clustering), by correcting cluster overlaps and source inconsistencies. To resolve cluster overlaps, a record appearing in more than one cluster is assigned to the cluster with which it has a strong link. When an overlapping record has strong links to several clusters, the cluster with which the record has the highest association degree (average similarity of a record with other vertices in the cluster) is chosen. Overlapping records with no strong links are assigned as singleton clusters. If an overlapping record is strongly connected only with other overlapping record(s), those are ignored in the first iteration and attempted to be resolved in the second iteration, or assigned as singletons if unsuccessful. The CLIP and RLIP algorithms were evaluated using three data sets from three domains (same as in [153]) whereas CLIP and RLIP were shown to produce considerably improved results, but with less scalability compared to other all other clustering algorithms used in the FAMER framework [153].

- Draisbach et al. [58] proposed three novel clustering techniques to conduct unsupervised RL by considering only the structure of the pairwise similarity graph but not the edge weights (record pair similarities $s_{i,j}$ which we discussed in Definition 1). Definitions of the proposed novel clustering techniques are as follows:

  - Maximum Clique Clustering (MCC)
    MCC is an iterative process, where the maximum clique [19] of a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is initially isolated and then all the vertices (which corresponds to a record $r_i \in \mathbf{V}$) belonging to that maximum clique is removed from the original graph $\mathbf{G}$. These two steps are repeated until all records are assigned to a maximum clique. The identified maximum cliques represent the final clusters.
  - Extended Maximum Clique Clustering (EMCC)
    EMCC is an extended version of MCC which is useful when a clique can

be extended by including few missing edges (which correspond to record pairs $(r_i, r_j) \in \mathbf{E}$). The first step in EMCC identifies the maximum clique. When multiple maximum cliques appear, the maximum clique that contains most edges to vertices which do not appear in the maximum clique is chosen, since that is the most likely to be extensible. The second step iteratively identifies the records $r_i \in \mathbf{V}$ which are eligible to join a cluster by having at least a threshold percentage of edges in the cluster. These two steps are executed iteratively until all vertices get assigned to some cluster.

– Global Edge Consistency Gain Clustering (GECG)
GECE clustering is based on the consistency of records linked during the graph construction phase. It takes into account all possible sets of vertex (or record) triangles in the graph. A triangle with records r_i, r_j and r_k would be inconsistent if there exist edges (r_i, r_j) and (r_j, r_k) but not (r_i, r_k) (meaning that transitivity does not hold in this triangle [35]). To resolve such inconsistencies, for each edge of each triangle in the graph, an edge switch is done (i.e. switching from link to non-link or vice versa). For each such switch, the consistency gain (difference between the number of consistent triangles after and before the edge switch) is calculated. If the consistency gain is positive, the prior mentioned steps of consistency calculation is repeated until a clique is obtained. Such cliques would be identified as clusters. If no consistency gain is obtainable by edge switch, all vertices are assigned to the same cluster.

An experimental evaluation conducted on seven real-world and synthetic data sets showed that EMCC was the overall best approach compared to MCC, GECG, and other baselines except Markov clustering which was proposed by Hassanzadeh et al. [86] as we discussed earlier. EMCC and Markov produced comparable results where EMCC led to higher precision and Markov clustering led to higher recall values.

- As discussed in Section 2.3.4, group linkage techniques have received considerable attention due to their applicability in linking groups of individuals, such as families or households. On et al. [134] introduced a group linkage method based on maximum bipartite graph matching (B_{S,δ_r}), where S refers to the pairwise similarity calculation method used and δ_r refers to the record level similarity threshold. The aim of the suggested approach is to find the top k matches from a list of target groups $\mathbf{H}'$ when a query group $\mathbf{h}$ is given. Note that each of these groups contains records referring to different entities. The best match(es) were selected such that $B_{S,\delta_r} \geq \delta_t$ where δ_t is a given similarity threshold. The record level similarity for query group $\mathbf{h}$ and a target group $\mathbf{h}' \in \mathbf{H}'$ is initially calculated using the Cosine similarity with Term Frequency-Inverse Document Frequency (TF-IDF) weighting [78]. The maximum bipartite graph matching is subsequently calculated using:

$$B_{S,\delta_r}(\mathbf{h},\mathbf{h}') = \frac{\sum_{(r_i,r_i')\in\mathbf{M}}(sim(r_i,r_i'))}{|\mathbf{h}|+|\mathbf{h}'|-|\mathbf{M}|}, \tag{3.1}$$

where $|\mathbf{M}|$ is the number of matching record pairs with similarity greater than threshold δ_r in a maximum weight bipartite matching. Since calculating B_{S,δ_r} is expensive due to the quadratic complexity in record comparisons [134], the authors introduced an upper and a lower bound for $BM_{sim,\rho}$, which together remove many record pairs in an earlier phase. Furthermore, if the maximum similarity between two vertices in the bipartite graph is less than threshold δ_t, such links are disregarded in Equation 3.1. This group linkage method based on maximum bipartite graph matching was shown to produce better recall when compared to a group linkage technique using the Jaccard similarity, achieving an average recall improvement of 16% to 17% when synthetic records and errors were present in the data set.

- Antonie et al. [3] proposed a group linkage approach for population data, which automatically identifies households in a census data set. A household constitutes of a group of people residing in the same dwelling. The suggested RL technique was based on the assumptions that the household head's record should precede the records of other household members, members of the same household must share the same district, household members from the same family must share the same surname, and the Numerical Personal Identifiers (PID) of members from the same household must be sequential. The records in a census data set were sorted by PIDs and district, and if record r_i shared the same district and surname with the previous record r_{i-1}, they were grouped into the same household. For records that could not be resolved, approximate string similarity calculations [35] were applied on the district and surname values to check for approximate similarity to consecutive household members using a sliding window approach.

  The proposed method was applied on the 1891 Canadian census, where the households corresponding to 99.1% of records (individuals) could be identified. A small subset (50 households) of results reviewed by domain experts were identified to be 100% accurate whereas the statistics of the generated households was found to be very similar to the aggregate information provided by Statistics Canada. However, the quality of this approach cannot be guaranteed provided that only a very small subset of households were reviewed by experts, compared to the very large data set of the total Canadian census, which comprised of approximately 4.8 million records.

- Kouki et al. [105] presented a novel collective RL technique to build familial networks based on the information (reports) gathered from different family members. The characteristics of this information comprises of statistical signals (comparison of attribute values), relational information (relationships among

family members), transitivity [35] (if x is the same entity as y, and y is the same entity as z, then x has to be the same entity as z), and bijective nature (there could only be a one-to-one match among mentions from two reports). Furthermore, to use relationship information for RL, a relationship normalisation was conducted which resulted in learning family relationships by the perspective of different mentions in a report.

The model proposed by Kouki et al. [105] comprised of seven rules, such as the name similarity rule which defines how the name similarity among mentions could be used to identify whether they refer to the same entity. This model outperformed the supervised learning techniques Logistic Regression, Logistic Model Trees, and Support Vector Machines (SVM) on both effectiveness and efficiency, in all the experiments conducted on two real-world data sets.

Even though the state-of-the-art unsupervised RL methods we have discussed above are shown to produce commendable linkage results, none of them is able to incorporate data characteristics that are commonly available in population data (such as temporal and spatial constraints) into the classification strategy. This is partially due to the lack of customisation of unsupervised algorithms to cater to the problem of linking population data. To address this research gap, we propose novel linkage algorithms in Chapters 4 and 5 which incorporate data characteristics with the aim of improving linkage quality.

3.2 Supervised Classification for Record Linkage

Supervised classification techniques such as Supervised Machine Learning (ML) and Supervised Deep Learning (DL) techniques (where DL is a subarea of ML) are widely applied for a number of analytic purposes in the world of science, whereas such techniques have shown promising results in many applications including the RL domain [55, 60]. However, the lack of ground-truth data in many RL projects hinders the application of supervised classification methods for linking records. For a comprehensive understanding of existing RL methods however, we next provide a summary of selected state-of-the-art supervised classification techniques for RL, and then discuss them in detail.

Among a range of powerful supervised learning approaches proposed for RL, the most distinctive and widely acclaimed are the Magellan tool by Konda et al. [103] offering different ML techniques for conducting RL tasks, and DL approaches proposed by Gottapu et al. [76], Ebraheem et al. [60], Mudgal et al. [120], and Li et al. [109]. The methods use record pairs labelled as matches and non-matches to train ML and DL techniques to effectively predict the class of a new record pair.

- Gottapu et al. [76] proposed a novel hybrid machine-human approach (where records were linked using a DL approach and any ambiguous classifications were reviewed by humans) for RL. A Convolutional Neural Network (CNN) was used as the learning algorithm since it has the advantages of not requiring

many attributes in the data set and not requiring to perform pairwise record comparison. The data set initially needs to be pre-processed by removing stop words [121], matching keywords with unique indices (integers) then representing records as a concatenation of their indices (with padding such that all records have the same length), and representing each keyword as a unique high dimensional vector of integers using word embedding.

When a pre-processed data set is given as input to the CNN, a sliding window maps the indices within the window to a class label, based on the high dimensional vector corresponding to those indices. Next, a softmax [16] function is used to calculate the classification probabilities for each record where a record is assigned to the class (match or non-match) with the highest probability. The authors used crowd-sourcing to review the records with uncertain labels. In experimental evaluations the proposed method was compared with a Jaccard baseline technique using a Boeing data set of sensor information, consisting of 1,950 records. CNN outperformed the Jaccard baseline in terms of linkage quality by classifying 55% of the records correctly as opposed to only 41% with Jaccard. However, CNN was less efficient (8.6 mins) than the Jaccard based technique (1.27 mins).

- The Magellan tool proposed by Konda et al. [103] is the most comprehensive and widely used state-of-the-art end-to-end RL system developed to date [60, 120, 166]. Magellan is distinctive from all other existing tools for RL such as D-Dupe, DuDe, Febrl, and IBM InfoSphere BigMatch [35] since (1) Magellan provides guidance to users to tackle RL problems, (2) it provides all the necessary tools to conduct a RL project end-to-end (not only for blocking and matching as provided by other RL tools), (3) it is built on top of the Python data science stack allowing the advanced learning methods, visualisation capabilities available in Python to be used in the RL task, and (4) Magellan allows users to introduce their own functions into the system.

  Magellan is primarily considered as a RL tool that supports supervised classification. It currently supports linking a pair of data sets using (1) supervised learning techniques, (2) rule-based methods, and (3) supervised learning with rules, where these techniques are made available through Python. The interface made available in Magellan allows users to conduct tasks such as debugging and selecting record pairs for training in a very interactive manner. Users can initially load and sample a subset of the data set to explore and select the methods to use in the RL workflow. The sampled data can be cleaned and visualised with Python packages such as pandas and matplotlib, which are made available through Magellan. Subsequently, blocking can be conducted in an iterative manner to select the best blocking method, and the user can then label record pairs for training also in an interactive manner. The best supervised learning technique and rules can also be selected interactively, where the user can improve classification by reporting errors via the debugger tool. Experiments conducted on data sets from twelve domains showed that the precision and re-

call achieved with Magellan ranged between 91.3% to 100% and 64.7% to 100% respectively, whereas the supervised baseline techniques resulted in precision from 56% to 100% and recall from 37.5% to 100%.

- Ebraheem et al. [60] proposed a novel DL based approach for linking records (known as DeepER), which interestingly uses minimal human effort for obtaining ground-truth data by considering prior knowledge of matched values. The authors used distributed representation (DR) of record attribute values, since DR allows to capture syntactic and semantic similarities and uses less memory. In DR, each word in a vocabulary is mapped to a high dimensional vector space with fixed length. Among the numerous methods of computing DRs of words, the authors chose Global Vectors for Word Representation (GloVe) [140], since it was observed that in GloVe the probability of co-occurrence of a word pair has a relationship with the meaning of the pair.

  The authors proposed two methods of converting records to DRs. The first method is the *simple averaging approach* where for each attribute value, the d-dimensional vectors of the words in that attribute value (as obtained from GloVe) were averaged. Therefore, if the attribute list corresponding to a data set is $\mathbf{A}$, the dimension of the DR of a record is $d \cdot |\mathbf{A}|$ as obtained with the simple averaging approach. The second method is the *compositional approach* which takes the word order (linguistic structures) into account to find the DR of a record. For this purpose, the authors used recurrent neural networks (RNN) with long short term memory (LSTM) hidden units, which learn long range sequential dependencies among words and generate a DR for each record, where the length of the DR is as determined by the LSTM. The authors used Locality Sensitive Hashing (LSH) based blocking [107], whereas pairwise record similarities were calculated using the Cosine similarity, vector difference (subtracting vectors), or the Hadamart product [48] (multiplying) depending on the DR generation method used.

  The proposed technique was evaluated against Magellan [103] as previously described, and other non-learning, learning, and crowd-sourcing based approaches [47, 75, 104]. In the experiments conducted using six different data sets, the proposed technique exceeded the quality of all baseline methods except in two instances, where the performance of one baseline method was marginally better (F-measure of 88.06 versus 89.3 and 97.67 versus 98.87 respectively).

- Mudgal et al. [120] presented the first ever comprehensive analysis of the application of DL for conducting RL, where they proposed four DL solutions of varying complexity, namely SIF (an aggregate function model), RNN (a sequence-aware model), Attention (a sequence alignment model), and Hybrid (a sequence-aware with attention model). The architecture proposed by Mudgal et al. for DL solutions for RL has three primary modules. (1) In the *attribute embedding module* word embedding vectors are generated for record pairs con-

sidering the words contained in attribute values. (2) In the *attribute similarity representation module* a vector of similarities is calculated using the word embedding vectors reflecting the similarity of the corresponding record pair. This module comprises of two steps, (2.1) *attribute summarisation* which aggregates the information in the word embedding vectors, and (2.2) *attribute comparison*, which applies a similarity function on the summarised vectors to obtain the similarity vectors. (3) Finally, in the *classifier module*, a record pair is classified as a match or a non-match based on the corresponding similarity vectors.

The four DL solutions are defined based on the techniques applied for steps (a) and (b) above. In the simple SIF model, a *weighted average* method is used for step (a) where word embedding vectors are summarised by calculating a weighted average, and an *element-wise absolute difference* method is applied for step (b). The medium complexity RNN method is different from SIF in using a *bidirectional RNN* for step (a) which takes the order of words into account when summarising embedding vectors. The attention model is also of medium complexity which uses *decomposable attention* in step (a) and *vector concatenation* in step (b), where both input sequences are analysed jointly while learning a similarity representation. The hybrid model with highest representational power uses a combination of these methods where step (a) is conducted using *a bidirectional RNN with decomposable attention* and step (b) using *vector concatenation augmented with element-wise absolute difference*.

The authors conducted experiments on eleven structured data sets, six textual data sets, and six dirty data sets, and compared the proposed DL methods with Magellan [103]. It was shown that the DL methods significantly outperform Magellan on textual and dirty data (3% to 22% and 6.2% to 32.6% F-measure improvements respectively) while for structured data the performance was on par with Magellan. Furthermore, the hybrid model was shown to be the best at exploiting the information encoded in training data, among the proposed DL models. The authors have publicly released this solution as a Python module names DeepMatcher.

- Li et al. [109] proposed a novel RL system based on pre-trained transformer-based language models, which is known as Ditto (Deep Entity Matching with Pre-Trained Language Models). Pre-Trained Language Models (LM) are deep neural networks that are pre-trained on large text corpus in an unsupervised manner. LM have the ability to learn the semantics of words better than conventional word embedding techniques such as Word2Vec, GloVe, or FastText. To leverage the power of Pre-Trained LM for RL, the authors represent record pairs in two data sets to be linked as a pair of text token sequences, where each sequence comprises a series of attribute and token pairs. Unlike in the DeepER and DeepMatcher DL models, the serialisation process applied to convert record pairs to token sequences in Ditto does not require them to adhere to the same schema.

Ditto uses three optimisation methods. The first embeds domain knowledge into the system with *span typing* and *span normalisation*. In span typing Ditto embeds the type of attributes (such as year, product ID, and brand) in the serialisation which helps to reduce mismatches by for example not comparing the year with product ID. In span normalisation, different strings with the same meaning are converted to identical strings to have the same embeddings. The second optimisation method summarises lengthy token sequences using a TF-IDF-based summarisation technique such that only the most informative tokens are provided as input to the LM. In the third optimisation approach, data augmentation is used to generate additional training data, and to allow the model to learn from difficult examples.

Experiments conducted on 13 data sets show that Ditto outperforms other state-of-the-art RL systems such as Magellan, DeepER, and DeepMatcher (which we have previously discussed under this section), by up-to 29% with regard to the F-measure. When the proposed optimisation techniques are applied, the linkage quality is further improved by up-to 9.8%. Furthermore, Ditto was shown to achieve results comparable to existing state-of-the-art RL methods with at most half the number of labelled data, whereas the effectiveness of Ditto on linking real-world data sets was established with running experiments on two company data sets consisting of 789,000 and 412,000 records achieving a high F-measure of 96.5%.

Even though supervised classification approaches often result in high linkage quality, as shown with the state-of-the-art work presented in this section, as we discussed earlier it is often not feasible to obtain an adequate number of labelled record pairs (or ground truth data) to train supervised classification algorithms. We therefore propose unsupervised and semi-supervised classification techniques for RL in Chapters 4, 5 and 7.

3.3 Semi-supervised Classification for Record Linkage

As we highlighted in Section 3.2, even though the application of supervised learning techniques for linking records has produced promising results, due to a lack of ground-truth data supervised classification approaches often cannot be utilised in real-world linkage applications. Therefore, semi-supervised classification techniques such as active learning [169] and transfer learning [167] have gained attention in recent years due to the possibility of exploiting the strength of supervised learning methods where limited ground-truth data is available. While active learning approaches aim to effectively sample a subset of data for manual labelling such that high classification quality can be obtained by training supervised classifiers on the limited labelled data, transfer learning aims to use already available training data from a semantically related domain to classify a target data set. Tejada et al. [165] and Sarawagi and Bhamidipaty [155] conducted early work on the application of

semi-supervised learning approaches such as active learning for RL, whereas in this section we describe more recent developments in this research area.

To summarise the contributions on this topic, Li et al. [108] proposed a RL technique using clustering, where temporal aspects related to linkage are learnt from a domain expert, Christen et al. [42] and Primpeli and Bizer [141] proposed active learning strategies based on the informativeness of record pairs, and graph characteristics respectively, and Kasai et al. [100] introduced a novel method that combines active learning and transfer learning for applying on RL tasks. The issue of unstable prediction models that is caused by the inadequacy of labelled record pairs in the early iterations of active learning strategies was addressed by Primpeli et al. [142] in their novel active learning approach. Furthermore, Meduri et al. [116] in their recent work bridges the gap of the absence of a framework for conducting RL using numerous active learning approaches. We now discuss these recent state-of-the-art active learning approaches for RL in detail.

- Population data is often considered as temporal data, since the values of attributes corresponding to a single entity often change with time. For example, people may change their occupation over time and the address may change with individuals changing their residence. The seminal work conducted by Li et al. [108] on linking temporal records introduced the first known technique of utilising temporal aspects in record linkage. We consider this as a semi-supervised approach since even though the authors have used clustering for linking records, the temporal aspects are learnt or obtained by domain experts. Li et al. introduced the two concepts of *disagreement decay* which accounts for the fact that in temporal data sets, certain attribute values of the same entity could potentially change over time, and *agreement decay* which implies that over a long period of time it is more likely to find two entities with the same attribute value compared to a short time frame. As shown in the experiments conducted by the authors, these decay values are highly reliant on the distribution of attribute values across time for the respective data sets.

  Using the agreement and disagreement decays, Li et al. introduced a novel method of calculating the adjusted similarity between record pairs in a data set. To calculate the adjusted similarity, the agreement decay probability p_a and the disagreement decay probability p_d for each attribute are calculated, and the attribute value similarities are weighted by $1 - p_a$ and $1 - p_d$ respectively. That is, in the pairwise record similarity calculation which we described in Definition 1, the agreement decay is used to reduce the weight (reward) for similar attribute values and the disagreement decay is used to reduce the weight (penalty) for dissimilar values. Three clustering techniques utilising these adjusted similarities were proposed and experimentally evaluated using two real-world data sets. The proposed temporal clustering methods outperformed three non-temporal baseline techniques by Hassanzadeh et al. [86] (which are partitioning, centre and merge clustering which we described earlier in Section 3.1), achieving a best result of 0.98 precision and 0.97 recall.

- Kasai et al. [100] proposed a novel method which allows combining the potential of active learning together with the capabilities of transfer learning to utilise DL for linkage tasks. They first propose a DL framework which uses transfer learning and extend it to adopt an active learning strategy when limited training data is available. The authors justify the use of DL instead of ML methods (such as SVM and decision trees) since, unlike in DL, they require human effort in selecting classification features (feature engineering) which is expensive. In the DL framework, for each record pair in the data sets to be linked, the attribute value tokens are obtained for which word embedding vectors are generated through fastText [18]. Next, a recurrent neural network (RNN) is used to combine the word vectors to obtain the representation vectors for each attribute, which are compared element-wise and summed to obtain the similarity vector for each record pair. Finally, a two-layer multilayer perceptron (MLP) with highway connections [161] (called the *matching classifier*) is used to classify record pairs based on their similarity vectors, with the training objective to minimise the negative log-likelihood loss.

  To make this DL framework applicable in a transfer learning setting, the authors model a *data set classifier* with the same architecture as in the *matching classifier*, for predicting the source of the input record pair. A data set independent internal representation is created by adding a gradient reversal layer, such that the quality of transfer learning is less affected by properties specific to the training data set. When limited training data is available, the authors propose an iterative active learning method to sample ambiguous record pairs for manual labelling and high confidence record pairs to increase the data available for training. Ambiguity and high confidence of record pairs is reflected by high and low entropy of the classification probabilities returned by the DL model. To sample a balanced set from the ambiguous and high confidence pairs, the authors first split the classified data set based on the predicted class, and from each class the n record pairs with the highest and lowest entropy are selected (ambiguous and high confident samples respectively). Experiments conducted on six data sets showed that DL conducted with transfer learning and active learning combined produced the best linkage results when compared against six learning algorithms made available by the Magellan framework [103].

- Christen et al. [42] proposed a novel informativeness based active learning strategy for RL, which, unlike previous work on active learning [5, 14, 175] does not rely on specific learning methods for effective selection of record pairs for manual labelling. Rather, the proposed method selects record pairs for labelling based on the *informativeness* of record pairs that are already manually labelled. A record pair is considered *informative* if it is surrounded by both matching and non-matching record pairs rather than record pairs from just one class. The proposed method uses a pairwise similarity vector $\mathbf{s}_{i,j}$ to represent a record pair (r_i, r_j) where $\mathbf{s}_{i,j}$ is calculated as we defined in Definition 1. A set of such vectors corresponding to unlabelled record pairs was denoted as $\mathbf{L}_u$, and labelled

vectors as $\mathbf{L}_l$ (where $\mathbf{L}_u \cap \mathbf{L}_l = \emptyset$). This is an iterative approach where k record pairs are selected in each iteration for manual labelling such that a total of up to β record pairs (where $k \leq \beta$) are labelled at the end of the process

In the initial step of this method, record pairs (similarity vectors) are sampled from $\mathbf{L}_u$ using *stratified sampling* or the *farthest first* method. In stratified sampling, the similarity vectors (record pairs) are clustered and the vectors closest to the cluster centroids are sampled. In the farthest first method the vectors farthest from the labelled vectors in $\mathbf{L}_l$ are selected. In the next step, the *entropy* and *uncertainty* for each selected vector is calculated considering its proximity to labelled vectors in $\mathbf{L}_l$ within a diameter up to the closest labelled vector from the opposite class (which defines the search space). The entropy is high if a selected vector is close to both matches and non-matches in $\mathbf{L}_l$ whereas the uncertainty is high if there are fewer vectors from the same class as the selected vector in $\mathbf{L}_l$ within the search space. Subsequently, n vectors are sampled based on a threshold δ_t for the combined entropy and uncertainty value.

In the next step, the most k diverse vectors among the selected n vectors are chosen for manual labelling with a farthest first approach. Experiments on three data sets showed that the proposed approach significantly outperforms the active learning approaches Entropy [158], Smallest Margin [168], Clu-AL [175], and Uncertainty [119], while obtaining comparable results to supervised methods with a 99% reduction in labelling effort.

- Primpeli et al. [142] proposed a novel method to resolve the cold start problem that is often encountered in active learning approaches, where the absence of adequate labelled data in the early iterations of an active learning approach leads to unstable prediction models. The authors developed an automatic labelling strategy that is applicable in the RL context to circumvent this cold start problem. In the initial step of the proposed approach, an overall similarity $s_{i,j}$ is calculated for each compared record pair (r_i, r_j). Subsequently, *confidence weights* are assigned to each record pair based on a threshold δ, where the authors proposed a method named *elbow point* based on the cumulative histogram of similarity scores to determine δ.

  Let $\mathbf{L}_s$ be a list of overall similarities of record pairs and $s_{i,j} \in \mathbf{L}_s$ be the overall similarity of a record pair (r_i, r_j). Then the confidence weight $w_{i,j}$ of record pair (r_i, r_j) is calculated as:

$$w_{i,j} = \begin{cases} \frac{|s_{i,j} - \delta|}{\delta - min(\mathbf{L}_s)}, & \text{if } s_{i,j} < \delta \\ \frac{|s_{i,j} - \delta|}{max(\mathbf{L}_s) - \delta}, & \text{if } s_{i,j} > \delta \\ 0.0, & \text{if } s_{i,j} = \delta \end{cases} \tag{3.2}$$

  With this approach, each record pair (r_i, r_j) is automatically assigned a class label (*match* label if $s_{i,j} > \delta$ and *non-match* otherwise) and a weight $w_{i,j}$ which re-

flects the confidence of belonging to that class. These record pairs with weighting are used to bootstrap the active learning model by training a random forest classifier, using the confidence weights as training weights for learning. Next, in an iterative approach, the most confident match and non-match labelled record pairs are used to train five classifiers and record pairs which have the highest disagreement rate in classification are selected for manual labelling. Experimental evaluation showed the improvement of quality by 86% in the cold start phase and up to 3% in subsequent iterations compared to baseline active learning approaches that do not use unsupervised bootstrapping. We use this concept in the active learning based RL method we proposed in Chapter 7.

- Meduri et al. [116] proposed a novel benchmark framework for assessing different active learning strategies for RL applications. Active learning strategies for RL often employ an iterative approach of obtaining a small labelled data set (record pairs) from the oracle, training a classifier(s), and getting the most ambiguous record pairs labelled by the oracle. The proposed framework allows users to pair different classifiers and selection strategies (the method of selecting ambiguous record pairs) as appropriate. The classification methods supported by this framework are linear classifiers (such as SVM), tree-based classifiers (such as random forests), non-linear classifiers (such as feed-forward neural networks), and rule-based classifiers. The three types of selection strategies supported are query by committee (QBC), margin-based, and heuristic.

  In QBC, a committee of classifiers is used where several classifiers are initially trained with a small oracle labelled set. Record pairs with the highest labelling disagreement, which is calculated using the entropy among the classifier outcomes, are given to an oracle for manual labelling. In the margin-based approaches, linear and non-convex non-linear classifiers are used, where in the former the record pairs closest to the separating hyperplane are considered most ambiguous, and high ambiguity is reflected with a classification probability close to 0.5 in the latter. Likely False Positives / Negatives (LFP / LFN) is an example heuristic selection strategy. Record pairs are classified using rules and likely incorrect classifications are identified such as by using an attribute value similarity heuristic. For example, if a record pair was classified as a match, but if the majority of its attribute value pairs are highly different it could be identified as a LFP and be given to an oracle of manual labelling. The time and quality of active learning strategies were improved with blocking which removes record pairs unlikely to be ambiguous, and active ensemble of linear classifiers for margin-based methods, where rather than using a single linear classifier, an ensemble of linear classifiers were used to improve recall.

  An experimental evaluation on nine public data sets showed that several active learning strategies made available in this framework outperformed state-of-the-art RL approaches including DL strategies [120]. The random forest based QBC method significantly outperformed all other active learning methods and all the compared baseline techniques.

- Primpeli and Bizer [141] proposed a graph-boosted Active Learning method for Multi-Source Entity Resolution (ALMSER). Rather than using query by committee (QBC) and margin-based strategies where the former determines the most informative (ambiguous) record pairs based on the disagreement among classifiers, and the latter selects record pairs closest to a classifier's decision boundary as the most informative, the ALMSER method relies on graph characteristics such as graph transitivity and minimum cuts to discover informative record pairs.

  The first step in ALMSER is the initialisation of active learning for which the authors use an unsupervised bootstrapping method to circumvent the cold start problem [142]. For the first twenty iterations in active learning, ALMSER uses the state-of-the-art committee based query strategy HeALER [30] before switching to the graph-based query strategy since the graph-based model is highly unstable in the beginning. Next, a correspondence graph is generated with all records from multiple sources and edges between record pairs with a *match* label for manual classifications. The remaining record pairs are labelled as a match or a non-match with an associated confidence score as per the prediction by the base model. A minimum cut of the graph is executed for each non-matching record pair such that no paths exist between any non-matching pairs. Next, graph inferred labels are assigned to record pairs to identify incorrect base learner predictions, by deriving clean connected components (with a maximum size equal to the number of sources to be linked) in the correspondence graph.

  Record pairs with contradicting graph inferred labels and base learner predicted labels are identified as the most informative record pairs to be manually classified. This method allows to effectively capture record pairs incorrectly labelled as non-matches (with graph inferred match labels based on transitivity), and matches (with graph inferred non-match labels with the minimum cut approach). Subsequently, a boosted learner is trained with manual classifications and graph inferred labels from the clean components of the correspondence graph, and the process continues until a satisfactory accuracy is achieved for the testing data prediction. The authors conducted an experimental evaluation on five multi-source matching tasks (each containing four or five data sets to link) and compare ALMSER with the QBC active learning method HeALER and a margin-based strategy using an SVM classifier. Results showed that the ALMSER method outperforms all baseline experiments as per the classification quality.

As we discussed above, there are several existing semi-supervised classification approaches for RL. However, given this area is relatively new, and since existing research work does not cover how active learning approaches can be utilised for efficiency enhancement in the RL context, we propose an active learning-based filtering technique in Chapter 6, and an active learning-based classification approach for RL in Chapter 7, with the aim of further expanding the literature on this topic.

3.4 Efficiency Enhancement in Record Linkage

As we discussed in Section 2.3.2, the quadratic complexity of naïve record pair comparison in RL has resulted in the requirement to apply methods such as indexing and blocking to improve the linkage efficiency by reducing the number of record pair comparisons [138]. Apart from indexing and blocking techniques, filtering methods are sometimes used to efficiently identify record pairs that are likely to satisfy predetermined similarity thresholds prior to applying costly comparison functions [138]. In this section, we present methods proposed for efficiency enhancement of the RL process.

To summarise the work in this area, Gu and Baxter [80] proposed a method to improve the efficiency of the RL process by post-processing large blocks, whereas Efthymiou et al. [61] proposed scalable algorithms for conducting meta-blocking. Zhang et al. [184] developed a blocking technique based on supervised learning, and a novel indexing approach was introduced by Akgün et al. [2]. A benchmark of existing efficiency enhancement methods for RL was conducted by Caldeira et al. [26]. We now discuss these state-of-the-art efficiency enhancement methods for RL in detail.

- Gu and Baxter [80] proposed an adaptive filtering method to post-process large blocks to enhance the blocking efficiency. The method is adaptive since it adapts to the blocking method to determine the number of blocks to be filtered. A *filtering variable* (a single attribute value or a combination of attribute values) different from the blocking key is used to quickly filter the likely non-matches prior to applying an expensive comparison method.

  The authors used the approximate comparison method proposed by Gravano et al. [77] for conducting adaptive filtering using the properties of edit distance without actually calculating the edit distance. The two properties of this technique are as follows. The first property states that two strings $\mathcal{S}_i$ and $\mathcal{S}_j$ with an edit distance greater than a predetermined threshold δ, cannot have a length difference greater than δ. The second property is based on the fact that potentially matching strings must share a large number of common bi-grams. Considering that $\mathcal{B}_i$ and $\mathcal{B}_j$ contain all bi-grams of $\mathcal{S}_i$ and $\mathcal{S}_j$ respectively, if strings $\mathcal{S}_i$ and $\mathcal{S}_j$ have an edit distance within δ then the condition $\mathcal{B}_i \cap \mathcal{B}_j = max(|\mathcal{S}_i|, |\mathcal{S}_j|) - 2\delta + 1$ must hold. Experiments conducted on four synthetic and one real-world data set showed that applying filtering after blocking helped improve the reduction ratio up to 63% at the cost of pairs completeness reduction by 5%.

- Efthymiou et al. [61] proposed scalable algorithms for conducting meta-blocking using the MapReduce framework [50] for parallelisation. In meta-blocking, a *blocking graph* is generated where each record r_i in a data set $\mathbf{D}$ ($r_i \in \mathbf{D}$) is represented as a vertex, whereas record pairs contained in a single block $(r_i, r_j) \in \mathbf{b}$ are represented as edges. The edges are weighted to reflect the likelihood for a record pair to correspond to the same entity (such as by assigning higher

weights to record pairs occurring more frequently in blocks) whereas a pruning step removes record pairs that are unlikely to be matches. The meta-blocks are generated based on this pruned blocking graph which reduces *redundant* (repetition of the same record pair in different blocks) and *superfluous* (record pairs that correspond to non-matches) comparisons.

The authors proposed three methods to conduct parallel meta-blocking. (1) The edge-based strategy explicitly creates a blocking graph with all the information about edges and corresponding weights. (2) The comparison-based strategy generates an implicit blocking graph where information used for calculating the weights of edges is indexed rather than explicitly building a blocking graph. (3) The entity-based strategy requires no blocking graph at all, where for each record (that corresponds to an entity) the other records co-occurring with it in at least one block is stored. All three methods explicitly or implicitly provide the edge weights needed for pruning.

Four different pruning methods were explored, namely weighted edge pruning (WEP), cardinality edge pruning (CEP), weighted node (vertex) pruning (WNP), and cardinality node pruning (CNP). In WEP and CEP, edges are pruned (filtered) based on a weight threshold or edge count threshold respectively, whereas for WNP and CNP, the filtering is done based on the average edge weight or edge count of a node's neighbourhood. The authors also proposed a new load balancing algorithm named *MaxBlock* which dynamically splits the input blocks into partitions for parallel processing. Experiments conducted on four data sets showed that on average, record pair comparison with meta-blocking took 95.5 minutes whereas without meta-blocking it took 14 days (a 93% improvement in efficiency). Furthermore, it was shown that the comparison-based strategy worked either with WEP and CEP whereas the entity-based strategy worked with WNP and CNP, and that these two methods were more efficient than the edge-based strategy in all experiments.

- Zhang et al. [184] proposed a novel learning based blocking method entitled *AutoBlock* which frees users from laborious data cleaning and blocking key tuning tasks while being efficient and scalable. The authors highlight the inability of key-based blocking techniques to handle dirty data due to reliance on exact attribute values. Even min-hashing based LSH can only conduct blocking based on the lexical similarity of record pairs but not the syntactic or semantic similarity. The AutoBlock method overcomes these issues by using a similarity-preserving representation learning method and nearest neighbor (NN) search.

  The similarity-preserving representation learning method comprises of four steps. In the (1) token embedding step, each record is represented as a list of tokens of attribute values, and the tokens are embedded using the fastText algorithm [18]. Next, an (2) attribute embedding is generated using the token embeddings for which the authors propose an attention-based attribute encoder. In this method attribute encoding is conducted by obtaining a weighted average of the corresponding token embeddings, where the weights are learnt

using a neural network depending on a token's semantics, position and surrounding tokens. Subsequently, a (3) tuple signature (signature of a record) is generated based on the attribute embeddings such that signatures for matched tuples have large Cosine similarity. The authors generate multiple such signatures for each record pair to mitigate the issue of obtaining low Cosine similarity due to lack of data quality (such as missing values). Then (4) model training is conducted with the objective of maximising the differences of the Cosine similarities between the tuple signatures of matched pairs and between unmatched pairs.

A fast NN search is then applied after computing the signatures for all tuples using the trained model. The authors use an LSH family for cosine similarity to retrieve the nearest neighbors of each record to generate the blocks. Experiments were conducted on three real-world data sets, where AutoBlock was compared with traditional key-based blocking, min-hashing based LSH blocking, and blocking as conducted in the DeepER system [60] (which we described in Section 3.2). It was shown that AutoBlock surpasses all baseline methods on the pairs completeness measure (see Section 2.4.2) by 18.3% to 25.8%. Furthermore, the AutoBlock method was shown to have sub-quadratic time complexity thus being scalable to millions of records.

- Akgün et al. [2] introduced a novel indexing technique for improving the efficiency of the RL process. They have introduced how Metric Space Indexing (MSI) can be applied in the context of RL. A noteworthy significance of MSI is its capability of combining three RL phases: indexing, pairwise comparison, and classification, into one. The M-tree [44] data structure was utilised as the MSI algorithm in this research.

  MSI was evaluated with three other baseline indexing methods, namely brute force (comparing each records pair), traditional linkage (blocking on attribute values) and the Locality Sensitive Hashing (LSH) technique which we described in Section 2.3.2. Overall, MSI outperformed both the LSH and traditional blocking techniques on precision and recall, whereas with MSI the same quality as for brute force was achieved but with considerably fewer comparisons. With certain parameter settings the precision achieved with LSH and traditional linkage was better than the precision achieved with MSI, since highly similar non-matching record pairs were captured by MSI but not by the other techniques (which resulted in more false positives for MSI and thus the low precision value). However, the precision improvements for the baseline techniques was achieved at the cost of considerable recall degrade, thus resulting in MSI being the overall superior approach. MSI is a desirable indexing approach due to its capability of combining three RL phases, little requirement of domain knowledge, and its capability to produce better RL results with improved precision.

- Caldeira et al. [26] conducted a comparative experimental evaluation of selected state-of-the-art filtering and meta-blocking techniques. As we described earlier,

these techniques aim to further improve the efficiency of RL by reducing the number of record pair comparisons resulting from the blocking or indexing step. Filtering approaches are based on the assumption that likely matches have high pairwise similarity. Methods such as filtering based on a threshold attribute value length difference (proposed by Gu and Baxter [80] as we discussed earlier), or retaining record pairs with common attribute value prefixes or suffixes, are used in this approach. Meta-blocking, on the other hand, attempts to reduce or eliminate redundant and superfluous record pair comparisons as we explained earlier in this section.

The authors evaluated the filtering techniques PPJoin [181], PPJoin+ [181], and AdaptJoin [171]. For each of these techniques the frequency of the attribute value tokens are calculated for each record, and the least frequent tokens are used as the *prefix* of each record. In the PPJoin technique, records are indexed (blocked) using these least frequent tokens, and the record pairs within a block are filtered based on a threshold size difference of the tokens. The PPJoin+ technique extends PPJoin by considering *suffixes* in the indexing, where all tokens excluding prefixes are classified as suffixes. AdaptJoin too is an extension to PPJoin, which dynamically calculates the prefix size based on the intuition that an adaptive prefix size can be more effective in filtering likely non-matches.

The meta-blocking approaches evaluated by Caldeira et al. [26] are Reciprocal Weighted Node Pruning (RecWNP) [137], Reciprocal Cardinality Node Pruning (RecCNP) [137], BLAST [160], and Blocking Process Based on Relevance of Terms (PBBRT) [25]. In all these methods, record pairs from the same block are represented as vertices in a graph with an edge connecting them, whereas an edge weight reflects the likelihood for a record pair to be a match. A threshold edge weight and a threshold edge count (cardinality) are used to filter record pairs in the RecWNP and RecCNP methods respectively. The BLAST method is a Locality Sensitive Hashing [107] based technique that determines the most informative attributes and accordingly improves the edge weights such that likely non-matches can be discarded. The PBBRT method uses the entropy of attribute value tokens to determine the likely matches to retain. An experimental evaluation conducted on four real-world and three synthetic data sets showed that even though filtering strategies performed well on smaller data sets, only meta-blocking techniques could efficiently and effectively reduce the comparison space in RL tasks [26].

As we discussed so far, all existing state-of-the-art efficiency enhancement methods proposed for RL attempt to improve the efficiency at the record pair comparison step. However, as we discussed in Section 1.2, the overall RL run-time is often dominated by the classification step due to the many record pairs retained for comparison even after blocking or indexing has been applied. Therefore, in Chapter 6, we propose a novel active learning-based filtering technique that aims to improve the efficiency of the RL classification step by removing likely non-matches that result from the comparison step.

3.5 Evaluation Measures for Record Linkage

In the existing literature the linkage quality of RL techniques are often assessed using evaluation measures such as precision, recall, and the F-measure (which we discussed in Section 2.4) as commonly used for assessing classification approaches in ML. However, as we also discussed in Section 2.4, these measures are sometimes problematic when applied in the RL domain, especially where group RL is considered. In this section, we explore existing work on alternative methods for assessing RL techniques.

To address the lack of evaluation measures to assess the quality of group RL techniques, Menestrina et al. [117] proposed a measure which determines linkage quality based on the number of cluster splits and merges required to convert the predicted clusters to the ground truth cluster set, while Hassanzadeh et al. [86] introduced two cluster measures which are based on the average number of record pairs that are correctly grouped together. Furthermore, in their recent work Hand and Christen [85] show how the widely used F-measure can be a misleading measure in the context of RL.

- Menestrina et al. [117] introduced a novel method to evaluate RL techniques, named the Generalised Merge Distance (GMD). This method was proposed for group RL techniques where a group (or cluster) of records is associated with a single entity. GMD uses a basic merge and split concept to assess RL techniques based on how many merges and splits are required to convert the clusters resulting from a RL technique to the gold standard (ground-truth) clusters. GMD allows the user to assign a cost to merges, f_m, and splits, f_s, as necessary, to reflect the desired outcome. Furthermore, it was shown that several existing evaluation measures, such as the F-measure (which we described in Section 2.4) and the variation of information (which calculates the information loss and gain when converting the predicted clusters to the ground-truth clusters) can be derived using the GMD measure by certain configurations of the cost functions f_m and f_s. Therefore, GMD has the potential of reflecting different aspects of the compared RL methods, with different cost metrics.

  Operation Order Independence (OOI) is an important aspect which needs to be adhered to by the chosen cost functions. The OOI property states that when there are three clusters, $\mathbf{c}_x, \mathbf{c}_y$ and $\mathbf{c}_z$, the total cost of first merging $\mathbf{c}_x$ with $\mathbf{c}_y$ and then merging the result with $\mathbf{c}_z$ is similar to the total cost of merging $\mathbf{c}_x$ with $\mathbf{c}_z$ and then merging the result with $\mathbf{c}_y$. In the GMD measure, a split first legal path (a path where splits occur before merges, and each intermediate cluster exists in the ground-truth) with minimum cost is identified by adhering to the OOI property. With an empirical evaluation conducted on two Yahoo data sets (related to shopping and hotels) the authors showed how different evaluation techniques come to different conclusions in the presence of varying types of error in the data sets. The GMD measure was shown to be superior to the other compared measures which combine the average similarity of predicted clusters

with respect to ground-truth clusters and vice versa due to its capability of reflecting different strengths of each RL method, based on the split and merge costs assigned.

- Hassanzadeh et al. [86] introduced two new evaluation measure to assess group RL techniques which are as follows.

  - **Clustering Precision (*CPr*)**: *CPr* reflects the average number of record pairs which are correctly grouped together in a single cluster. For each predicted cluster, the record pairs in the cluster are identified. Then the number of such pairs which occur together in a single cluster in the ground-truth set is calculated. *CPr* is the average number of such pairs, across all clusters which have at least two elements. Let k and k' denote the number of ground-truth clusters and the number of predicted clusters respectively. Let $\mathbf{C}_p$ denote the list of predicted clusters where each predicted cluster $\mathbf{c}_i^p \in \mathbf{C}_p$ contains at least two records, and let $\mathbf{c}_j^g$ denote a ground-truth cluster in the ground-truth cluster list $\mathbf{C}_g$ ($\mathbf{c}_j^g \in \mathbf{C}_g$). The authors present the clustering precision CPr_i for a predicted cluster $\mathbf{c}_i$ as;

$$CPr_i = \frac{|(r_x, r_y) \in \mathbf{c}_i^p \times \mathbf{c}_i^p : x \neq y \wedge \exists j \in 1...k, (r_x, r_y) \in \mathbf{c}_j^g \times \mathbf{c}_j^g|}{\binom{k'}{2}} \tag{3.3}$$

$$CPr = \frac{\sum_{i=1}^{k} CPr_i}{|\mathbf{C}_p|} \tag{3.4}$$

    However, with the *CPr* measure, the number of clusters in the prediction and the ground-truth are not compared. Therefore, Hassanzadeh et al. proposed a variation of *CPr* which is as follows.

  - **Penalised Clustering Precision (*PCPr*)**: *PCPr* penalises any algorithm that creates fewer or a greater number of clusters than what occurs in the ground-truth data set. *PCPr* is calculated as;

$$PCPr = \begin{cases} \frac{k}{k'} CPr, & \text{if } k < k' \\ \frac{k'}{k} CPr, & \text{if } k \geq k' \end{cases} \tag{3.5}$$

- Hand and Christen [85] showed why the F-measure (F), can be an inadequate evaluation measure for assessing RL techniques, when viewed as a weighted arithmetic mean of precision (P) and recall (R). The F-measure formula could be rewritten as $F = \theta R + (1 - \theta)P$, where $\theta = (FN + TP)/(FN + FP + 2TP)$ and FN, TP, and FP represent the number of False Negatives, True Positives, and False Positives respectively as we discussed in Section 2.4. Therefore, the F-measure is actually a weighted mean of P and R, meaning that this weight θ must be equal for all RL methods being compared, for the evaluation among

them to be fair. However, since in practical applications different RL approaches generate different numbers of predicted matches (TP + FP) on which the weight θ relies, using the F-measure for assessing RL methods is inadvisable.

Hand et al. [83] attempt to address this issue with the F-measure in their recent work from 2021. As per their analysis, another issue with the F-measure is that even though precision and recall can be considered as empirical estimates of the conditional probability of a correct classification given a predicted class and a true class respectively, their harmonic average (which is the F-measure) cannot be interpreted as a probability. Furthermore, the harmonic mean of two values lies closer to the smaller of the two, and when one is zero, the harmonic mean becomes zero as well, thus ignoring the value of the other value. To resolve this issue, Hand et al. proposed the F^*-measure which is defined as $F^* = F/(2-F)$ (see Eq. 2.5 for detail). F^* is the number of true matches identified against the misclassified and correctly classified matches [83].

Only a few works such as those discussed above have presented novel methods of evaluating group RL approaches and question the applicability of widely used measures such as the F-measure for assessing linkage results. In Chapter 8, we therefore present a novel evaluation measure for group RL, where the linkage quality is assessed based on how records are allocated into predicted clusters as compared to ground truth clusters. Furthermore, we discuss the limitations of existing evaluation measures such as precision and recall for assessing group RL techniques.

3.6 Graph Anonymisation Techniques

As we highlighted in the previous chapters, ensuring the privacy of sensitive data sets used for RL is highly important especially when population data is being linked, which is our focus in this thesis. Since the data sets to be linked are often represented as pairwise similarity graphs (as we discussed in Section 2.3.3) we explore graph anonymisation techniques for preserving the privacy of the data. Privacy-Preserving Record Linkage (PPRL) techniques [40] are excluded from this section since they are beyond our scope, as we highlighted in Section 1.5.

To summarise the work in this area, Zhou et al. [188] proposed graph anonymisation techniques for social networks where anonymisation is conducted by adding and/or deleting edges and vertices. Furthermore, novel graph anonymisation techniques were proposed by Wang and Li [173], and Delanaux et al. [51] as we discuss in detail below.

- Zhou et al. [188] conducted a survey of graph anonymisation techniques developed for social networks. They explained how anonymising graph data is more complex than anonymising relational data due to the presence of more information related to vertices and edges in a graph. Furthermore, they emphasised the importance of understanding the utility of graph data prior to anonymi-

sation, since otherwise an anonymised graph data set might be rendered of limited use for the analytic purpose.

Zhou et al. presented existing graph anonymisation techniques under two categories, clustering-based approaches and graph modification approaches. Clustering-based approaches can be further categorised based on whether vertices or edges are being grouped into a cluster, where a cluster is then represented as a super-vertex for anonymisation.

- Vertex clustering method: Hay et al. [89] proposed a vertex clustering method which mimics the k-anonymity [162] approach for relational data. Vertices are grouped such that each group contains at least k vertices, and the number of vertices in each group and the densities of edges within and across groups are published as the anonymised data set. The best grouping that maximises the utility and adheres to k-anonymity is identified using a maximum likelihood approach.
- Edge clustering method: Zheleva and Getoor [185] proposed an edge clustering method, where they focused on graphs having multiple types of edges, and only a single type of vertices. One of the edge types is considered to be sensitive. The vertices are initially grouped and collapsed into a single vertex and the edges between the collapsed vertices are reported, for example by publishing the number of edges of a given type. Since certain sensitive edges are removed in the process, the utility of this anonymisation approach is measured by the number of edge deletions.
- Vertex and edge clustering method: Campan and Truta [27] proposed a clustering technique for graphs where edges are not labelled but the vertices have attributes, some of which are sensitive. The k-anonymity model [162] was used to anonymise vertices. The super-vertex (group) is labelled with the number of vertices and edges in the cluster. Even though this approach is somewhat similar to the method proposed by Zheleva and Getoor [185], the method proposed by Campan and Truta has the flexibility of allowing the user to achieve a desirable trade-off between preserving more structural information (edge related) or preserving more vertex attribute information.
- Vertex attribute mapping clustering method: Cormode et al. [45] proposed a method to anonymise bipartite graphs, where the vertices are of two distinct types, and edges appear across vertices of these two types. For the two vertex types $\mathbf{V}_x \subset \mathbf{V}$ and $\mathbf{V}_y \subset \mathbf{V}$, where $\mathbf{V} = \mathbf{V}_x \cup \mathbf{V}_y$, the proposed method groups vertices of each type such that two vertices in the same group of $\mathbf{V}_x$ have no common neighbors in $\mathbf{V}_y$, and vice versa. Furthermore, the anonymisation granularity is controlled by two user defined parameters i and j where each group in $\mathbf{V}_x$ is constrained to have at least i vertices, and j vertices for each group in $\mathbf{V}_y$.

In the graph modification approached proposed by Zhou et al. [188] a sensitive graph is anonymised by inserting and/or deleting certain edges and vertices. Methods in this category can be further categorised as follows.

- Optimisation graph construction method: Liu and Terzi [110] proposed a graph modification technique to anonymise graph data for scenarios where knowledge of the degree of vertices is used by attackers to identify the people represented by target vertices. The k-degree anonymity method [162] is used such that for every vertex in the graph, there exist at least $k-1$ other vertices with the same degree.
- Randomised graph modification approaches: Several studies have used this idea for graph anonymisation. Hay et al. [90] proposed a method where anonymisation is achieved through deletion of n edges followed by insertion of n edges. Ying and Wu [183] proposed a method of randomly adding, deleting, or switching edges, while Liu et al. [111] proposed two methods for anonymising a graph where the edge weights were considered to be sensitive. The first method proposed used Gaussian randomisation multiplication and the second used a greedy perturbation algorithm.
- Greedy graph modification approach: Zhou and Pei [186] proposed a k-anonymisation based method for protecting graphs from neighbourhood attacks (where the adversary attempts to identify target vertices based on knowledge about their neighbourhood). They further extended their work in [187] to handle the l-diversity problem [1] which ensures that the sensitive attribute values of a record cannot be inferred with a confidence greater than $1/l$.

- Wang and Li [173] proposed a novel graph-based technique for anonymising sensitive data with attributes of multiple types, which they abbreviated as AMT data sets. The multiple types of attributes in an AMT data set are (1) the relational attributes which consist of *quasi-identifiers* (QIDs) $\mathbf{A}$, such as age and zipcode of individuals and sensitive information such as occupation and salary, and (2) transactional attributes such as products bought by a person or diseases they have, which can have some sensitive values. An AMT data set is initially presented as a graph $\mathbf{G}$ with $\mathbf{V}$ vertices and $\mathbf{E}$ edges. The set of vertices are categorised into four groups where $\mathbf{V}_i$ identifies records in the AMT data set (such as people), $\mathbf{V}_r$ identifies sensitive relational attribute values, $\mathbf{V}_{tn}$ identifies non-sensitive transaction attribute values, and $\mathbf{V}_{ts}$ identifies sensitive transaction attribute values. An edge $e \in \mathbf{E}$ connects a record vertex in $\mathbf{V}_i$ and a sensitive relational vertex in $\mathbf{V}_r$, or a record vertex and a non-sensitive transaction vertex in $\mathbf{V}_{ts}$.

  Their proposed method anonymises both sensitive relational and transactional attributes, while maintaining the information required to analyse relationships among them. The sensitive relational attributes are anonymised using a nearest neighbour based fuzzy clustering approach which is defined as follows. For

each QID attribute value $a \in \mathbf{A}$, assuming the distance between two records (vertices) $v_x \in \mathbf{V}_i$ and $v_y \in \mathbf{V}_i$ is given by $|v_x(a) - v_y(a)|$ (or the semantic distance for non-numeric values), each record is linked with the sensitive relational attribute value in $\mathbf{V}_r$ corresponding to the record with which it has the minimal distance. In this manner, each record would be associated with more than one sensitive relational attribute value thus ensuring privacy. The sensitive transactional data is anonymised as follows. For each sensitive transaction vertex $v_x \in \mathbf{V}_{ts}$, the probability of encountering v_x given a non-sensitive transaction $v_y \in \mathbf{V}_{tn}$ is calculated as the probability $p(v_x|v_y)$, which is represented with the directed edge from v_y to v_x. A similar probability calculation is applied for pairs of sensitive transactions. Since there are no direct edges connecting record vertices $\mathbf{V}_i$ and sensitive transactional vertices $\mathbf{V}_{ts}$ an adversary will not be able to directly associate a sensitive product with an individual (only a probability is given). An experimental evaluation on two widely used benchmark data sets showed that the method proposed by Wang and Li [173] resulted in minimal information loss (and therefore retains maximum utility) when compared with two k-anonymisation based techniques.

- Delanaux et al. [51] proposed a method to anonymise a Resource Description Framework (RDF) graph such that it is not vulnerable to attacks subsequent to being linked with an external RDF graph. RDF graphs store information in the form of *subject-predicate-object* (known as *triples*), where the subject is often used to identify individuals and the object can therefore reveal sensitive information, especially when combined with other triples. For example t_1 = <mary, specialist of, cancer> and t_2 = <bob, seen by, mary> are two triples which reveal sensitive information about Bob. Even if the triple t_2 is deleted in an RDF graph, if that anonymised graph is linked with an external RDF that contains triple t_2, the sensitive information about Bob can still be revealed.

  In the safe anonymisation model proposed by Delanaux et al. [51] every critical subject and object which can reveal information if used in conjunction with another triple, is replaced with an anonymised vertex. For example, the triples t_1 and t_2 would be anonymised as t'_1 = <b1, specialist of, cancer> and t'_2 = <b2, seen by, b1> respectively. The significance of this method is that it preserves the results obtained with statistical queries such as counting the number of people seen by 'b1'. This task can be achieved by running the RDF graph against a set of private conjunctive queries (which assess the privacy of a triple by joining it with a given triple), and performing the necessary insertions or deletions to a triple to anonymise it.

  Furthermore, the authors introduce a privacy query that can detect triples, where a subject or an object can be uniquely identified as representing the same element from an external RDF graph, and replace such subjects / objects with blank vertices. With an experimental evaluation conducted on three real-world RDF graphs, where the largest contained approximately 6.5 million triples, the authors showed that this graph anonymisation method can be conducted in

under one second for each data set. Furthermore, the precision loss of the data sets increases linearly with the number of blank vertices that are added to the graph for anonymisation.

With the exception of this last described work by Delanaux et al. [51], all other work we discussed generate anonymised graphs that cannot be interpreted by humans. Furthermore, existing graph anonymisation methods which we discussed in this section do not preserve the graph structure, which renders them inapplicable in the RL domain. In our work in Chapter 9 we highlight the need for anonymisation techniques that produce human understandable anonymised graph data, while also preserving the graph structure.

3.7 Summary

In this chapter, we presented the seminal and state-of-the-art methods for RL, focusing on supervised, unsupervised and semi-supervised classification techniques, efficiency enhancement techniques for RL, evaluation measures developed to assess linkage results, and graph anonymisation approaches. While high linkage quality has been achieved with supervised RL methods, these methods are often not applicable in real-world linkage projects due to the lack of availability of training data. Furthermore, existing unsupervised and semi-supervised RL approaches do not utilise the data characteristics (such as temporal and geographic information) available in population data in the linkage process, which we address with our contributions. We also contribute to the limited literature available on evaluation measure for assessing group RL techniques, and propose a novel anonymisation technique for ensuring the privacy of sensitive linked data. In the next chapter, we present three novel graph-based clustering techniques which utilise data characteristics to improve linkage quality.

Chapter 4

Graph-based Clustering for Record Linkage Using Data Characteristics

As we discussed in Section 1.2, the lack of availability of ground-truth data for real-world Record Linkage (RL) applications hinders the use of supervised learning techniques for linking records. In this chapter we address this limitation by proposing three novel unsupervised graph-based clustering techniques for group RL. These clustering approaches enhance linkage quality by utilising data characteristics such as temporal and spacial information available in the data.

In Section 4.1 we provide an introduction to unsupervised group RL techniques and how data characteristics available in population data can be utilised in these linkage approaches. Next, in Section 4.2, we discuss how data characteristics in population data sets can be modeled and incorporated in the pairwise similarity graph **G** where **G** is created as described in Definition 2. In Section 4.3 we describe the three clustering algorithms we propose in detail, whereas in Section 4.4 we empirically evaluate our proposed methods and compare them with RL clustering techniques that do not incorporate data characteristics. Finally, in Section 4.5, we provide a summary of the novel clustering techniques we presented in this chapter.

4.1 Introduction

Even though the applicability of supervised linkage approaches for RL has been extensively explored [60, 76, 103, 120], such techniques are often of less value in real-world RL projects due to the limited availability of training (ground-truth) data. Numerous unsupervised RL techniques [3, 86, 105, 108, 153, 154] have therefore been explored to address the linkage problem. The challenge with unsupervised learning, however, is the necessity of distinguishing between matches and non-matches when training data is unavailable to learn from. Therefore, domain knowledge of the data is often employed in unsupervised RL techniques to determine the characteristics of matching record pairs.

As we discussed in Chapter 2, group RL techniques have recently received significant attention in contrast to traditional pairwise RL due to their applicability in linking groups of individuals in population data sets, such as families and house-

holds [134]. The identification of relationships between individuals can enrich 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 [17]. Among the different unsupervised RL techniques explored in the literature, clustering approaches are most applicable when conducting group RL. Therefore, in this chapter we explore how groups of records can be linked using unsupervised clustering approaches.

As we highlighted earlier in this section, unsupervised techniques for RL such as clustering can employ domain knowledge of the data characteristics to identify records corresponding to the same entity. Since the scope of this thesis is to address linkage of population data, data characteristics such as temporal and spatial information inherent to population data can be employed to classify record pairs as matches and non-matches. While existing clustering methods have explored how relationships among entities [105], and similarities in the pairwise similarity graph $\mathbf{G}$ (see Definition 2) [86, 153, 154] can be manipulated to improve the clustering quality, data characteristics as implied by time and space have not been utilised in clustering for RL to the best of our knowledge. We therefore propose three novel graph-based unsupervised clustering techniques for RL which employ data characteristics with the aim to improve linkage quality.

4.2 Modelling Constraints Implied by Data Characteristics

In this section, we provide a formal definition of data characteristics in population data, and how it corresponds to the likelihood of a record pair being a match. Furthermore, we describe how we model the constraints implied by these data characteristics to be employed in graph clustering for RL.

Definition 3 (Data Characteristics) *Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a pairwise similarity graph (see Definition 2). For each record pair $(r_i, r_j) \in \mathbf{E}$, we define a data characteristic value c as a comparison between records r_i and r_j (where $r_i \in \mathbf{V}$ and $r_j \in \mathbf{V}$) with respect to a common attribute $a \in A$. Furthermore, a given data characteristic value c has a corresponding probability p associated with it, which reflects the likelihood (plausibility) for a record pair to be a match given value c.*

Following Definition 3, a data characteristic value c can be the distance between records r_i and r_j, where for data characteristics identified by a numerical attribute (such as the time and geocode [102] which identifies a location), the distance is calculated as the absolute difference in attribute values. If the attribute is a string (such as the occupation or publication venue) the distance can be calculated as the semantic difference in attribute values [151]. Domain experts can provide information about the constraints implied by different data characteristics which we need to model mathematically to be incorporated in our linkage algorithms.

Most population data sets contain a time related attribute such as the date of birth in birth data sets, and the publication date in bibliographic data. Therefore,

we present our concept of modeling data constraints using temporal characteristics. Temporal constraints can be modeled based on domain experts' knowledge to reflect the likelihood for a record pair with a given temporal difference to be a match. Temporal constrains between records can include that the birth record of a person must be before their death record, a marriage record can only occur once an individual has reached a certain age, or that the same mother can only give birth to several babies according to certain biological limitations (at least nine to ten months apart or within a few days for multiple births such as twins) [149].

We model such temporal constraints as a list $\mathbf{T}$ of time intervals where for each such interval it is plausible (or not) for two records to be linked (such as a mother to give birth to two babies). We assume each record $r_i \in \mathbf{V}$ includes a time-stamp, $r_i.t$, such as a date of birth, marriage, or death. Based on these time-stamps we can calculate the absolute temporal difference $\Delta t_{i,j} = |r_i.t - r_j.t|$ between two records. Note that here the temporal difference is the data characteristic value ($c = \Delta t_{i,j}$).

The list $\mathbf{T}$ contains time intervals and *temporal plausibilities* (which are probability values), p, where $p = 1$ means two records are temporally plausible and $p = 0$ means they are not, in the form of tuples $(\Delta t^{start}, p^{start}, \Delta t^{end}, p^{end})$, with $\Delta t^{start} < \Delta t^{end}$. For example, for birth records, $\mathbf{T}$ = [(0,1,2,1), (3,0,273,0), (333,1,12783,1), (14610,0,99999,0)] means that two births by the same mother up to two days apart are plausible, as are two births ten months to 35 years (274 to 12,783 days) apart, but not two births between three days to ten months or more than 35 years apart. The plausiblity list $\mathbf{T}$ is initially constructed based on domain experts' knowledge which is then expanded as follows.

We obtain all unique temporal differences Δt (or data characteristic values c) corresponding to all record pairs $(r_i, r_j) \in \mathbf{E}$. Then we identify the time interval (Δt^{start}, Δt^{end}) in $\mathbf{T}$ to which each temporal difference Δt belongs ($\Delta t^{start} \leq \Delta t \leq \Delta t^{end}$), and calculate the temporal plausibility, p, corresponding to Δt using linear interpolation as:

$$p = \begin{cases} p^{start}, & \text{if } \Delta t = \Delta t^{start}, \\ p^{end}, & \text{if } \Delta t = \Delta t^{end}, \\ (p^{end} - p^{start}) \cdot \frac{(\Delta t - \Delta t^{start})}{(\Delta t^{end} - \Delta t^{start})} + p^{start}, & \text{if } \Delta t^{start} < \Delta t < \Delta t^{end}. \end{cases} \tag{4.1}$$

If the calculated p is below a given minimum temporal plausibility threshold δ_p, then record pairs with the corresponding Δt time difference are deemed not to be temporally plausible and they will not be linked in the clustering algorithm, as we discuss further in the following section. Figure 4.1 illustrates temporal constraints applicable when identifying sibling groups (births by the same mother).

4.3 Graph-based Clustering Using Data Characteristics

In this section, we present three novel unsupervised graph-based clustering techniques which incorporate the temporal constraints we modeled in the previous section. As we described in Definition 2, the list of vertices $\mathbf{V}$ in a pairwise similarity

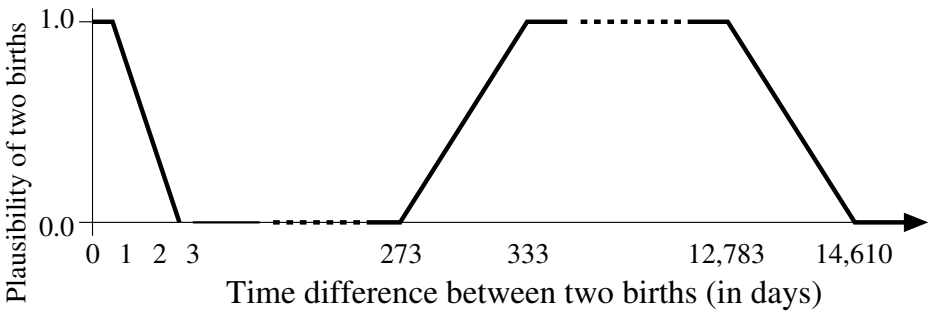

Figure 4.1: Temporal constraints as the plausibility for the same mother to be able to give birth to two children, where the horizontal axis shows the time difference Δt (in days) and the vertical axis the plausibility p that two birth records are possible for a certain time difference. Due to errors in registration dates, for multiple births we allow for a few days difference for twins and triplets, and then have a plausible interval between births from 10 months onwards up to 35 years. Two births by the same woman more than 40 years (14,610 days) apart is deemed not to be plausible.

graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ represent records from a data set $\mathbf{D}$ to be linked. Therefore, a vertex $v_i \in \mathbf{V}$ represents a record r_i (i.e. $r_i = v_i$) and in the clustering algorithms we refer to a record r_i as a vertex v_i, and a record time-stamp $r_i.t$ as a vertex time-stamp $v_i.t$. Note that all the clustering methods proposed in this chapter are unconstrained by the number of clusters (the clustering algorithms do not need the number of clusters to be specified as input), and they produce disjoint clusters (each record appears only in one cluster). That is, with our clustering approaches we aim to generate clusters where each cluster represents only one true related group of entities, and for each group of true related entities we have only one cluster.

We chose three clustering approaches named greedy, star, and CLIP clustering to extend with incorporating data characteristics. The star and CLIP approaches (we refer to CLIP with data characteristics incorporated as robust graph clustering) by Saeedi et al. [153, 154] were chosen due to them being reported as algorithms that surpass the performance of other state-of-the-art clustering algorithms in the context of RL. We propose greedy clustering as a simple baseline method for group RL. Note that even though the literature related to graph clustering comprises many algorithms, not all of them are applicable in the RL context due to the constraints applicable in RL tasks, such as the size of cluster growth not being linear with regard to the data sets size, and clusters being disjoint [86].

4.3.1 Greedy Clustering

The greedy clustering approach is based on the idea of iteratively adding vertices to clusters using a greedy selection method, as illustrated in Figure 4.2. We initially create one cluster per record, and insert these singleton clusters into a priority queue that is sorted according to time-stamps with the smallest time-stamp first. We then process the earliest cluster first, and aim to expand this cluster with a new record

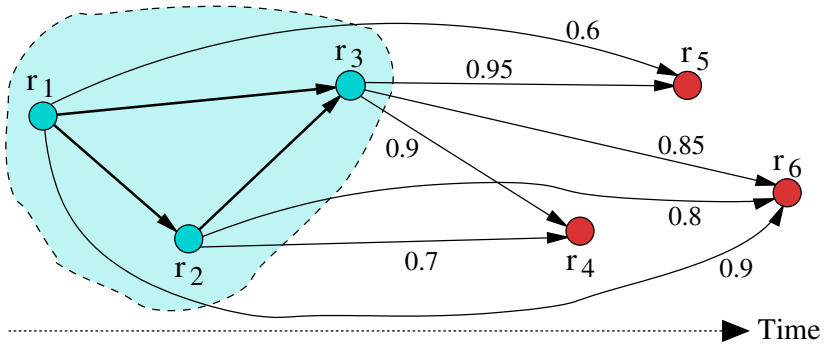

Figure 4.2: Example of the greedy temporal linkage approach described in Section 4.3.1, showing vertices (records) and edges (similarities) from the pairwise similarity graph **G**. Records r_1 to r_3 show an existing cluster, and the question now is which best future record (from r_4, r_5, and r_6) is to be added to the cluster next. We consider three selection methods: (a) the earliest next temporally possible 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).

that is in the future (of the latest record in the cluster), as Figure 4.2 shows. In this greedy approach the question is how to select the best future (next) vertex (record) to add to a cluster. We implement three different such selection methods to identify the next best record m_n:

- **Next**: Select the next record (temporally plausible if temporal constraints are considered) 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 vertices (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 overall similarity $s_{i,j}$ (see Definition 1) with any record in the cluster. This method generates clusters where vertices 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.

Algorithm 1 outlines our proposed greedy clustering approach where we can consider temporal constraints when selecting the next record to be added into a cluster, or we can ignore any temporal constraints.

The main input to the algorithm are the pairwise similarity graph, **G**, and a list of temporal constraints, **T**, as discussed in Section 4.2 (if **T** is empty, temporal

Algorithm 1: Greedy clustering

Input: **G** - Undirected pairwise similarity graph
T - List of temporal constraints (as discussed in Section 4.2)
δ_p - Minimum plausibility for record pairs to be considered
δ_s - Minimum similarity for a record to be added to a greedy cluster
m_n - Method on how to select the next vertex to add to a cluster
Output: **C** - Final list of clusters

1 $\mathbf{G}_D = \mathbf{GenerateTempDirGraph}(\mathbf{G}, \mathbf{T}, \delta_p)$ // A temporal directed graph
2 $\mathbf{C} = [\,]$ // Initialise an empty list of clusters
3 $\mathbf{Q} = [\,]$ // Initialise an empty priority queue
4 **for** $v \in \mathbf{G}_D.\mathbf{V}$ **do** // Loop over all vertices in $\mathbf{G}_D$
5 **if** $(|v.in()| = 0) \wedge (|v.out()| = 0)$ **then**
6 $\mathbf{C}.add(\{v\})$ // Add singletons to the final list of clusters
7 **else**
8 $\mathbf{Q}.add((v.t, \{v\}))$ // Add vertex with its time-stamp to queue **Q**
9 **Sort**$(\mathbf{Q})$ // Sort queue according to time-stamps (earliest first)
10 **while** $\mathbf{Q} \neq [\,]$ **do** // Loop over temporal clusters until **Q** is empty
11 $(t, \mathbf{c}) = \mathbf{Q}.pop()$ // Get first cluster tuple in **Q**
12 $\mathbf{o} = \cup v_i.out(), v_i \in \mathbf{c}$ // Set of all outgoing vertices
13 **if** $\mathbf{o} = \emptyset$ **then**
14 $\mathbf{C}.add(\mathbf{c})$ // Add cluster **c** with no outgoing edges to the final list **C**
15 **else**
16 **if** $m_n =$ ***Next*** **then** // Get vertex with smallest time-stamp
17 $v_n = v_j \in \mathbf{o} : \text{argmin}\{v_j.t : v_i \in \mathbf{c}, v_j \in \mathbf{o}, s_{i,j} \geq \delta_s\}$
18 **else if** $m_n =$ ***Max-sim*** **then** // Get vertex with highest similarity
19 $v_n = v_j \in \mathbf{o} : \text{argmax}\{s_{i,j} : v_i \in \mathbf{c}, v_j \in \mathbf{o}, s_{i,j} \geq \delta_s\}$
20 **else if** $m_n =$ ***Avr-sim*** **then** // Get vertex with highest average similarity
21 $v_n = v_j \in \mathbf{o} : \text{argmax}\{\bar{s} = \sum s_{i,j} / |\{(v_i, v_j) : v_i \in \mathbf{c}, v_j \in \mathbf{o}\}|, \bar{s} \geq \delta_s\}$
22 $p_{\Delta t} = \mathbf{CheckTempConstr}(v_n.t, \mathbf{c}, \mathbf{T})$ // Temporal plausibility
23 **if** $p_{\Delta t} \geq \delta_p$ **then**
24 $\mathbf{Q}.add((v_n.t, \mathbf{c} \cup \{v_n\}))$ // Add expanded **c** to **Q**
25 **Sort**$(\mathbf{Q})$ // Sort queue according to time-stamps (earliest first)
26 **else**
27 $\mathbf{C}.add(\mathbf{c})$ // Add **c** to the list of final clusters
28 **return C**

constraints are ignored). We also input a minimum plausibility threshold δ_p which is used to consider which record pairs are to be added into clusters based on their temporal constraints, a minimum similarity threshold δ_s to decide whether a record (vertex) can be added to a cluster, and the selection method m_n (one of the three methods described above) which determines which vertices to add into a cluster.

We first (in line 1) convert the undirected similarity graph **G** into a directed graph where each vertex (birth record) has an outgoing edge to any future vertex, as shown in Figure 4.2. The function **GenerateTempDirGraph**() generates such a directed

graph $\mathbf{G}_D$ by considering the time differences between the pairs of vertices in $\mathbf{G}$, such that $\forall(v_i, v_j) \in \mathbf{G}_D.\mathbf{E} : v_j.t \geq v_i.t$. Furthermore, if the temporal constraints are given (i.e. $\mathbf{T}$ is non-empty), then the pairwise similarities $s_{i,j}$ are weighted by the temporal plausibility values calculated as defined in Equation 4.1. In line 4, the algorithm then loops over each vertex $v \in \mathbf{G}_D$ and adds v to the final list of clusters $\mathbf{C}$ if v does not have any incoming or outgoing edges to other vertices (lines 5 and 6), i.e. the vertex is a singleton. Otherwise, a new cluster is created containing only vertex v, and this cluster is added together with its time-stamp, $v.t$, as a tuple into the priority queue $\mathbf{Q}$ for further processing (line 8). In line 9 we sort $\mathbf{Q}$ 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 $\mathbf{c}$ with the earliest time-stamp t (line 11). We then find for each vertex $v_c \in \mathbf{c}$ all its outgoing vertices in $\mathbf{G}_D$, and in line 12 we combine these into the set $\mathbf{o}$ of all outgoing vertices for $\mathbf{c}$. If $\mathbf{o}$ is empty for the current cluster $\mathbf{c}$ then $\mathbf{c}$ is added to the final list of clusters $\mathbf{C}$ in line 14 because it cannot be expanded further.

On the other hand, if there are outgoing vertices (the set $\mathbf{o}$ is not empty), then based on the selection method m_n, as described above, the algorithm selects the next best vertex, v_n, to be added into the current cluster $\mathbf{c}$ in lines 16 to 21. For the *Next* and *Max-sim* selection methods, the chosen vertex v_n has to have a minimum pairwise similarity of δ_s with at least one vertex in the cluster $v_i \in \mathbf{c}$, whereas for the *Avr-sim* method, the average similarity $\bar{s}$ of v_n with all other records in cluster $\mathbf{c}$ has to be at least δ_s.

Using the function **CheckTempConstr()** in line 22 we then check the temporal plausibility $p_{\Delta t}$ between vertex v_n and all vertices in $\mathbf{c}$ based on the list of temporal constraints $\mathbf{T}$ (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 δ_p (i.e. v_n is temporary plausible with all other vertices in $\mathbf{c}$), then v_n is added to the current cluster $\mathbf{c}$ and the expanded cluster is added as a new tuple into $\mathbf{Q}$ with $v_n.t$ as the tuple's time-stamp (line 24). $\mathbf{Q}$ is sorted again in line 25 to ensure the cluster with the smallest time-stamp is selected in the next iteration (line 25). If v_n is not temporally plausible with at least one vertex in $\mathbf{c}$, then $\mathbf{c}$ is added to the final list of clusters $\mathbf{C}$ in line 27 because it cannot be expanded further.

Complexity analysis: We now conduct a complexity analysis of our proposed greedy clustering method. The initial for loop (lines 4 to 8) in Algorithm 1 iterates through every vertex in the directed similarity graph $v \in \mathbf{G}_D.\mathbf{V}$. Therefore, considering $\mathbf{G}_D = (\mathbf{V}, \mathbf{E})$, the time complexity of the initial for loop is $O(|\mathbf{V}|)$. The priority queue sorting (line 9) has a complexity of $O(|\mathbf{V}| \cdot log(|\mathbf{V}|))$. The while loop (lines 10 to 27) has a maximum time complexity of $O(|\mathbf{E}|)$, since for each record (vertex) in the graph $\mathbf{G}$, all edges connected to that vertex would be searched to find the next best vertex, thus resulting in a linear search across all edges. Therefore the total time complexity of the greedy clustering algorithm is $O(|\mathbf{V}| + |\mathbf{V}| \cdot log(|\mathbf{V}|) + |\mathbf{E}|)$.

Algorithm 2: Star clustering

Input: **G** - Undirected pairwise similarity graph
T - List of temporal constraints (as discussed in Section 4.2)
δ_p - Minimum plausibility for record pairs to be added to a star cluster
δ_s - Minimum similarity for record pairs to be added to a star cluster
m_s - Method to sort vertices for processing
m_r - Method to resolve overlapping clusters
Output: **C** - Final list of clusters

```
1  C = [ ]                                          // Initialise an empty list of clusters
2  U = [ ]                          // Initialise an empty list to hold unassigned vertices
3  for v_i ∈ G.V do                                 // Loop over all vertices in graph
4      n_i = GetSimNeighbours(G, v_i, δ_s)          // Similar neighbours of v_i
5      d_i = |n_i|                                  // Degree of v_i
6      a_i = CalcAvrSimNeighbours(G, v_i, n_i)      // Calculate average similarity
7      U.add((v_i, d_i, n_i, a_i))                  // Add tuple to list of unassigned vertices
8  SortTuples(U, m_s)                               // Sort according to sorting method
9  for (v_i, d_i, n_i, a_i) ∈ U do
10     U.removeTuple(v_i)                           // Remove assigned vertex from unassigned list
11     c_i = {v_i}                                  // Initialise a new cluster with selected vertex as centre
12     while n_i ≠ ∅ do
13         v_j = GetNextBestNeighbour(c_i, n_i)     // Select next best neighbour
14         n_i.remove(v_j)                          // Remove selected next best neighbour
15         if IsTempPossSimNeighbour(v_j, c_i, T, δ_p) then
16             c_i ∪ {v_j}                          // Add temporally plausible vertex to cluster
17             U.removeTuple(v_j)                   // Remove vertex added to the cluster
18     C.add(c_i)                                   // Add cluster to the final cluster list
19 v_rep = GetRepeatVertices(C)                     // Get vertices that occur in multiple clusters
20 C = ResolveOverlap(C, v_rep, m_r, δ_s)           // Assign vertices to best cluster
21 return C
```

4.3.2 Star Clustering

In the star clustering approach, records with the highest degree (maximum number of neighbours) and similarity to neighbouring vertices are selected as cluster centers, and their neighbouring vertices are added to the corresponding cluster. This algorithm was shown to be one of the best performers in a previous evaluation study of clustering algorithms for RL [153]. Our contribution to improve star clustering is two-fold: (a) we introduce temporal constraints as discussed in Section 4.2, and (b) we develop several methods for cluster centre selection and resolve cluster overlaps.

Algorithm 2 outlines our modified star clustering algorithm, which is able to either consider temporal constraints (if the list of constraints **T** is provided) or ignore them (if **T** is empty) in cluster generation. The input to the algorithm includes the pairwise similarity graph, **G**, which is generated as specified in Definition 2, and the list **T** of temporal constraints. We also require the minimum plausibility δ_p and minimum similarity δ_s thresholds to decide if a vertex is to be added to a cluster, and the sorting and overlap resolving methods, m_s and m_r.

The algorithm starts by initialising an empty list of clusters, **C**, and an empty list **U** which will hold information about the vertices that are not yet assigned to clusters. Initially, all vertices in the similarity graph **G** are marked as unassigned by adding them to the list **U** in the loop starting in line 3. For each vertex $v_i \in \mathbf{G.V}$, using the function **GetSimNeighbours**() in line 4 we get the set of its neighbours $\mathbf{n}_i \subseteq \mathbf{G.V}$ that have an edge similarity of at least δ_s. We count the number of these neighbours as the degree d_i of vertex v_i in line 5, and also calculate the average similarity of all edges between v_i and its similar neighbours in $\mathbf{n}_i$ in line 6. Next, in line 7 we append a tuple containing v_i, d_i, $\mathbf{n}_i$, and a_i to the list of unassigned vertices **U**.

Once tuples for all vertices in **G** have been added into **U**, we sort **U** such that the best vertex to select as a cluster centre is at the beginning of this list. We investigate three different methods of how to order vertices based on the sorting method provided in m_s:

- **Avr-sim-first**: We order the tuples in descending order based on their average similarities a_i first and then based on the degree d_i. With this ordering we will process vertices that have high similarities to other vertices first.

- **Degree-first**: We order the tuples in descending order based on their degree d_i first and then based on their average similarity a_i. With this ordering we will process vertices that have many high similarity edges to other vertices first.

- **Comb**: With this method we order vertices in descending order based on a combined score where we multiply their average similarity with the logarithm of their degree, i.e. $a_i \times log(d_i)$. We take the logarithm of d_i because a_i is normalised into $0 \leq a_i \leq 1$ while d_i is a positive integer value and therefore would dominate the combined score. With this method we aim to weigh both degree and average similarities to obtain an improved ordering.

In lines 9 to 18 of the algorithm, we process one tuple in **U** after another. Only an unassigned vertex can become the centre of a new star cluster. The tuple of vertex $v_i \in \mathbf{U}$ selected to become a star centre is removed from the list of unassigned vertices and a new cluster $\mathbf{c}_i$ is created in line 11. Then we find the next best vertex to add to cluster $\mathbf{c}_i$, using the function **GetNextBestNeighbour**(). This function selects the vertex $v_j \in \mathbf{n}_i$ which has the highest average similarity with the vertices that are currently assigned to the cluster $\mathbf{c}_i$. The selected vertex v_j is removed from $\mathbf{n}_i$ in line 14 so it cannot be selected as the best neighbour in the next iteration. For each next best neighbour v_j we check in line 15 if v_j is plausible with every other vertex in $\mathbf{c}_i$ with regard to the temporal constraints given in the list **T** using the function **IsTempPossSimNeighour**() (note that if **T** is empty then this function always returns true), and the minimum plausibility threshold δ_p. We add each plausible vertex v_j to the cluster $\mathbf{c}_i$ in line 16 and remove its corresponding tuple from **U** in line 17. This means that these vertices cannot become the centre of another star cluster. In line 18, each processed cluster $\mathbf{c}_i$ is added to the final cluster list **C**.

The final steps of Algorithm 2 (lines 19 and 20), deal with those vertices that are members of more than one cluster (note these are not star cluster centres). Overlapping clusters are not allowed in RL because each cluster represents one entity. In line 19 we therefore identify the set $\mathbf{v}_{rep}$ of vertices which occur in more than one cluster in the list $\mathbf{C}$, and in line 20 we use the function **ResolveOverlap**() to resolve overlapping clusters, where the method m_r determines how we assign a vertex $v_j \in \mathbf{v}_{rep}$ to its best cluster. We investigate three methods to resolve overlaps:

- **Avr-all**: We average the similarities between the vertex v_j and all the vertices in a cluster it is connected to in the similarity graph $\mathbf{G}$ by dividing this similarity sum by $n-1$ where n is the number of vertices in the cluster (including v_j), i.e. we do take vertices in a cluster which are not connected to v_j in $\mathbf{G}$ into account.
- **Avr-high**: We calculate the average similarity between the vertex v_j and all the vertices in a cluster it is connected to in the similarity graph $\mathbf{G}$, with similarities of at least δ_s.
- **Edge-ratio**: In this method we count the number of edges between v_j and vertices in a cluster that have a similarity of at least δ_s and divide this number by $n-1$ where n is the number of vertices in the cluster (including v_j).

For each vertex $v_j \in \mathbf{v}_{rep}$, 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 vertex 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 edges with a similarity of at least δ_s. At the end of this process, the final list of clusters $\mathbf{C}$ contains no overlapping clusters.

Complexity analysis: We now conduct a complexity analysis of the star clustering algorithm. The initial for loop (lines 3 to 7) iterates over every vertex in the pairwise similarity graph $v \in \mathbf{G}.\mathbf{V}$ and calculates the similarity to its connected (neighbouring) vertices. This is equivalent to searching across all edges in the graph, and therefore, the initial for loop has a total time complexity of $O(|\mathbf{E}|)$ where $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. The list sorting (line 8) has a complexity of $O(|\mathbf{V}| \cdot log(|\mathbf{V}|))$. The next for loop (lines 9 to 18) also runs for $|\mathbf{V}|$ iterations at most (if no record pairs are connected), and in the worst case, if all record pairs are connected in the similarity graph $\mathbf{G}$, the for loop would execute once, and the internal while loop (lines 12 to 17) would execute $|\mathbf{V}|-1$ number of times (all records would be added to one cluster). Therefore, the time complexity of this for loop is $O(|\mathbf{V}|)$. The time complexity of resolving cluster overlaps (lines 19 and 20) is $O(|\mathbf{V}|)$ in worst case, and therefore, the total time complexity of the star clustering algorithm is $O(|\mathbf{V}| + |\mathbf{V}| \cdot log(|\mathbf{V}|) + |\mathbf{E}|)$.

4.3.3 Robust Graph Clustering

Our robust graph clustering technique is inspired by the CLIP algorithm proposed by Saeedi et al. [154] for conducting multi-source RL (which we discussed in Section 3.1). We use their concept of *link strength* (categorising record pairs as *strong*,

normal, and *weak*, as we describe below), and furthermore incorporate temporal constraints in robust graph clustering. This clustering approach comprises of two steps, where in the first step we generate a set of *base clusters* which need to be temporally consistent if temporal characteristics are being considered. Base clusters are expected to have high precision but may have low recall (see Section 2.4 for definitions) since we only use links with higher strength (more confident links) in the base cluster generation. In the second step, these base clusters are iteratively merged in a greedy manner (while maintaining temporal consistency if temporal constraints are being considered) to improve the recall if the linkage result.

4.3.3.1 Generating Base Clusters

In the first step in our proposed robust graph clustering approach, we generate a set of base clusters using the concept of link strength proposed by Saeedi et al. [154]. These base clusters are essentially connected components (i.e. sub-graphs in a graph where any two vertices are connected via a path and not connected to any other vertex in the parent graph [21]) which we generate using the pairwise similarity graph **G**, where every pair of records in a cluster must be temporally consistent if temporal constraints **T** are given. The original connected component based CLIP clustering approach by Saeedi et al. [154] differs from ours in that it does not consider temporal constraints, and also it assumes the linkage of records across multiple data sources only. The requirement of incorporating temporal constraints makes the problem more complex, since simply obtaining the connected components does not ensure temporal consistency between all records within a component, as the example in Figure 4.3 shows.

We initially filter graph edges **G**.**E** based on their pairwise similarity $s_{i,j}$ (see Definition 1) using a minimum similarity threshold δ_s ($s_{i,j} \geq \delta_s$). Next, extending the ideas described by Saeedi et al. [154], we categorise the filtered edges into three types as follows:

- **Strong**: An edge (r_i, r_j) is *strong* if the corresponding pairwise similarity $s_{i,j}$ is the highest similarity for both records r_i and r_j with regard to any other edges they have with other records in **G**.
- **Norm**: An edge (r_i, r_j) is *normal* if the corresponding similarity $s_{i,j}$ is the highest similarity for either record r_i or r_j (but not both) with regard to any other edges they have with other records in **G**.
- **WeakHigh**: An edge (r_i, r_j) is *weak high* if it is neither strong nor normal.

As detailed in Algorithm 3, one or several of these edge types are used to create the initial connected components (named *base clusters*). Edges (r_i, r_j) with similarity $s_{i,j} < \delta_s$ are ignored. The temporally implausible base clusters are then split further until all are temporally consistent. Similar to the previous clustering algorithms, the main inputs to the Algorithm 3 are an undirected pairwise similarity graph **G** and

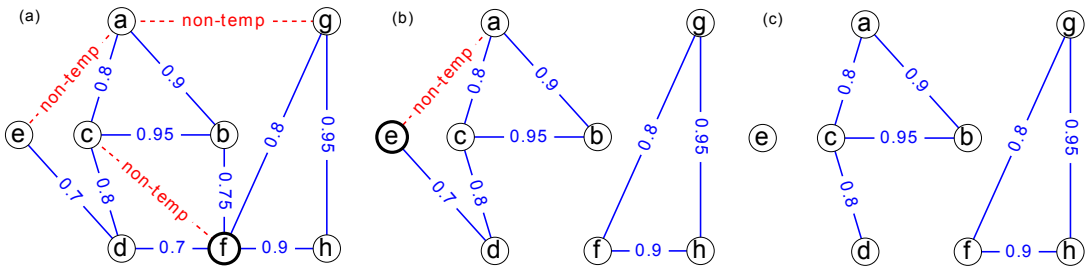

Figure 4.3: Example iterative temporal cluster refinement in the base cluster generation phase, as detailed in Algorithm 3, where in each step we identify the best edge(s) to be removed that most improves the temporal consistency of the cluster(s).

Algorithm 3: Robust graph clustering - Base cluster generation

Input: $\mathbf{G}$ - Undirected pairwise similarity graph
$\mathbf{T}$ - List of temporal constraints (as discussed in Section 4.2)
δ_p - Minimum plausibility for record pairs to be added to a cluster
δ_s - Minimum similarity for record pairs to be added to a cluster
e_b - Type(s) of edges to use to create base clusters
Output: $\mathbf{C}_b$ - Set of generated base clusters

$\mathbf{C}_b = \{\}$ // Initialise an empty set of clusters
$\mathbf{E}_b = \textbf{FindTempEdges}(\mathbf{G}, e_b, \mathbf{T}, \delta_p, \delta_s)$ // Get temporal edges of type e_b
$\mathbf{C}_{cc} = \textbf{GetConnComp}(\mathbf{G}, \mathbf{E}_b)$ // Get the set of connected components
for $\mathbf{c}_i \in \mathbf{C}_{cc}$ **do** // Iterate through the connected components
  if $\textbf{IsTempPlausibleCluster}(\mathbf{c}_i, \mathbf{T}, \delta_p)$ **then**
    $\mathbf{C}_b = \mathbf{C}_b \cup \{\mathbf{c}_i\}$ // Add the cluster to the final cluster set
    $\mathbf{C}_{cc} = \mathbf{C}_{cc} \setminus \{\mathbf{c}_i\}$ // Remove the processed cluster
while $\mathbf{C}_{cc} \neq \emptyset$ **do** // Iterate through the temporally inconsistent clusters
  $\mathbf{c}_j = \mathbf{C}_{cc}.pop()$ // Get the first connected component
  $\mathbf{I} = [\,]$ // Initialise a list to hold cluster vertex information
  for $v_i \in \mathbf{c}_j$ **do** // Iterate through the vertices in cluster $\mathbf{c}_j$
    $\mathbf{v}_n = \textbf{GetNonTemp}(\mathbf{c}_j, v_i, \mathbf{T}, \delta_p)$ // Get temporally implausible vertices
    $\mathbf{I}.add((\textbf{CalcSim}(v_i, \mathbf{c}_j, \mathbf{G}), \textbf{GetNeigh}(v_i, \mathbf{c}_j), \mathbf{v}_n, v_i))$
  $\mathbf{n}_{ref}, \mathbf{nnt}_{ref}, v_{ref} = \textbf{GetVertexToRefineCluster}(\mathbf{I})$ // Get vertex to refine $\mathbf{c}_j$
  $\mathbf{C}_r = \textbf{GetTempImproved}(\mathbf{c}_j, v_{ref}, \mathbf{n}_{ref}, \mathbf{nnt}_{ref})$ // Partition $\mathbf{c}_j$ based on v_{ref}
  for $\mathbf{c}_i \in \mathbf{C}_r$ **do**
    if $\textbf{IsTempPlausibleCluster}(\mathbf{c}_i, \mathbf{T}, \delta_p)$ **then**
      $\mathbf{C}_b = \mathbf{C}_b \cup \{\mathbf{c}_i\}$ // Add cluster to the final base cluster set
    else // If not temporal, add cluster to $\mathbf{C}_{cc}$ to be refined
      $\mathbf{C}_{cc} = \mathbf{C}_{cc} \cup \{\mathbf{c}_i\}$
return $\mathbf{C}_b$

a list of temporal constraints $\mathbf{T}$ (which is empty if temporal constraints are being ignored). Furthermore, we provide the minimum plausibility δ_p and minimum sim-

ilarity δ_s thresholds to decide if a vertex is to be added to a cluster, and the type(s) of edges e_b (one or several of *Strong*, *Norm* and *WeakHigh* link strengths, as described above) to be considered for base cluster generation.

First, in lines 1 to 7 of Algorithm 3, we generate the connected components $\mathbf{C}_{cc}$ based on the edges (record pairs) in $\mathbf{G}$ of the selected edge type(s) e_b which we retrieve in the set $\mathbf{E}_b$ in line 2. Only the temporally plausible edges in $\mathbf{G}$ are considered in this step if the list of temporal constraints $\mathbf{T}$ is not empty. We then check, in line 5, if all pairs of records in a connected component $\mathbf{c}_i \in \mathbf{C}_{cc}$ are temporally plausible. This check is necessary since in the previous step we did not assess the temporal consistency of record pairs not contained in the similarity graph $\mathbf{G}$. If they are temporally consistent, then $\mathbf{c}_i$ is added to the set of base clusters, $\mathbf{C}_b$, and removed from the set of connected components $\mathbf{C}_{cc}$. At the end of this step the clusters left in $\mathbf{C}_{cc}$ are those that contain record pairs that are temporally implausible (like two birth records five months apart).

We next process the clusters in $\mathbf{C}_{cc}$ (lines 8 to 20) one by one. We pick one $\mathbf{c}_j \in \mathbf{C}_{cc}$ (line 9) and generate a list $\mathbf{I}$ which for each vertex $v_i \in \mathbf{c}_j$ contains its average similarity with the other vertices in $\mathbf{c}_j$, its neighbours in $\mathbf{c}_j$, and the other vertices in $\mathbf{c}_j$ it is temporally not plausible with ($\mathbf{v}_n$). In line 14, using the function **GetVertexToRefineCluster**() we identify from $\mathbf{I}$ the best vertex $v_{ref} \in \mathbf{c}_j$ to process which reduces by most the number of temporal implausible edges in $\mathbf{c}_j$.

To select the best vertex v_{ref}, in line 14 we first attempt to find the first vertex in $\mathbf{I}$ with a non-empty intersection between its set of neighbours $\mathbf{n}_{ref}$ and the set of neighbours of vertices which v_{ref} is temporally inconsistent with, $\mathbf{nnt}_{ref}$. If the intersection is empty for all vertices in $\mathbf{I}$, v_{ref} will be the vertex with the lowest average similarity in the cluster, the lowest number of neighbours, and that is involved in the highest number of implausible edges in $\mathbf{c}_j$. In the example shown in Figure 4.3, assuming the vertices with non-temporal connections are ordered as $\mathbf{I} = [f, e, a, g, c]$, we select vertex f first since it is the first vertex in $\mathbf{I}$ with a non-empty intersection ($\mathbf{n}_{ref} \cap \mathbf{nnt}_{ref} = \{b, d\}$). Subsequently, in Figure 4.3 (b), we check vertices e and a in that order, for non-empty intersection. However, since the intersection is empty for both vertices a and e, vertex e which has a lower average similarity in the cluster is selected for removal.

Based on the selected vertex v_{ref}, and sets $\mathbf{n}_{ref}$ and $\mathbf{nnt}_{ref}$, we then partition the cluster $\mathbf{c}_j$ (line 15) using the function **GetTempImproved**() which returns the set $\mathbf{C}_r$ of two or more temporally improved clusters. In lines 16 to 20 we check each cluster $\mathbf{c}_i \in \mathbf{C}_r$ if it is temporally consistent (in which case we add it to the set of base clusters, $\mathbf{C}_b$) or not (in which case we add it to the set of clusters $\mathbf{C}_{cc}$ to be processed further). In Figure 4.3 (a), the edges that vertex f has with its neighbours $\{b, d\}$ are removed first, and then edges of vertex e are removed next resulting in the three temporally consistent clusters shown in Figure 4.3 (c). The algorithm ends once all clusters in $\mathbf{C}_{cc}$ have been processed and the set of temporally consistent base clusters, $\mathbf{C}_b$, that is to be refined in the next phase of our approach, is returned in line 21 of Algorithm 3.

Complexity analysis: We first conduct a complexity analysis of the base cluster generation phase of robust graph clustering, in terms of the vertex and edge counts in the pairwise similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. The complexity of filtering temporal edges of a given type (line 2) is $O(|\mathbf{E}|)$, whereas the time complexity for connected component identification is $O(|\mathbf{V}| + |\mathbf{E}|)$. The for loop from line 4 to 7 iterates $|\mathbf{V}|$ times at most, whereas in the implementation the temporally plausible vertices for each vertex can be stored in a dictionary ($O(1)$ complexity for line 5), thus resulting in a time complexity of $O(|\mathbf{V}|)$ for the entire for loop. For the while loop running from line 8 to 20, let us consider the worst case scenario of having one large connected component containing all vertices from the similarity graph $\mathbf{G}$. For one connected component the while loop would execute only once, but the internal for loop (lines 11 to 13) would have a time complexity of $O(|\mathbf{V}|^2)$ given that the similarity calculation step (line 13) would consider every record pair. Given that this is the most time consuming phase of the base cluster generation algorithm, the total time complexity of based cluster generation is $O(|\mathbf{V}|^2)$. Note that in practice, the base cluster generation algorithm takes much less time as opposed to comparing all record pairs (which also has quadratic time complexity) since in line 13 of Algorithm 3, we consider pre-calculated similarity values and due to the application of blocking.

4.3.3.2 Iterative Cluster Merging

In this section we describe the second step in our proposed robust graph clustering approach, where we merge base clusters that have high overall similarities between all their individual records. This process is iterative, in that merged clusters will be further compared until no cluster is highly similar with any other cluster. We ensure that all merged clusters are temporally consistent.

The main inputs to Algorithm 4 are an undirected pairwise similarity graph $\mathbf{G}$, the list of base clusters resulting from Algorithm 3, and a list of temporal constraints $\mathbf{T}$ (which is empty if temporal constraints are being ignored). We also provide the minimum plausibility δ_p and minimum similarity δ_s thresholds to decide if clusters can be merged, and the type(s) of edges e_m (one or several of *Norm* and *WeakHigh* link strengths, as described above) to be considered for base cluster merging. Furthermore, the pairwise cluster similarity calculation method m_m and the weight w to assign to cluster similarity, which we describe later, are provided as input.

As detailed in Algorithm 4, we use a priority queue and sets of similar clusters to keep track of cluster pairs that are similar in order to prevent a full pairwise recalculation of cluster similarities each time a merged cluster is generated. We start the algorithm (lines 1 and 2) by initialising the empty list of final clusters to be generated, $\mathbf{C}$, and the empty priority queue, $\mathbf{Q}$, which will hold cluster pairs and their similarities.

In lines 3 to 5, we calculate the similarities between every pair of base clusters in $\mathbf{C}_b$, where we consider a set of edge types, e_m, different to Algorithm 3, with one or several of *Norm*, and *WeakHigh*, as described before. Compared to the edge types used for base cluster generation in Algorithm 3, the edge types used for cluster merg-

Algorithm 4: Robust graph clustering - Similar base cluster merging

Input: $\mathbf{G}$ - Undirected pairwise similarity graph
$\mathbf{C}_b$ - Base clusters (as generated in Algorithm 3)
$\mathbf{T}$ - List of temporal constraints (as discussed in Section 4.2)
δ_p - Minimum temporal plausibility for clusters to be merged
δ_s - Minimum similarity for clusters to be merged
e_m - Type(s) of edges to use to merge base clusters
m_m - Method to merge base clusters (cluster similarity calculation method)
w - Weight to assign to cluster similarity ($1 - w$ weight for cluster coverage)
Output: $\mathbf{C}$ - Final list of clusters

1 $\mathbf{C} = [\,]$ // Initialise an empty list of clusters
2 $\mathbf{Q} = [\,]$ // Initialise an empty priority queue
3 $\mathbf{E}_m = \textbf{FindTempEdges}(\mathbf{G}, e_m, \mathbf{T}, \delta_p, \delta_s)$ // Get temporal edges of type e_m
4 **for** $(\mathbf{c}_i, \mathbf{c}_j) \in \mathbf{C}_b, i < j$ **do** // Calculate pairwise cluster similarities
5 $\quad \mathbf{Q}.add((\textbf{CalcSim}(\mathbf{c}_i, \mathbf{c}_j, \mathbf{G}, \mathbf{E}_m, m_m, w), \mathbf{c}_i, \mathbf{c}_j))$
6 **while** $\mathbf{Q} \neq \emptyset$ **do** // Iterate through cluster pairs in $\mathbf{Q}$
7 $\quad s_{x,y}, \mathbf{c}_x, \mathbf{c}_y = \mathbf{Q}.pop()$ // Get the most similar cluster pair from $\mathbf{Q}$
8 $\quad \mathbf{C}_x = \{\mathbf{c}_p : (\mathbf{c}_p, \mathbf{c}_x) \in \mathbf{Q} \ \wedge \ s_{p,x} \geq \delta_s, \ \mathbf{c}_p \neq \mathbf{c}_y\}$ // Clusters similar with $\mathbf{c}_x$
9 $\quad \mathbf{C}_y = \{\mathbf{c}_q : (\mathbf{c}_q, \mathbf{c}_y) \in \mathbf{Q} \ \wedge \ s_{q,y} \geq \delta_s, \ \mathbf{c}_q \neq \mathbf{c}_x\}$ // Clusters similar with $\mathbf{c}_y$
10 $\quad \textbf{RemoveAllTuplesWithCluster}(\mathbf{Q}, \mathbf{c}_x)$ // Remove tuples with $\mathbf{c}_x$ from $\mathbf{Q}$
11 $\quad \textbf{RemoveAllTuplesWithCluster}(\mathbf{Q}, \mathbf{c}_y)$ // Remove tuples with $\mathbf{c}_y$ from $\mathbf{Q}$
12 $\quad$ **if** $(s_{x,y} \geq \delta_s)$ ***and*** $\textbf{IsTempPlausibleClusterPair}(\mathbf{c}_x, \mathbf{c}_y, \mathbf{T}, \delta_p)$ **then**
13 $\quad\quad \mathbf{c}_{x+y} = \mathbf{c}_x \cup \mathbf{c}_y$ // Merge highly similar, temporally plausible clusters
14 $\quad\quad$ **if** $\mathbf{C}_x \cup \mathbf{C}_y = \emptyset$ **then** // If no other clusters are similar with $\mathbf{c}_x$ or $\mathbf{c}_y$
15 $\quad\quad\quad \mathbf{C}.add(\mathbf{c}_{x+y})$ // Add the merged cluster to final cluster list
16 $\quad\quad$ **else** // Add $\mathbf{c}_{x+y}$ with clusters similar to $\mathbf{c}_x$ or $\mathbf{c}_y$ into $\mathbf{Q}$
17 $\quad\quad\quad$ **for** $\mathbf{c}_z \in \mathbf{C}_x \cup \mathbf{C}_y$ **do**
18 $\quad\quad\quad\quad \mathbf{Q}.add((\textbf{CalcSim}(\mathbf{c}_z, \mathbf{c}_{x+y}, \mathbf{G}, \mathbf{E}_m, m_m, w), \mathbf{c}_z, \mathbf{c}_{x+y}))$
19 $\quad$ **else** // Add clusters $\mathbf{c}_x$ and $\mathbf{c}_y$ to the final cluster list if non-mergeable
20 $\quad\quad \mathbf{C}.add(\mathbf{c}_x), \mathbf{C}.add(\mathbf{c}_y)$
21 **return** $\mathbf{C}$

ing in Algorithm 4 should have a lower strength, since base clusters are expected have high precision as we highlighted earlier. Therefore, *Strong* edges, which are the edges with the highest strength, are not considered for cluster expansion. Furthermore, if the list of temporal constraints $\mathbf{T}$ is not empty, only temporally plausible record pairs (edges) of type(s) e_m are considered.

In line 5 we calculate the similarity between a cluster pair $\mathbf{c}_i$ and $\mathbf{c}_j$ using the edges $\mathbf{E}_m$ of type e_m, merge method, m_m, and a cluster similarity weight, w. The merge method m_m determines how the overall similarity between clusters is calculated, where it can be one of the aggregation functions minimum, average, or maximum. The aggregated similarity and the coverage between two clusters are assigned weights w and $1 - w$ respectively. The coverage is the ratio between the number of edges across $\mathbf{c}_i$ and $\mathbf{c}_j$ in $\mathbf{E}_m$, and the number of edges across $\mathbf{c}_i$ and $\mathbf{c}_j$ in $\mathbf{G}$, which

reflects the proportion of edges covered in our similarity calculation. The cluster similarity, $s_{x,y}$, returned by **CalcSim**() is the weighted sum of similarity and coverage.

The main loop starts in line 6 and iterates over each cluster pair tuple in the queue **Q**. For both clusters in the tuple, $\mathbf{c}_x$ and $\mathbf{c}_y$, we next (lines 8 and 9) identify all other clusters that they are similar with, and we keep these clusters in two sets $\mathbf{C}_x$ and $\mathbf{C}_y$, respectively. We then remove all tuples in **Q** that contain $\mathbf{c}_x$ or $\mathbf{c}_y$ since they should not be re-processed. In line 12 we check if the similarity between $\mathbf{c}_x$ and $\mathbf{c}_y$ is at least the minimum cluster merge similarity δ_s and if they are temporally consistent with each other. If this is the case we merge clusters $\mathbf{c}_x$ and $\mathbf{c}_y$ into $\mathbf{c}_{x+y}$ in line 13.

If both $\mathbf{c}_x$ and $\mathbf{c}_y$ are not similar with any other clusters (i.e. both $\mathbf{C}_x$ and $\mathbf{C}_y$ are empty), then based on the triangular inequality [35] we know that the merged cluster $\mathbf{c}_{x+y}$ cannot be merged further with any other clusters. Therefore $\mathbf{c}_{x+y}$ is added to the list of final clusters, **C**, in line 15. Otherwise, in line 17 we calculate the similarity of the merged cluster, $\mathbf{c}_{x+y}$, with each cluster in $\mathbf{C}_x$ and $\mathbf{C}_y$ and add new tuples into the queue **Q** in line 18.

If a cluster pair in the queue **Q** was not similar enough or not temporally consistent, we do not merge $\mathbf{c}_x$ and $\mathbf{c}_y$ but instead add both into **C** in line 20. Finally we return the set of merged and temporally consistent clusters, **C**.

Complexity analysis: We now conduct a complexity analysis of the iterative cluster merging phase of robust graph clustering. Similar to the base cluster generation phase, the complexity of filtering edges of a given type (line 3) is $O(|\mathbf{E}|)$, whereas the initial for loop (lines 4 and 5) is quadratic by the number of base clusters, and therefore has a time complexity of $O(|\mathbf{C}_b|^2)$. The main while loop (lines 6 to 20) iterates at most for $|\mathbf{C}_b|^2$ number of times, since even though we add new cluster pairs to the queue **Q** in line 18, we delete a larger or equal number of entries from the queue in lines 10 and 11 in such occurrences. All functions within the while loop can be executed with $O(1)$ complexity, except the for loop in lines 17 and 18 which has $O(|\mathbf{C}_b|)$ time complexity at worst (when a given merged cluster pair is compared with every other base cluster). Therefore, the while loop has a total time complexity of $O(|\mathbf{C}_b|^3)$, which is the overall time complexity of Algorithm 4 given that it is the dominant term. Hence, considering both the base cluster generation phase and cluster merging phase, the overall time complexity of the robust graph clustering technique is $O(|\mathbf{V}|^2 + |\mathbf{C}_b|^3)$.

4.4 Experimental Evaluation

In this section, we present the results obtained with an empirical evaluation of our three graph clustering methods using data characteristics. We conducted experiments on the real-world Isle of Skye (IoS) and synthetic UK birth data sets which we described in Section 2.5. Furthermore, we implemented our algorithms in Python 2.7, and all experiments were conducted on a server running Ubuntu 18.04 with 64-bit Intel Xeon 2.10 GHz CPUs and 512 GB of memory.

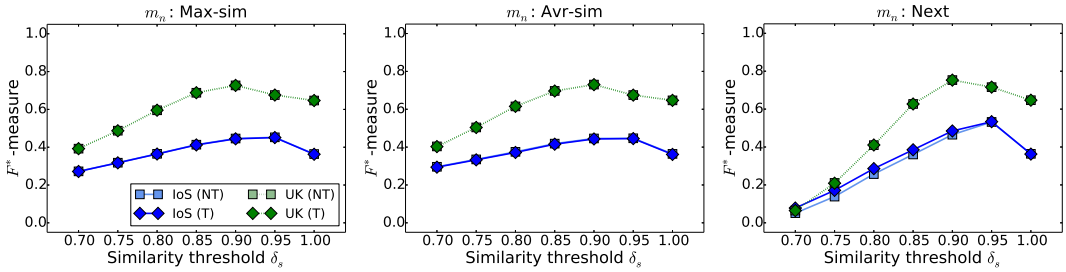

Figure 4.4: Greedy clustering results obtained with (T) and without (NT) temporal constraints: Average of F^* values obtained with different similarity graphs, shown for different similarity thresholds δ_s, and different next vertex selection methods m_n.

As discussed in Section 2.6, we generated three types of pairwise similarity graphs $\mathbf{G}$ for each birth data set by comparing *All* attributes ($\mathbf{G}_A$), *Parent names and address* attributes ($\mathbf{G}_{NA}$), and *Parent names only* ($\mathbf{G}_N$). For each algorithm, we explored the similarity threshold values δ_s ranging from 0.7 to 1.0 in steps of 0.05, whereas we set the plausibility threshold value δ_p to 0.5 subsequent to conducting an initial set of experiments on temporal possibilities of record pairs in birth data sets. As the baseline techniques, we used the proposed clustering techniques disregarding the data characteristics. Since we linked the IoS and UK birth data sets with the aim of identifying sibling groups, we applied temporal constraints based on temporal data characteristics (such as a sibling link with a five month age gap being biologically implausible) in our experiments.

For evaluation we used the F^*-measure presented in Equation 2.5. As shown in Figure 2.2 (b), F and F^* are monotonically related thus resulting in decisions made based on F^* being identical to the decisions made based on the F-measure. Furthermore, based on Equation 2.5, F^*-measure values are always less than or equal to the precision, recall and F-measure values. Therefore, the F^*-measure is a robust and appropriate measure for evaluating our proposed clustering algorithms.

We also report the average of percentage improvements/declines in precision and recall achieved by applying temporal constraints as opposed to not applying temporal constraints. Considering the precision values achieved with a certain algorithm for a given parameter configuration disregarding (P_{NT}) and applying (P_T) temporal constraints, the average precision improvement is calculated as $(P_T - P_{NT}) \cdot 100/P_{NT}$ (where a negative outcome is interpreted as a precision decline). The percentage recall improvements can be calculated in a similar manner, whereas we report the average of such percentage precision and recall improvements across the different parameter configurations.

4.4.1 Linkage Quality Evaluation

We will now analyse the linkage quality improvements achieved by incorporating temporal constraints, compared to disregarding temporal constraints. To summarise

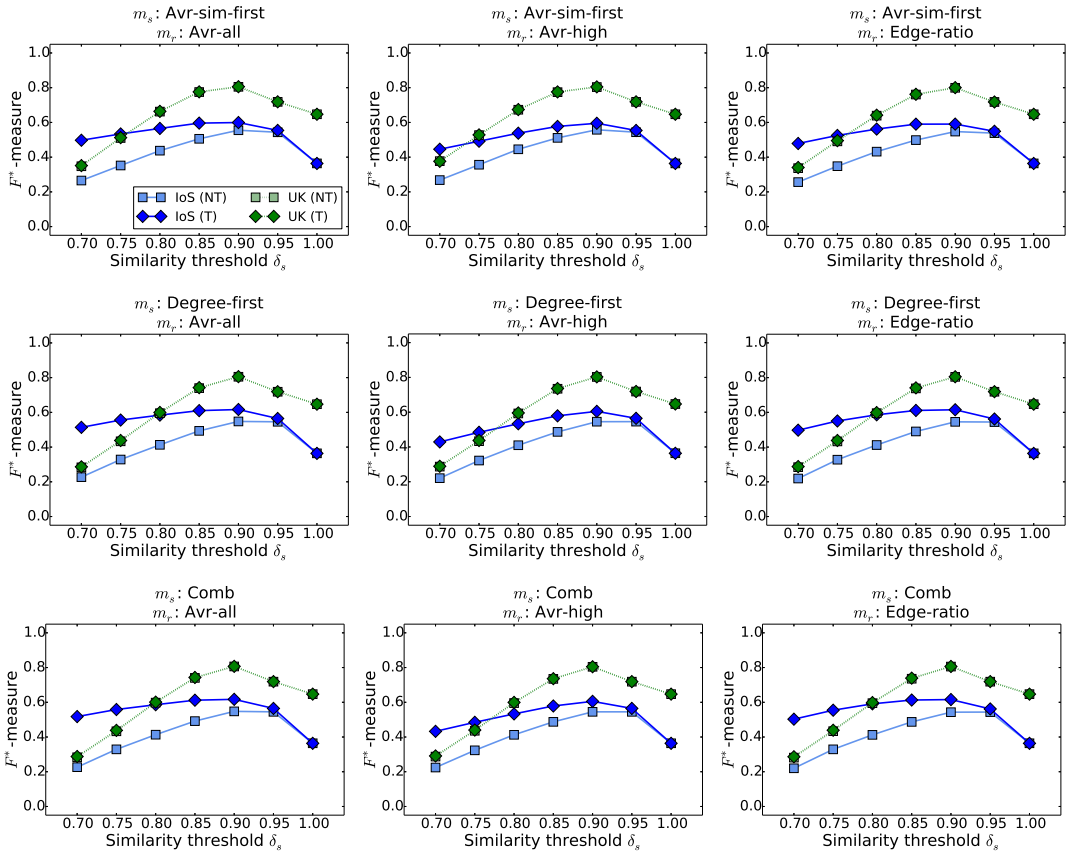

Figure 4.5: Star clustering results obtained with (T) and without (NT) temporal constraints: Average of F^* values obtained with different similarity graphs, shown for different similarity thresholds δ_s, different vertex sorting methods m_s, and different overlap cluster resolving methods m_r.

the results we show the average of the F^* values obtained for the three different similarity graphs $\mathbf{G}_A$, $\mathbf{G}_{NA}$, and $\mathbf{G}_N$.

Figure 4.4 shows the linkage quality achieved with applying the greedy clustering approach discussed in Section 4.3.1 on the IoS and UK birth data sets. With the greedy clustering approach an average percentage precision improvement of 16.72% was achieved for the IoS data set, at the cost of a 2.31% decline in recall. However, for the UK data set the precision declined by an average 0.92% whereas the recall improved by an average 0.19%. As implied by these results and the F^* values shown in Figure 4.4, the precision improvement/decline achieved with applying temporal constraints (T) was approximately equal to the decline/improvement in recall and therefore the overall linkage quality was not improved by applying temporal constraints in greedy clustering.

The linkage quality achieved with star clustering is shown in Figure 4.5. Notice how with the IoS data set the F^* values have considerably improved when temporal constraints are applied (T) compared to not applying temporal constraints (NT). The

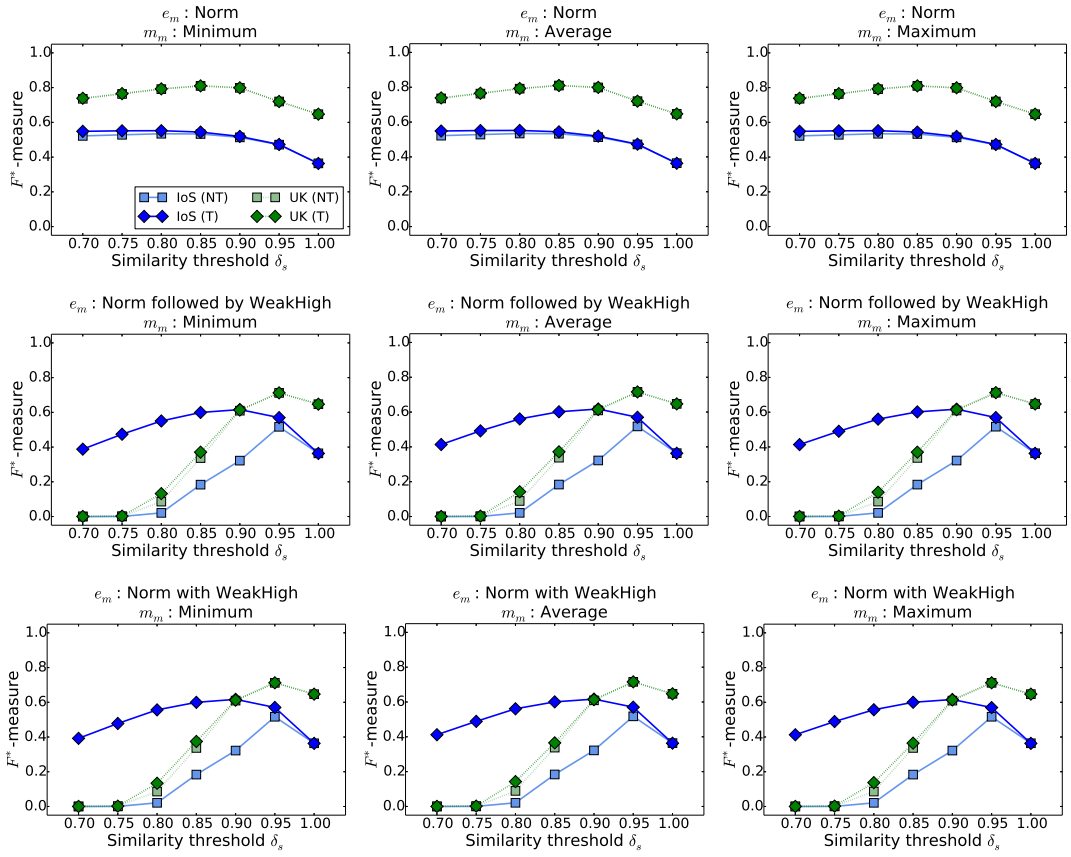

Figure 4.6: Robust graph clustering results obtained with (T) and without (NT) temporal constraints: Average of F^* values obtained with different similarity graphs, shown for different types of edge combinations to merge clusters e_m, and different cluster similarity calculation methods m_m.

improvement is greater for lower similarity thresholds δ_s since at higher threshold values the precision is already quite high even without applying temporal constraints thus giving limited allowance for further improvement. With the IoS data set, a significant average precision improvement of 74.33% was achieved at the cost of a small 1.64% decline in recall when temporal constraints were applied. With the UK data set, however, the precision improvement and decline in recall were 0.59% and 0.23% respectively, thus resulting in no significant improvement with the application of temporal constraints. We believe this behaviour to be the result of the UK data set being a synthetic one, which does not have temporal patterns as clearly defined as in real-world data sets. We further explore if temporal star clustering is effective on real-world data sets in Chapter 10.

Figure 4.6 shows the F^* values obtained with conducting RL using the robust graph clustering approach. With a set of preliminary experiments we identified that the base clusters with the highest precision were obtained when only *Strong* edges were used for base cluster generation (e_b). Therefore, we show results achieved with

Table 4.1: The parameter configurations that produced the best F^* value per each data set and each clustering algorithm. Applying temporal constraints consistently produced the best results, whereas the choice of algorithm specific parameter configurations was mostly consistent across the two data sets

Data set	Algorithm	F^*	Best parameter configurations			
			G	δ_s	Algorithm specific parameters	Run-time (sec)
IoS	Greedy	0.75	$\mathbf{G}_N$	0.95	Temporal, m_n = Next	150.82
	Star	0.83	$\mathbf{G}_N$	0.95	Temporal, m_s = Avr-sim-first, m_r = Avr-all	30.02
	Robust	0.83	$\mathbf{G}_N$	0.95	Temporal, m_m = Average, e_m = Norm with WeakHigh	1,952.42
UK	Greedy	0.89	$\mathbf{G}_A$	0.85	Temporal, m_n = Next	15.49
	Star	0.91	$\mathbf{G}_A$	0.85	Temporal, m_s = Degree-first, m_r = Avr-all	5.78
	Robust	0.87	$\mathbf{G}_A$	0.9	Temporal, m_m = Average, e_m = Norm with WeakHigh	225.28

merging *Strong* link based base clusters. Furthermore, we set the weight $w = 1.0$ such that only cluster similarity is considered in cluster merging, but not the cluster coverage since better linkage values were obtained with $w = 1.0$. While the linkage results are not significantly different for applying or disregarding temporal constraints when using the *Norm* type edges only for merging, a considerable quality improvement is indicated for the IoS data set when both *Norm* and *WeakHigh* edges are used with temporal constraints. We experiment with merging base clusters by first expanding with *Norm* edges and then with *WeakHigh* edges, and expanding with both types of edges at once. Only a minor improvement is shown when temporal constraints are applied on the UK data set, which we believe to be a result of that being a synthetic data set as highlighted before. With robust graph clustering, a large average precision improvement of 5106.43% was achieved at the cost of a 2.05% reduction in recall for the IoS data set, whereas the precision improvement and decline in recall were 18.03% and 0.08% respectively, for the UK data set. The reason for the average precision improvement achieved with the IoS data set being greater than 100% is due to the linkage precision being very low when temporal constraints are disregarded.

In Table 4.1 we present the parameter configurations per each data set and each clustering algorithm, which produced the best results with regard to the F^*-measure. Notice how the best results have consistently been achieved when temporal constraints were incorporated, with the $\mathbf{G}_N$ and $\mathbf{G}_A$ pairwise similarity graphs for the IoS and UK data sets respectively. Furthermore, the best results were obtained at higher similarity thresholds ($\delta \geq 0.85$) for both data sets. Surprisingly, the vertex selection method (m_n) *Next* produced the best linkage results in the greedy approach for both data sets. Furthermore, in the robust graph clustering approach, for both IoS and UK data sets, merging base clusters with *Norm* and *WeakHigh* edges at once (e_m), using an *Average* cluster similarity calculation method (m_m) produced the best results. With star clustering, the cluster overlap resolving method (m_r) *Avr-all* worked

Table 4.2: The minimum (Min), maximum (Max), average (Avr), and median (Med) run-times (in seconds) of each clustering algorithm per birth data set.

Data set	Algorithm	Run-time (NT)				Run-time (T)			
		Min	**Max**	**Avr**	**Med**	**Min**	**Max**	**Avr**	**Med**
IoS	Greedy	0.09	6,929.68	135.41	10.70	0.09	736.58	66.21	5.94
	Star	6.85	1,165.77	45.55	22.17	7.02	1,535.16	124.58	23.38
	Robust	156.94	77,162.53	4,273.28	274.50	1,801.49	5,501.22	2,720.17	2,545.94
UK	Greedy	0.07	13,827.01	128.42	6.21	0.07	47,634.59	399.27	11.79
	Star	5.50	31.61	8.36	7.56	5.53	330.29	24.55	7.75
	Robust	85.38	30,222.08	1,359.94	108.67	210.23	24,405.55	1,301.05	253.73

best for both data sets. The *Avr-sim-first* vertex sorting method (m_s) produced best results for the IoS data set, and the *Degree-first* method for the UK data set.

4.4.2 Run-time Evaluation

In the last column of Table 4.1, we show run-times in seconds achieved for the experiments run with the best parameter configurations. Considering the trade-off between the effectiveness and efficiency across the three proposed algorithms, it is evident that temporal Star clustering is clearly the best performer as it produces the highest linkage quality in the shortest run-times.

We present the minimum, maximum, average and median values of algorithm run-times in seconds in Table 4.2. With the IoS data set, the minimum run-times have either increased or not changed, whereas the maximum run-times have reduced for all three algorithms, when temporal constraints were applied (T) compared to not using temporal constraints (NT). The median run-time values have either decreased (for greedy clustering) or increased (for star and robust) with the application of temporal constraints due to the variations in these run-time ranges.

The minimum run-time values correspond to higher similarity thresholds δ_s, where the additional computational effort required for checking temporal constraints increases the temporal run-time. However, at lower δ_s, which correspond to the maximum run-times, the computational effort for temporal constraint checks is outweighed by the advantage of removing many temporally implausible record pairs. Therefore, the maximum temporal run-time values are less than the corresponding maximum non-temporal run-times. The average run-time values have reduced with the application of temporal constraints except for star algorithm, indicating it is often more efficient than the baseline non-temporal technique.

As a result of the reduction in the maximum run-time, the average run-time was slightly improved for conducting RL on the UK data set with robust graph clustering using temporal constraints. Otherwise, the minimum, maximum, average, and median run-time values have always increased with the application of temporal constraints compared to the non-temporal approach. This is due to the extra computational effort required for checking temporal constraints in the UK data set outweighing the benefit from the removal of temporally implausible record pairs.

Given that IoS is a real-world data set and UK is a synthetic one, the application of temporal constraints is beneficial mostly for the former data set in terms of both the linkage quality and efficiency since real-world data sets have more realistic data characteristic patterns.

4.5 Summary

In this chapter, we have presented the novel concept of applying data characteristics for RL to improve the linkage quality. This concept was presented using temporal data characteristics and the corresponding temporal constraints which are available in most population data sets. We proposed three unsupervised clustering techniques, namely greedy clustering, star clustering, and robust graph clustering, which incorporate these temporal constraints in the classification process. Empirical evaluation of the clustering techniques were conducted on one real-world and one synthetic birth data set.

The empirical evaluation showed substantial quality improvements when temporal constraints were applied on the real-world data set with the star and robust graph clustering approaches. The quality improvements were however not very significant for the synthetic data set, due to the temporal patterns not being as realistic as for the real-world data set. The greedy clustering approach too did not produce significant quality improvements with the application of temporal constraints due to the precision improvement being approximately similar to the decline in recall. Therefore, the temporal star and temporal robust graph clustering methods are more suitable to be applied on real-world RL applications. We further assess the quality of these algorithms using a novel evaluation measure in Chapter 8, and by applying them on a new real-world data set in Chapter 10.

We furthermore showed that the run-time considerably improves when temporal constraints are applied on robust graph clustering for real-world data sets. In the next chapter, we further extend our concept of using data characteristics to include transition probabilities, and conduct a comparative evaluation with the methods proposed in this chapter.

Chapter 5

Record Linkage Using Transition Probabilities on Data Characteristics

In the previous chapter we presented the novel concept of applying data characteristics to improve the quality of Record Linkage (RL) applications. In this chapter, we further develop this concept to consider the transition probability distributions corresponding to data characteristics, where we consider transitions across *states* associated with records. In Section 5.1 we provide an introduction to our concept of using state transition probabilities on data characteristics, and describe what we refer to as a state in population data. Next, in Section 5.2, we discuss how to model these transition probabilities to determine the likelihood of a state transition in the real-world (referred to as the *population goodness* of a state transition). In Section 5.3 we then discuss how this population goodness measure can be combined with attribute value-based record pair similarities to obtain a *cluster goodness* measure, which reflects the plausibility of a cluster that was generated with a RL clustering approach. Furthermore, we propose a method to incorporate this cluster goodness measure in the RL process to improve the linkage quality. Subsequently, in Section 5.4 we empirically evaluate our cluster goodness-based RL techniques and compare them with the data characteristics-based methods we proposed in the previous chapter. Finally, in Section 5.5, we conclude this chapter with a summary of our findings.

5.1 Introduction

As we discussed in the previous chapter in page 61, group RL techniques using clustering are more frequently used compared to traditional linkage approaches, due to their applicability in linking groups of individuals in population data sets. In the previous chapter, we therefore proposed a method to enhance the linkage quality achieved with unsupervised clustering techniques by incorporating data characteristics such as temporal and spatial information.

In the real-world, an entity may assume different *states*, where a state is an event or a role attached to the entity. Such states may change over the lifetime of the

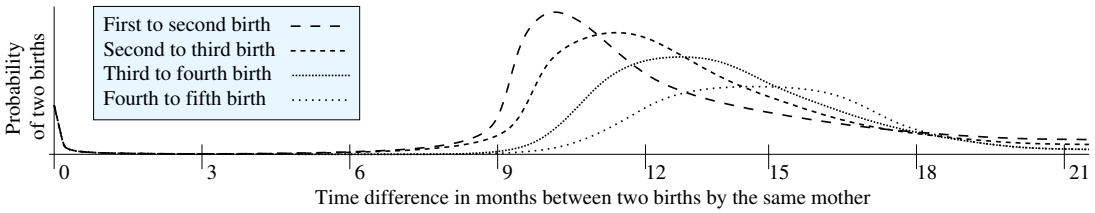

Figure 5.1: Temporal probability distributions for different pairs of birth records.

entity. For example, population data from national censuses and civil registries [149] generally contain records that describe a group of individuals, where each has a different role, such as a *baby* and her *mother* and *father*. These roles change over time, for example, once old enough the *baby* can get married (becomes a *bride* on a marriage certificate) and has her own children (becomes the *mother* on the birth certificates of her babies). Furthermore, if we consider the group RL task of bundling sibling records (births by the same parents), the birth records corresponding to each parent assume events *birth of baby 1*, *birth of baby 2* and so on, depending on the number of children per family. Considering a bibliographic data set, the authors may assume the roles of *student*, *postdoctoral researcher* and *professor* at different times. A state transition refers to transitioning from one state to another, such as a person transitioning from being a bride to a mother in a historic population data set.

In this chapter we investigate how the distribution of probabilities of transitioning between states based on a data characteristic can be incorporated with group RL techniques. We specifically explore how well a set of linked records (a group or a cluster), where each record has data characteristics, fits the distribution of data characteristics of such records as found in a large population. To achieve this task, we introduce *population goodness* measures which reflect the goodness or quality of each state transition in a cluster, compared to state transitions encountered in the real-world. For example, are the linked birth records by a mother over time what is common, or are her births timed rather unusual. This is illustrated in Figure 5.1, which shows that the first two births by a mother are more likely to occur within a shorter period of time compared to the fourth and fifth births [62].

In this chapter, we explore two novel population goodness measures based on state transition probability calculation methods. The first assumes a Markov property [150] of state transitions over data characteristic values. The second measure considers all transition probabilities from one state to future states (such as the likelihood that a mother gives birth to three more children within five years after her first birth). We assume that these probability distributions (or population goodness) can be calculated based on a publicly available linked population data set from a similar domain, or based on a domain expert's knowledge. We then develop three *cluster goodness* measures to assess the plausibility of clusters generated by a clustering technique for RL, by combining such a population goodness of fit with attribute similarities of record pairs.

5.2 Modelling Population Goodness

In this section, we describe how we model the population goodness (or the state transition probabilities with data characteristics) using two probabilistic techniques, where the first is based on Markov chains and the second on overall transition probabilities. As we described in Section 4.2, most population data sets contain temporal data characteristics, and we therefore model population goodness considering the temporal characteristics implied by the event time differences across records (such as the time difference between the birth and marriage of a person).

To calculate population goodness we assume to have a set, $\mathbf{C}_g$, of non-overlapping ground-truth clusters. Each cluster $\mathbf{c}_g \in \mathbf{C}_g$ contains a group of records, where each record $r_i \in \mathbf{c}_g$ has a time stamp, $r_i.t$ and an event type (or a state), $r_i.\mathcal{s}$. The set of all states, $\boldsymbol{\mathcal{S}} = \{\mathcal{s}_1, \mathcal{s}_2, ..., \mathcal{s}_n\}$ contains all possible event types, such as *birth*, *marriage*, *birth of a baby*, and *death* [149].

5.2.1 Markov Chain-based Population Goodness

Here we assume the simplified view that events (or states) in people's lives follow a Markov process [72], where the probability of a certain event occurring only depends upon the previous event in the person's life.

For a set of ground-truth clusters, $\mathbf{C}_g$, the Markov state transition probability of $\mathcal{s}_x \rightarrow \mathcal{s}_y$ (from $\mathcal{s}_x$ to $\mathcal{s}_y$) is calculated as the ratio of the total number of consecutive transitions from $\mathcal{s}_x \rightarrow \mathcal{s}_y$ across all ground-truth clusters $\mathbf{c}_g \in \mathbf{C}_g$, divided by the number of all consecutive transitions from $\mathcal{s}_x$ to any other state $\mathcal{s}_z \in \boldsymbol{\mathcal{S}}$. Consecutive transitions are identified using $r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}$ where $(r_i, r_j) \in \mathbf{c}_g, \forall \mathbf{c}_g \in \mathbf{C}_g$, and where $r_i.t \leq r_j.t$. Furthermore, r_i and r_j need to occur consecutively in time and no record in $\mathbf{c}_g$ can be between them. This means $\nexists r_x$ such that $r_i.t \leq r_x.t \wedge r_x.t \leq r_j.t$. We can then calculate such a Markov-based population goodness at two levels:

- **Markov chain independent of time**: In this method, probabilities (or population goodness values) are calculated for each consecutive state transition $\mathcal{s}_x \rightarrow \mathcal{s}_y$ in all record pairs (r_i, r_j) in all $\mathbf{c}_g \in \mathbf{C}_g$. These transitions are assumed to hold the Markov property with the requirements discussed above, but we do not consider the actual time when these records occur.

$$PG_m(\mathcal{s}_x \rightarrow \mathcal{s}_y) = \frac{|\forall(r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}), r_i.\mathcal{s} = \mathcal{s}_x, r_j.\mathcal{s} = \mathcal{s}_y|}{|\forall(r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}), r_i.\mathcal{s} = \mathcal{s}_x, r_j.\mathcal{s} = \mathcal{s}_z, \forall \mathcal{s}_z \in \boldsymbol{\mathcal{S}}|}. \tag{5.1}$$

- **Markov chain based on time**: Probabilities for each consecutive state transition $\mathcal{s}_x \rightarrow \mathcal{s}_y$ can also be calculated for a specific time interval (or temporal difference), Δt. We use kernel density estimation (KDE) [159] to obtain smoothed probability curves (as shown in Figure 5.1) to reduce overfitting the ground-truth. Furthermore, we assume that $r_i.t \leq r_j.t$.

$$PG_{mt}(\mathcal{s}_x \rightarrow \mathcal{s}_y, \Delta t) = \frac{|\forall(r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}), r_i.\mathcal{s} = \mathcal{s}_x, r_j.\mathcal{s} = \mathcal{s}_y, (r_j.t - r_i.t) = \Delta t|}{|\forall(r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}), r_i.\mathcal{s} = \mathcal{s}_x, r_j.\mathcal{s} = \mathcal{s}_y|}. \tag{5.2}$$

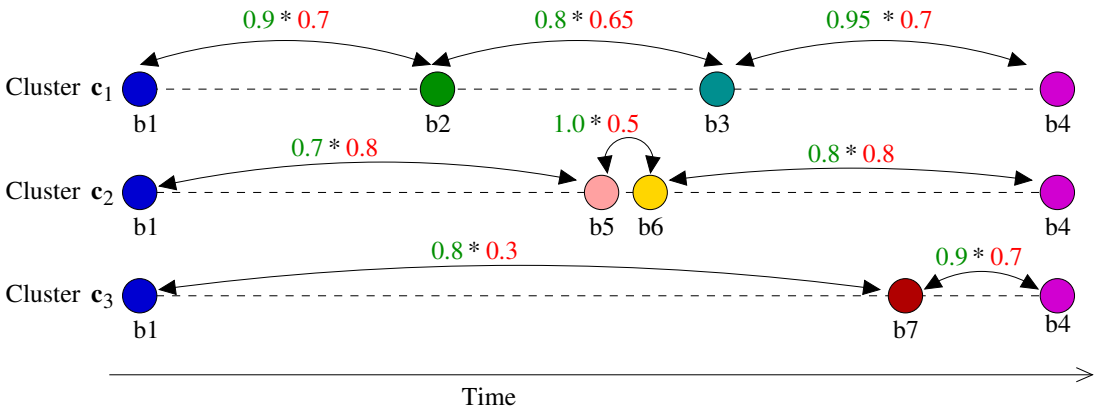

Figure 5.2: Example of overlapping clusters (all three contain birth records b1 and b4) with pairwise attribute similarities multiplied by population goodness.

For the example cluster $\mathbf{c}_1$ in Figure 5.2, the Markov transitions consist of b1 $\rightarrow$ b2, b2 $\rightarrow$ b3, and b3 $\rightarrow$ b4. The Markov chain-based population goodness calculation method is justifiable when we assume that the state transitions encountered in a population are influenced only by the current state. For example, with this approach we can calculate the probability of a person immediately transitioning to becoming a *professor* from his or her current occupation.

5.2.2 Overall Transition Probability-based Population Goodness

Transition probabilities can also be applied independent of the Markov property. For a set of ground-truth clusters, $\mathbf{C}_g$, the overall state transition probability of $\mathcal{s}_x \rightarrow \mathcal{s}_y$ can be calculated as the ratio of the total number of transitions from $\mathcal{s}_x \rightarrow \mathcal{s}_y$ across all ground-truth clusters $\mathbf{c}_g \in \mathbf{C}_g$, over the total number of times state $\mathcal{s}_x$ was encountered in clusters in $\mathbf{C}_g$. Overall transitions within ground-truth clusters are identified using $r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}$ where $(r_i, r_j) \in \mathbf{c}_g, \forall \mathbf{c}_g \in \mathbf{C}_g$, such that $r_i.t \leq r_j.t$ (this means state or event $r_j.\mathcal{s}$ occurs after $r_i.\mathcal{s}$ but not necessarily consecutively). Such an overall transition probability-based population goodness can again be calculated on two levels:

- **Overall transition probability independent of time**: In this method, probabilities are calculated for every possible state transition $\mathcal{s}_x \rightarrow \mathcal{s}_y$, where state $\mathcal{s}_x$ is encountered before $\mathcal{s}_y$.

$$PG_o(\mathcal{s}_x \rightarrow \mathcal{s}_y) = \frac{|\forall(r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}), r_i.\mathcal{s} = \mathcal{s}_x, r_j.\mathcal{s} = \mathcal{s}_y|}{|\forall r_i \in \mathbf{c}_g, \forall \mathbf{c}_g \in \mathbf{C}_g, r_i.\mathcal{s} = \mathcal{s}_x|}. \tag{5.3}$$

- **Overall transition probability based on time**: Probabilities are calculated for every state transition $\mathcal{s}_x \rightarrow \mathcal{s}_y$ for a given time interval, Δt. For example, we can calculate the probability of a death event following a birth event either within a

few days or many years apart (where other states or events for a person might have occurred in between, such as a marriage or the birth of a baby). We again apply KDE to minimise overfitting.

$$PG_{ot}(\mathcal{s}_x \rightarrow \mathcal{s}_y, \Delta t) = \frac{|\forall(r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}), r_i.\mathcal{s} = \mathcal{s}_x, r_j.\mathcal{s} = \mathcal{s}_y, (r_j.t - r_i.t) = \Delta t|}{|\forall(r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}), r_i.\mathcal{s} = \mathcal{s}_x, r_j.\mathcal{s} = \mathcal{s}_y|}. \quad (5.4)$$

For the example cluster $\mathbf{c}_1$ in Figure 5.2, the overall transitions comprise of b1 $\rightarrow$ b2, b1 $\rightarrow$ b3, b1 $\rightarrow$ b4, b2 $\rightarrow$ b3, b2 $\rightarrow$ b4, and b3 $\rightarrow$ b4. Unlike in the Markov method, the overall transition probability-based population goodness calculation method is justifiable when we assume that all previous states encountered in a population data set influence the transitions to future state. For example, with this approach we can calculate the probability of a person transitioning from any occupation to being a *professor* in the future, regardless of the other occupations they may hold in between. Modelling population goodness using both these methods is useful for identifying which is more applicable in real-world population data sets.

5.3 Record Linkage Clustering with Population Goodness

We now describe how the population goodness measures can be used in the RL process. As we described in the previous section, these measures reflect the likelihood of state transitions in a population data set based on a domain expert's knowledge, or a publicly available linked data set from a related domain. Our aim is to refine the clusters generated by a RL clustering technique based on the knowledge of these state transition probabilities. Therefore, we propose a method to rank clusters generated by a RL clustering technique that generates overlapping clusters, such that clusters with a higher ranking can be retained and the others can be discarded. The methods we propose in this section are applicable to any clustering technique for RL that generates a set of overlapping clusters $\mathbf{C}_o$. Therefore, the clustering algorithm can be treated as a *black box* which produces overlapping clusters.

We first discuss the three methods to calculate the *cluster goodness* (i.e. goodness of a predicted cluster) based on the population goodness measures. Next, we describe how clusters are ranked based on the selected cluster goodness measure to refine clusters, such that cluster overlaps are resolved. In the following we denote the size of an overlapping cluster $\mathbf{c}_o \in \mathbf{C}_o$ by $n = |\mathbf{c}_o|$.

5.3.1 Markov Chain-based Cluster Goodness (MC)

In this approach we use the Markov chain-based population goodness calculation described in Section 5.2.1 to determine the probability for a record to transition from one state to another, either independent of (Equation 5.1), or dependent on (Equation 5.2) time. For every consecutive record pair $(r_i, r_j) \in \mathbf{c}_o, \forall \mathbf{c}_o \in \mathbf{C}_o$, where $r_i.t \leq r_j.t$ and $\nexists r_x$ such that $r_i.t \leq r_x.t \wedge r_x.t \leq r_j.t$, the transition probability $p_{i,j}$ is obtained for transition $r_i.\mathcal{s} \rightarrow r_j.\mathcal{s}$, using the Markov-based population goodness calculation in Equation 5.1 or Equation 5.2, where the similarity $s_{i,j}$ for (r_i, r_j) is obtained

from a pairwise similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ as we described in Section 2.3.3. The overall *cluster goodness* for cluster $\mathbf{c}_o$ is then calculated as:

$$CG_{MC}(\mathbf{c}_o) = \sum_{(r_i,r_j)\in\mathbf{c}_o} \frac{p_{i,j} \cdot s_{i,j}}{n-1}. \tag{5.5}$$

The denominator in Equation 5.5 is $n-1$ since there are $n-1$ consecutive (ordered) record pairs in a cluster with n records. Once we have calculated cluster goodness for each cluster, we order (rank) all clusters by their cluster goodness (with highest goodness first) and iteratively select the best cluster into a final set of non-overlapping clusters. Any record encountered in a selected cluster is marked as being assigned, and when we select the next best cluster we check for already assigned records and remove these from the selected cluster. This ensures that no clusters are overlapping.

Figure 5.2 provides an example of such a Markov chain-based cluster goodness calculation, where the goodness of cluster $\mathbf{c}_1$ is calculated as $CG_{MC}(\mathbf{c}_1) = ((0.9 \times 0.7) + (0.8 \times 0.65) + (0.95 \times 0.7))/3$ which is equal to 0.605. Similarly, the goodness of clusters $\mathbf{c}_2$ and $\mathbf{c}_3$ can be calculated as $CG_{MC}(\mathbf{c}_2) = 0.567$ and $CG_{MC}(\mathbf{c}_3) = 0.435$ respectively, and therefore cluster $\mathbf{c}_1$ with the highest goodness is selected into the final set of clusters.

5.3.2 All Pairs-based Overall Cluster Goodness (AP)

This approach uses the overall transition probability-based population goodness calculation described in Section 5.2 to determine state transition probabilities, independent of (Equation 5.3), or dependent on (Equation 5.4) time. For every record pair $(r_i, r_j) \in \mathbf{c}_o, \forall \mathbf{c}_o \in \mathbf{C}_o$, where $r_i.t \leq r_j.t$, the transition probability $p_{i,j}$ is obtained for $r_i.\mathcal{S} \rightarrow r_j.\mathcal{S}$, using the overall transition-based population goodness calculation, whereas the similarity $s_{i,j}$ for (r_i, r_j) is again obtained from the pairwise similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Note that for record pairs which are potentially removed from the pairwise similarity graph $\mathbf{G}$ due to application of methods such as blocking or indexing (described in Section 2.3), we set $s_{i,j} = 0.0$. The overall cluster goodness for $\mathbf{c}_o$ is then calculated as:

$$CG_{AP}(\mathbf{c}_o) = \sum_{(r_i,r_j)\in\mathbf{c}_o} \frac{p_{i,j} \cdot s_{i,j} \cdot 2}{n \cdot (n-1)}. \tag{5.6}$$

The denominator in Equation 5.6 is $n \cdot (n-1)/2$ since there are that many record pairs in total in a cluster with n records. To resolve overlaps, the cluster with the highest goodness is iteratively selected. As with the Markov chain method described above, once a record has been assigned to a cluster of the final cluster set it is removed from all other clusters to ensure no cluster is overlapping.

Algorithm 5: Resolving cluster overlaps with goodness measures

Input: $\mathbf{G}$ - Undirected pairwise similarity graph
$\mathbf{C}_o$ - List of overlapping clusters
$\mathcal{P}$ - Transition probability matrix
f - Flag to indicate dependence on time
m - Cluster goodness measure (one of *MC*, *AP*, or *RB*)
Output: $\mathcal{C}$ - List of non-overlapping clusters

$\mathbf{C} = [\,]$ // Initialise an empty list of non-overlapping clusters
$\mathcal{I} = \{\}$ // Initialise index of goodness per record or cluster
for $\mathbf{c}_o \in \mathbf{C}_o$ **do** // Iterate through the overlapping clusters
  for $(r_i, r_j) \in \mathbf{c}_o$ **do** // Iterate through each record pair in the cluster
    $s_{i,j} = \textbf{GetPairwiseSim}(\mathbf{G}.\mathbf{E}, (r_i, r_j))$ // Get similarity of record pair
    $p_{i,j} = \textbf{GetTransitionProb}(\mathcal{P}, (r_i, r_j), f)$ // Get transition probability
    if $m = RB$ **then** // Update goodness per each record in a record pair
      UpdateGoodnessPerRecord$(\mathcal{I}, r_i, r_j, \mathbf{c}_o, s_{i,j}, p_{i,j})$
    else // Update goodness per cluster
      UpdateGoodnessPerCluster$(\mathcal{I}, \mathbf{c}_o, s_{i,j}, p_{i,j})$
if $(m = RB)$ **then** // Resolve cluster overlaps as per record level goodness
  for $r_i \in \mathbf{G}.\mathbf{V}$ **do** // Only keep a record in its best cluster
    $\mathbf{C} = \textbf{RemoveOverlappingRec}(\mathbf{C}_o, r_i, \mathcal{I})$
else // Resolve cluster overlaps as per cluster level goodness
  $\mathbf{C}'_o = \textbf{SortByGoodness}(\mathcal{I})$ // Get clusters sorted by their goodness
  for $\mathbf{c}_o \in \mathbf{C}'_o$ **do** // Add clusters without already assigned records
    $\mathbf{C}.add(\mathbf{c}_o \setminus \textbf{RecordsInClusters}(\mathbf{C}))$
return $\mathbf{C}$

5.3.3 Record-based Overall Cluster Goodness (RB)

This approach uses the overall transition probability-based population goodness calculation described in Section 5.2 to determine state transition probabilities, independent of (Equation 5.3), or dependent on (Equation 5.4) time. For each record $r_i \in \mathbf{c}_o$, its pairwise record similarities, $s_{i,j}$, and transition probabilities, $p_{i,j}$ with every other record $r_j \in \mathbf{c}_o$, $r_i \neq r_j$, are obtained based on their time-stamps, $r_i.t$ and $r_j.t$, and states, $r_i.\mathcal{s}$ and $r_j.\mathcal{s}$. Instead of an overall cluster goodness we calculate a *record level goodness* for each record r_i in cluster $\mathbf{c}_o$ as:

$$CG_{RB}(r_i, \mathbf{c}_o) = \sum_{r_j \in \mathbf{c}_o} \frac{p_{i,j} \cdot s_{i,j}}{n-1}. \tag{5.7}$$

Subsequently, we retain each record r_i in the cluster $\mathbf{c}_o$ only where it has the highest cluster goodness and remove this record from all other clusters. At the end of this process we again obtain a set of final, non-overlapping clusters.

Algorithm 5 outlines how our three cluster goodness measures *MC*, *AP*, and *RB* are used to resolve cluster overlaps. The input to the algorithm includes a pairwise similarity graph, $\mathbf{G}$, a list of overlapping clusters, $\mathbf{C}_o$, and the transition probability

matrix, $\mathcal{P}$ containing population goodness values calculated as described in Section 5.2. The flag f identifies whether population goodness was calculated dependent on or independent of time, whereas parameter m specifies the cluster goodness measure to use. The output of the algorithm is a set of non-overlapping clusters, $\mathbf{C}$.

In lines 1 and 2, we initialise an empty list $\mathbf{C}$ to contain non-overlapping clusters, and an empty index $\mathcal{I}$ to hold goodness values (per record within a cluster if $m = RB$, or per cluster otherwise). In lines 3 to 6 we obtain the pairwise record similarities $s_{i,j}$ and transition probabilities $p_{i,j}$ for every record pair $(r_i, r_j) \in \mathbf{c}_o, \forall \mathbf{c}_o \in \mathbf{C}_o$. Then, in lines 7 and 8, if $m = RB$, we calculate the goodness for each record r_i and r_j (in record pair (r_i, r_j)) in cluster $\mathbf{c}_o$ as per Equation 5.7 for the *RB* method, and store this value in index $\mathcal{I}$. Otherwise, if $m = MC$ or $m = AP$, the overall goodness is calculated for cluster $\mathbf{c}_o$ in lines 9 and 10 and stored in $\mathcal{I}$.

Lines 11 to 13 execute the cluster overlap resolving for the *RB* method, as described in Section 5.3.3, whereas lines 14 to 17 execute the *MC* or *AP* method, as described in Sections 5.3.1 and 5.3.2 respectively. Note that the distinction between the *MC* and *AP* methods comes from the population goodness values contained in $\mathcal{I}$. Finally, a list of non-overlapping clusters $\mathbf{C}$ is returned in line 18.

Complexity analysis: We now conduct a complexity analysis of our proposed cluster goodness measures, and the cluster overlap resolving technique outlined in Algorithm 5. Note that we do not conduct a complexity analysis of the population goodness calculation we discussed in Section 5.2 since it is a one time calculation, based on a domain expert's knowledge or an already available public linked data set, which can be reused in several RL projects. The cluster goodness measure *MC* has a complexity of $O(n \cdot |\mathbf{C}_o|)$ (where $n = |\mathbf{c}_o|$) given that only consecutive record pairs in each overlapping cluster $\mathbf{c}_o \in \mathbf{C}_o$ are considered as shown in Equation 5.5. For the cluster goodness measures *AP* and *RB*, however, the complexity is $O(n^2 \cdot |\mathbf{C}_o|)$ given that every record pair in a cluster is considered, as shown in Equations 5.6 and 5.7.

In lines 3 to 10 of Algorithm 5, we iterate through every record pair in each overlapping cluster $\mathbf{c}_o \in \mathbf{C}_o$. There are $n \cdot (n-1)/2$ comparisons per cluster $\mathbf{c}_o$. Therefore, the for loop from lines 3 to 10 has a time complexity of $O(n^2 \cdot |\mathbf{C}_o|)$. For the cluster goodness measure $m = RB$, we iterate over every record (or vertex) in the similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ in lines 11 to 13, which has a complexity of $O(|\mathbf{V}|)$. For the cluster goodness measures $m = MC$ and $m = AP$, we first sort the clusters by their goodness values in line 15, which has a time complexity of $O(|\mathbf{C}_o| \cdot log(|\mathbf{C}_o|))$, and then we iterate over each cluster in lines 16 and 17 which has a time complexity of $O(|\mathbf{C}_o|)$. Therefore, the total time complexity for conducting cluster overlap resolving with the *MC*, *AP*, and *RB* cluster goodness measures are $O(n^2 \cdot |\mathbf{C}_o| + |\mathbf{C}_o| \cdot log(|\mathbf{C}_o|))$, $O(2 \cdot n^2 \cdot |\mathbf{C}_o| + |\mathbf{C}_o| \cdot log(|\mathbf{C}_o|))$, and $O(2 \cdot n^2 \cdot |\mathbf{C}_o| + |\mathbf{V}|)$ respectively.

5.4 Experimental Evaluation

In this section, we present the results obtained with an empirical evaluation of cluster overlap resolving using our three cluster goodness measures. We conducted ex-

periments on the real-world Isle of Skye (IoS) and the synthetic UK birth data sets which we described in Section 2.5.1. Furthermore, we implemented our algorithms in Python 2.7, and all experiments were conducted on a server running Ubuntu 18.04 with 64-bit Intel Xeon 2.10 GHz CPUs and 512 GB of memory.

As we discussed in Section 5.3, our proposed methods can be combined with RL clustering techniques that generate overlapping clusters. Since the temporal star clustering technique which we proposed in Chapter 4 generates overlapping clusters, we use this clustering algorithm to evaluate cluster overlap resolving with our proposed cluster goodness measures. Note that we therefore disregard the overlap resolving methods *Avr-all*, *Avr-high*, and *Edge-ratio* which we used with the star approach in the previous chapter (see page 70) and replace them with the method proposed in Algorithm 5 using one of the cluster goodness measures *MC*, *AP*, or *RB*. As the baseline technique we used the star clustering approach with temporal (T) data characteristics using the best algorithm specific parameter configurations and similarity graph for each data set, as we specified in the previous chapter in Table 4.1. The same configurations were used with the technique outlined in Algorithm 5 to comparatively evaluate our overlap resolving technique with the baseline approach. Furthermore, we evaluated different similarity threshold values δ_s (see Algorithm 2) ranging from 0.7 to 1.0 in steps of 0.05 in star clustering.

For evaluation we used the F^*-measure presented in Equation 2.5 since it is monotonically related to, but is relatively more robust, than the F-measure in evaluating RL outcomes, as we discussed in the previous chapter (see page 77). Furthermore, we report the run-times of our algorithm for efficiency analysis.

Given that the aim of linking the IoS and UK birth data sets is identifying births by the same parents, the state transitions for these data sets are of the format *birth of baby* $x \rightarrow$ *birth of baby* $x + 1$ for the Markov chain-based probability calculations (Equations 5.1 and 5.2), and *birth of baby* $x \rightarrow$ *birth of baby* y where $y > x$ for the overall transition probability-based calculations (Equations 5.3 and 5.4). For each cluster goodness measure (*MC*, *AP*, and *RB*) we ran cluster overlap resolving experiments dependent on and independent of time, where each experiment was further split into two cases, where pairwise record similarities $s_{i,j}$ in Equations 5.5 to 5.7 were either set to 1.0 (to measure the effect of population goodness only), or retrieved from the pairwise similarity graph edge weights $\mathbf{G}.\mathbf{E}$.

5.4.1 Linkage Quality Evaluation

Figure 5.3 shows the linkage quality achieved when cluster overlap resolving was conducted with our proposed cluster goodness measures *MC*, *AP*, and *RB*, using different parameter configurations on the IoS (top six plots) and UK (bottom six plots) birth data sets. For the IoS data set, the baseline temporal constraints-based (T) star clustering approach has clearly performed better compared to the methods we proposed in this chapter. However, with the UK data set, we can observe improved quality with our proposed methods at lower similarity thresholds δ_s.

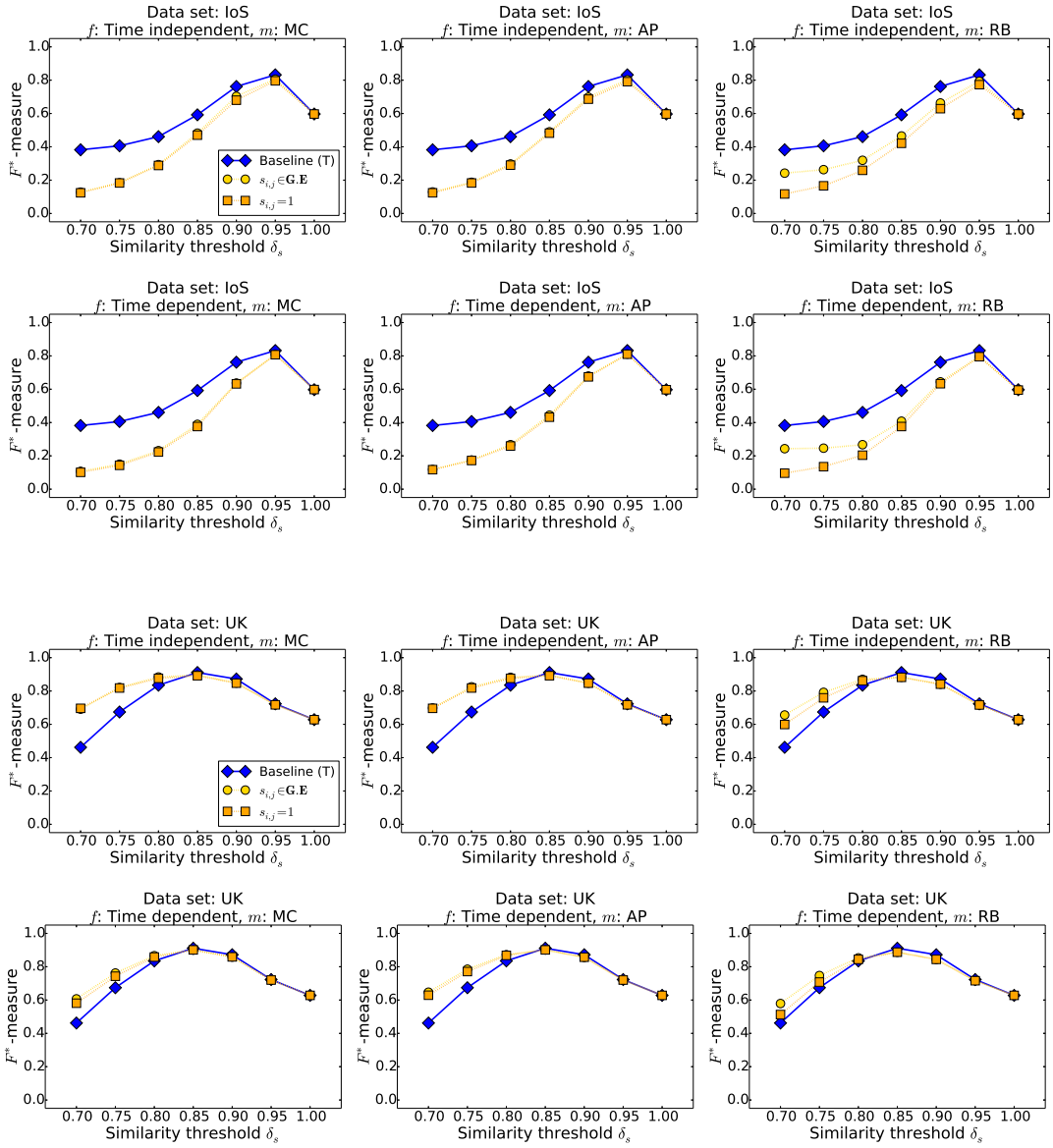

Figure 5.3: Clustering with transition probabilities: F^* values obtained for different similarity threshold values δ_s (discussed in Chapter 4), shown for time independent and dependent population goodness calculations (f) for different cluster goodness measures (m). The pairwise similarity is either set to a constant value ($s_{i,j} = 1$) or taken from the edge weights in the pairwise similarity graph ($s_{i,j} \in \mathbf{G}.\mathbf{E}$) in the cluster goodness calculations, and clustering with temporal (T) constraints (proposed in Chapter 4) is used as the baseline.

Further investigation on this behaviour showed that for most similarity thresholds, the slight improvement in precision is outweighed by a considerable decrease in recall. This behaviour is potentially caused by the overfitting that occurs in the real-world IoS birth data set that has a few infeasible patterns in the birth transition

Table 5.1: The minimum (Min), maximum (Max), average (Avr), and median (Med) run-times (in seconds) of star clustering using the goodness measures *MC*, *AP*, and *RB* for cluster overlap resolving, for the IoS and UK birth data sets.

Data set and baseline run-times	**Algorithm**	**Run-time**			
		Min	**Max**	**Avr**	**Med**
IoS	MC	6.90	8,369.94	638.16	27.21
Min = 6.85, Max = 1,165.77	AP	6.92	8,530.93	638.21	29.18
Avr = 45.55, Med = 22.17	RB	6.79	8,782.38	644.65	29.22
UK	MC	5.48	18,319.60	1,087.24	12.79
Min = 5.50, Max = 31.61	AP	5.42	19,010.33	1,030.46	10.02
Avr = 8.36, Med = 7.56	RB	5.48	21,529.71	1,116.76	11.57

distributions due to data quality issues (such as siblings with a birth difference of five days). When there are errors in the transition distributions, a relatively higher cluster goodness can be assigned to record groups which are not feasible in the real-world. Therefore, overlapping records can be removed from the correct clusters due to them having lower cluster goodness, which can result in lower recall. However, in the UK data set, for lower thresholds, the precision improves significantly at the cost of only a slight decline in recall, due to the absence of infeasible patterns in the birth transition distributions.

Including the pairwise similarity in the cluster goodness measure ($s_{i,j} \in \mathbf{G}.\mathbf{E}$) has produced slightly better results compared to considering only the population goodness of state transitions ($s_{i,j} = 1$) in the *RB* method only, for both the IoS and UK data sets. Given that the data characteristics-based clustering methods we proposed in the previous chapter have produced better results compared to the transition probability-based methods we proposed in this chapter (especially for the real-world IoS data set) we can conclude that incorporating data characteristics alone is adequate for improving RL results. Furthermore, even small mistakes in the data used for calculating population goodness can significantly reduce results quality.

5.4.2 Run-time Evaluation

Table 5.1 shows the minimum, maximum, average, and median run-times obtained for star clustering incorporating overlap resolving with our proposed cluster goodness measures *MC*, *AP*, and *RB* for each birth data set. Note that these run-times also include the time taken to generate the overlapping star clusters. For both data sets, the time taken for executing star clustering with our proposed overlap resolving methods is considerably greater than the time taken to run the baseline star clustering approach with temporal constraints, especially considering the maximum and average run-times.

The run-times do not vary largely across the different algorithms for a given data set. Interestingly, the average algorithm run-times obtained with the IoS data set is less than the average run-times obtained with the UK data set, even though a higher run-time was reported for the IoS data set for the baseline approach. This is probably

due to a higher number of overlapping clusters being generated with the UK data set compared to the IoS data set, which results in more time being consumed for the overlap resolving. The run-time results also suggest that applying data characteristics alone, as proposed in the previous chapter is more efficient and effective in the RL context, as opposed to considering the transition probabilities on data characteristics.

5.5 Summary

In this chapter, we have extended the concept of applying data characteristics for RL which we presented in the previous chapter, by including the transition probabilities on data characteristics. We proposed two *population goodness* measures which can be used to calculate the likelihood of state transitions encountered in the real-world based on a linked population data set or domain experts' knowledge. We then proposed three *cluster goodness* measures which combine the population goodness values with pairwise similarities to assess the overall goodness of a cluster generated by a RL clustering technique. We also developed a method to use these cluster goodness measures to resolve cluster overlaps in clustering techniques that generate overlapping clusters.

We empirically evaluated our proposed goodness-based cluster overlap resolving methods using one real-world and one synthetic birth data set. As the baseline, we used the data characteristics-based star clustering method which we presented in the previous chapter. We observed linkage quality improvements at lower similarity thresholds for the synthetic birth data set when our proposed cluster overlap resolving methods were applied, compared to the baseline technique. However, the methods we proposed in this chapter were less effective and efficient compared to the baseline for all other experiments. Therefore, we can conclude that incorporating data characteristics alone as proposed in the previous chapter produces better linkage results compared to applying transition probabilities on data characteristics. However, we further explore the applicability of our transition probability-based methods on real-world data sets in Chapter 10. In the next chapter we explore how these data characteristics can be utilised to improve the efficiency of population RL using an active learning approach.

Chapter 6

Active Learning-based Graph Filtering for Record Linkage

As we highlighted in Section 1.2, the quadratic time complexity of conducting naïve pairwise comparison of records has made the application of methods such as blocking and indexing a requirement (as we discussed in Section 2.3) to improve the scalability of Record Linkage (RL) projects. However, as we also highlighted in Section 1.2, the majority of record pairs retained for comparison even after applying blocking and indexing methods still correspond to true non-matches due to the class imbalance in RL projects, which is detrimental to the efficiency of the RL classification step. In this chapter, we present a novel approach that, based on the expected number of true matches between databases to be linked, applies *active learning* to remove compared record pairs that are likely non-matches before a computationally expensive classification or clustering algorithm is employed to classify record pairs.

In Section 6.1, we provide an introduction to our method of filtering record pairs using an active learning strategy. Next, in Section 6.2, we discuss in detail our proposed active learning-based record pair filtering method. In Section 6.3, we then conduct an empirical evaluation and compare the efficiency improvement obtained when RL classification is conducted after applying our filtering technique as opposed to conducting RL classification without filtering. Finally, in Section 6.4, we conclude this chapter with a summary of our findings.

6.1 Introduction

Due to the quadratic time complexity when comparing every possible pair of records across two databases to be linked, the comparison step in RL is often preceded by a *blocking* or *indexing* step [138] as we discussed in Section 2.3, where similar records are grouped into blocks such that only pairs of records within a block are compared. Additional *meta-blocking* [61] methods can be applied to further reduce the number of record pairs that need to be compared by analysing records within and across blocks to prevent redundant and superfluous record pair comparisons [138].

As we discussed in Section 2.3, the pairwise comparison step of RL then consists of the calculation of similarities between two records, generally using string com-

parison functions applied on attributes such as names and addresses [35]. A pairwise similarity graph **G** can then be generated as shown in Definition 2 on page 20, where nodes correspond to records and edges to the calculated similarities between them. However, even with blocking, indexing, and meta-blocking applied, many of these similarities will be low, and furthermore they do not correspond to true matches [35, 138]. In the classification step all compared record pairs are then classified as *matches* (records assumed to correspond to the same entity) or *non-matches* (records assumed to correspond to different entities) using a decision model that can be as simple as a similarity threshold, that can take match and error probabilities into account, or that uses training data to learn a supervised classification model [35].

While blocking, indexing, and meta-blocking can significantly reduce the number of record pairs that need to be compared in the comparison step, the similarity graph generated from pairwise comparisons can still be very large. As we showed in Table 2.4 on page 30, applying a min-hashing based Locality Sensitive Hashing (LSH) [107] blocking technique on the birth data sets we use for evaluation in this thesis resulted in similarity graphs **G** containing over four million edges (record pairs), even though the number of true matches is less than 69,000, as shown in Table 2.1 on page 25. Such large graphs are commonly required to ensure that the majority of true matches are included in order to obtain a high recall of the final linkage results [35].

Large similarity graphs can, however, challenge any algorithm used to classify record pairs because these graphs are likely very imbalanced and contain a majority of non-matching record pairs. The size of these graphs can also result in the classification step to become the computational bottleneck of the RL process [58].

In our work we remove record pairs from a similarity graph that are unlikely true matches before this graph is being used for clustering or classification. We assume that for a given linkage problem an approximate number of expected true matches can be obtained from a domain expert. For example, when linking product databases from two online stores (where one-to-one links are expected), then the number of true matches is limited by the number of records in the smaller of two databases being linked. On the other hand, when linking birth records of families, then the known distribution of family sizes in a population can be used to estimate an expected number of true matches [149].

We develop an active learning process where we bin the record pairs in a similarity graph according to a suitable data dimension, such as time or space. For example, work on temporal linkage [94] has shown that people will move over time and possibly even change their names, resulting in lower similarities for true matches. Similarly, if people move longer distances then a larger number of their address details will change (such as state or even country). Our approach recursively splits a similarity graph into bins, where we then obtain, via active learning, information from a domain expert about the distribution of matches and non-matches in these bins. We finally select a desired number of record pairs with the highest similarities from each bin, resulting in a much reduced similarity graph that still has a high recall, which facilitates accurate clustering or classification with substantially reduced run-times.

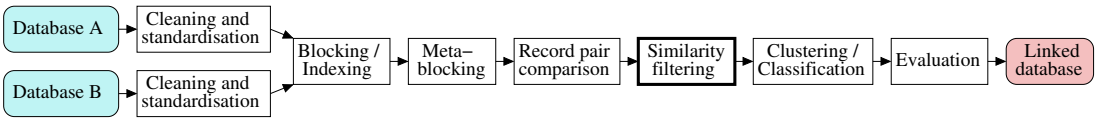

Figure 6.1: The steps of the record linkage process, with our contribution highlighted.

6.2 Active Learning-based Record Pair Filtering

We now describe our record pair similarity filtering approach based on domain knowledge. Domain experts often have a good understanding about what the number of true matches in their databases might be, depending upon the linkage situation (such as one-to-one or many-to-many links) and application [35]. As shown in Figure 6.1, similarity filtering is an additional step applied between the comparison and classification steps in the RL process [35]. The aim of filtering is to improve effectiveness and run-time of the classification step by reducing the number of non-matching record pairs (represented by their similarities) that are given to a classification or clustering algorithm [15, 58, 86, 154].

6.2.1 Problem Definition

We assume that for a data set(s) $\mathbf{D}$ to be linked, a pairwise similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ has been generated as described in Definition 2 on page 20, where a vertex $v_i \in \mathbf{V}$ represents a record in the data set $r_i \in \mathbf{D}$ ($r_i = v_i$) and an edge represents a record pair $(r_i, r_j) \in \mathbf{E}$. Furthermore, we assume that an edge represents the overall attribute similarity of a record pair $s_{i,j}$ as we showed in Definitions 1 and 2. We also assume each record pair has a distance, $d_{i,j}$, in a specific data dimension, such as time and/or space. For example, records about people often contain addresses, and using geocoding [102] these can be used to calculate geographical distances between records. Similarly, for records that contain time-stamps (such as publication records, birth, marriage, or death certificates, or census records) temporal distances can be calculated between record pairs [94]. The problem we aim to solve can now be defined as follows.

Definition 4 (Similarity Graph Filtering) *Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a pairwise similarity graph generated as shown in Definition 2, β_t be a budget of the number of manual classifications of record pairs that can be conducted by an oracle, o_m be the expected number of true matches in $\mathbf{G}$, and ϵ be a multiplier for the number of links to select. The aim of similarity filtering is to select a subset of record pairs $(r_i, r_j) \in \mathbf{E}$ into a similarity graph $\mathbf{G}_f = (\mathbf{V}_f, \mathbf{E}_f)$, with $\mathbf{V}_f \subseteq \mathbf{V}$ and $\mathbf{E}_f \subset \mathbf{E}$, based on manual classification of up to β_t record pairs in $\mathbf{E}$, such that the number of matches in $\mathbf{E}_f$ is maximised while $|\mathbf{E}_f| = o_m \cdot \epsilon$.*

Our similarity filtering approach is based on the assumption that record pairs that have a higher similarity are generally more likely to be true matches. While

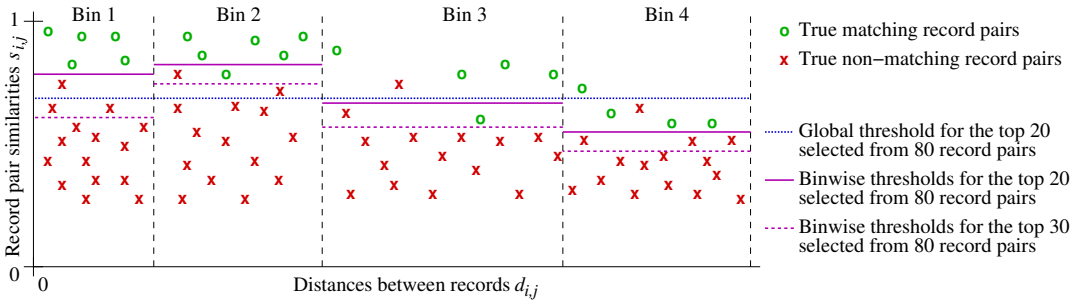

Figure 6.2: Filtering of record pairs (links) with the highest similarities. Compared to using all 80 links, with $o_m = 20$ and $\epsilon = 1$, the filtered similarity graph contains a much smaller number of true non-matches at the cost of losing only few true matches. If the top 20 links are chosen globally (no binning), then the recall of the filtered graph is only 0.8 (4 out of 20 true matches are missed), whereas when links are chosen locally using bin specific thresholds, then recall would be 0.9 (only 2 true matches are removed by the filtering process). If we set $\epsilon = 1.5$ and select 30 record pairs then recall will be 1 for the binned approach.

this assumption does not necessarily hold for every record pair in **G**, it is a common assumption used in RL [5, 164]. We also assume that the distances, $d_{i,j}$, of record pairs affect the values in their corresponding similarity vectors, $\mathbf{s}_{i,j}$, as is illustrated in Figure 6.2. For example, the further people move the more details in their addresses will likely change. While a local move will result in a changed street address only, a move further away can also lead to changed city, zip-code, and even state values. As we discuss next, we employ a binning-based active learning approach to identify different similarity thresholds for filtering on different subsets (bins) of record pairs in **E** using the distances $d_{i,j}$ of record pairs.

6.2.2 Binning-based Filtering

Algorithm 6, which outlines our filtering technique, takes a pairwise similarity graph **G**, the total manual classification budget for an oracle (domain expert) β_t, the minimum budget per bin β_m ($\beta_m \geq 1$), the expected number of true matches as identified by a domain expert o_m, a multiplier for the number of record pairs to select ϵ, and the binning method (either equal width or equal depth) γ as input.

In lines 1 and 2 of Algorithm 6 the main data structures, a list of bins **B**, and a queue **Q**, are initialised to store the final bins, and the bins that need to be further processed respectively. Next, in line 3, we generate the first bin $\mathbf{b}_1$ that contains the full similarity graph **G**, and set the level of this bin to $\mathbf{b}_1.l = 1$. The budget β_1, of how many record pairs are manually classified (labelled) by the human oracle (domain expert) in this first bin is calculated with the **CalcBudget**() function. Due to the recursive process of splitting a bin into two in each iteration, we allocate a labelling budget that depends on the level of a bin. With a total budget of β_t, for a bin at level l we allocate a budget of $\beta_l = \beta_t/(2^{2l-1})$, such that a budget of $\beta_t/2^l$ is

Algorithm 6: Binning-based similarity graph filtering using active learning

Input: $\mathbf{G}$ - Undirected pairwise similarity graph
β_t - Total budget (maximum number of manual record pair classifications)
β_m - Minimum number of manual classifications a bin must contain
o_m - Expected number of true matches
ϵ - Multiplier for number of record pairs (links) to select
γ - Binning method (either equal width or equal depth)
Output: $\mathbf{G}_f$ - Filtered pairwise similarity graph containing $o_m \cdot \epsilon$ selected links

1 $\mathbf{B} = [\,]$ // Initialise an empty list to store the final bins
2 $\mathbf{Q} = [\,]$ // Initialise a queue to hold bins to be processed further
3 $\mathbf{b}_1 = \textbf{InitBin}(\mathbf{G})$; $\mathbf{b}_1.l = 1$ // Initialise first bin and set bin level to 1
4 $\beta_1 = \textbf{CalcBudget}(\beta_t, 1)$ // Get budget for the first bin
5 $\mathbf{b}_1.\delta = \textbf{GetTopPairsThresh}(\mathbf{b}_1, o_m \cdot \epsilon)$ // Get threshold for the top $o_m \cdot \epsilon$ links
6 $\mathbf{b}_1.\mathbf{l} = \textbf{GetOracleLabels}(\mathbf{b}_1, \beta_1, \gamma)$ // Manual Classification of β_1 links
7 $\mathbf{b}_1.s = \textbf{CalcScore}(\mathbf{b}_1)$ // Calculate the score for bin $\mathbf{b}_1$
8 $\mathbf{Q}.add(\mathbf{b}_1)$ // Add the first bin to the queue
9 **while** $(\mathbf{Q} \neq [])$ **do** // Process queue sorted by bin scores
10 $\quad \mathbf{b}^p = \mathbf{Q}.pop()$ // Get the next (parent) bin to process based on its score
11 $\quad \mathbf{b}^l, \mathbf{b}^r = \textbf{SplitBin}(\mathbf{b}^p, \gamma)$ // Split parent bin based on binning method γ
12 $\quad \beta_c = \textbf{CalcBudget}(\beta_t, \mathbf{b}^p.l + 1)$ // Get the budget for the two child bins
13 $\quad$ **if** $(|\mathbf{b}^l.\mathbf{l}| + \beta_c) \geq \beta_m \wedge (|\mathbf{b}^r.\mathbf{l}| + \beta_c) \geq \beta_m$ **then**
14 $\quad\quad \mathbf{b}^l.\mathbf{l} = \mathbf{b}^l.\mathbf{l} \cup \textbf{GetOracleLabels}(\mathbf{b}^l, \beta_c, \gamma)$
15 $\quad\quad \mathbf{b}^r.\mathbf{l} = \mathbf{b}^r.\mathbf{l} \cup \textbf{GetOracleLabels}(\mathbf{b}^r, \beta_c, \gamma)$
16 $\quad\quad \delta^l, \delta^r = \textbf{CalcBestThresh}(\mathbf{b}^l, \mathbf{b}^r)$ // Get optimal bin thresholds
17 $\quad\quad$ **if** $(\delta^l == \mathbf{b}^p.\delta) \wedge (\delta^r == \mathbf{b}^p.\delta)$ **then** // Same threshold as for parent bin
18 $\quad\quad\quad \mathbf{B}.add(\mathbf{b}^p)$; **go to line 9** // Add parent bin to final bin list
19 $\quad\quad \mathbf{b}^l.\delta = \delta^l$; $\mathbf{b}^l.s = \textbf{CalcScore}(\mathbf{b}^l)$; $\mathbf{Q}.add(\mathbf{b}^l)$ // Add child bins to queue
20 $\quad\quad \mathbf{b}^r.\delta = \delta^r$; $\mathbf{b}^r.s = \textbf{CalcScore}(\mathbf{b}^r)$; $\mathbf{Q}.add(\mathbf{b}^r)$
21 $\quad$ **else if** $(|\mathbf{b}^l.\mathbf{l}| + \beta_c) \geq \beta_m)$ **then** // Only the left child bin has enough labels
22 $\quad\quad \mathbf{b}^l.\mathbf{l} = \mathbf{b}^l.\mathbf{l} \cup \textbf{GetOracleLabels}(\mathbf{b}^l, \beta_c, \gamma)$; $\mathbf{b}^l.s = \textbf{CalcScore}(\mathbf{b}^l)$; $\mathbf{Q}.add(\mathbf{b}^l)$
23 $\quad\quad \mathbf{B}.add(\mathbf{b}^r)$ // Add right child bin to final bin list
24 $\quad$ **else if** $(|\mathbf{b}^r.\mathbf{l}| + \beta_c) \geq \beta_m)$ **then** // Only the right child bin has enough labels
25 $\quad\quad \mathbf{b}^r.\mathbf{l} = \mathbf{b}^r.\mathbf{l} \cup \textbf{GetOracleLabels}(\mathbf{b}^r, \beta_c, \gamma)$; $\mathbf{b}^r.s = \textbf{CalcScore}(\mathbf{b}^r)$; $\mathbf{Q}.add(\mathbf{b}^r)$
26 $\quad\quad \mathbf{B}.add(\mathbf{b}^l)$ // Add left child bin to final bin list
27 $\quad$ **else**
28 $\quad\quad \mathbf{B}.add(\mathbf{b}^p)$ // Add parent bin to the final bin list
29 $\mathbf{G}_f = (\mathbf{V}_f = \emptyset, \mathbf{E}_f = \emptyset)$ // Initialise empty similarity graph of selected links
30 **for** $\mathbf{b} \in \mathbf{B}$ **do** // Iterate through the bins in the final bin list
31 $\quad \mathbf{G}_f.insert(\textbf{GetLinks}(\mathbf{b}, \mathbf{b}.\delta))$ // Generate the filtered similarity graph
32 **return** $\mathbf{G}_f$ // Return the final filtered pairwise similarity graph

allocated across all bins at level l. For example, with $\beta_t = 1{,}000$, we will manually label $\beta_1 = 500$ record pairs in $\mathbf{b}_1$ (with level $l = 1$), $\beta_2 = 125$ in each of the two bins at level $l = 2$, $\beta_3 = 31$ in each of the four bins at level $l = 3$, and so on. Note that the set of manually labelled (classified) record pairs in a bin $\mathbf{b}$, denoted by $\mathbf{b}.\mathbf{l}$,

is propagated from a parent bin to its two child bins in the recursive bin splitting process.

In line 5 of Algorithm 6 we calculate the optimal similarity threshold $\mathbf{b}_1.\delta$ corresponding to the $o_m \cdot \epsilon$ record pairs with the highest similarities in $\mathbf{b}_1$. In line 6 the oracle then manually classifies β_1 record pairs in bin $\mathbf{b}_1$ as $\mathbf{b}_1.\mathbf{l}$ using the **GetOracleLabels**() function. This function conducts labelling such that both the child bins of $\mathbf{b}_1$ inherit labelled record pairs from $\mathbf{b}_1$ based on the binning method γ (which we describe below). The function selects record pairs for labelling that are close to the bin threshold $\mathbf{b}_1.\delta$, with $\beta_1/2$ pairs selected above and $\beta_1/2$ pairs below the threshold. This helps to effectively shift the bin threshold depending upon the manual labels obtained, as we discuss below. We then calculate the score $\mathbf{b}_1.s$ of bin $\mathbf{b}_1$ in line 7, where we describe four score functions in Section 6.2.4, and add bin $\mathbf{b}_1$ to the queue $\mathbf{Q}$ in line 8 for processing. These scores are used to order the queue $\mathbf{Q}$ and determine which bin to process next in the iterative phase of our approach.

We iteratively process bins in $\mathbf{Q}$ starting in line 9 as long as the queue is not empty. In line 10 we select the next (parent) bin, $\mathbf{b}^p$, with the highest score, which we then split into two child bins, $\mathbf{b}^l$ and $\mathbf{b}^r$, using the binning method γ. The function **SplitBin**() performs either equal width or equal depth binning [82] on the parent bin $\mathbf{b}^p$ as specified by γ, using the distances $d_{i,j}$ of each record pair in $\mathbf{b}^p$. **SplitBin**() also increases the level of the child bins as $\mathbf{b}^l.l = \mathbf{b}^p.l + 1$ and $\mathbf{b}^r.l = \mathbf{b}^p.l + 1$, propagates the optimal threshold ($\mathbf{b}^l.\delta = \mathbf{b}^p.\delta$ and $\mathbf{b}^r.\delta = \mathbf{b}^p.\delta$), and splits the set of manual classifications in $\mathbf{b}^p$ according to the binning method such that $\mathbf{b}^l.\mathbf{l} \cup \mathbf{b}^r.\mathbf{l} = \mathbf{b}^p.\mathbf{l}$.

In line 12 we calculate the oracle budget β_c for the child bins based on their level, and in line 13 we check if both child bins will contain enough manual classifications (based on their allocated budgets as well as the labels inherited from their parent). The reason for checking if a bin can have at least β_m labels (where $\beta_m \geq 1$) is to avoid underfitting (where not enough manual labels are available in a bin to calculate an optional similarity threshold). If both bins can have β_m labels, then in line 14 and 15 we obtain new manual classifications ($\mathbf{b}^l.\mathbf{l}$ and $\mathbf{b}^r.\mathbf{l}$) for them, and in line 16 we calculate the new optimal similarity thresholds for the child bins using the function **CalcBestThres**(), as we describe in Sect 6.2.3. If it turns out that the optimal threshold of the parent, $\mathbf{b}^p.\delta$ cannot be improved (in line 17) because the distribution of the similarities of links in both child bins is homogeneous (highly similar), then we add the parent bin $\mathbf{b}^p$ to the final list of bins $\mathbf{B}$ in line 18, and go back to line 9 to process the next bin in $\mathbf{Q}$.

Otherwise, in lines 19 and 20, for each child bin $\mathbf{b}^l$ and $\mathbf{b}^r$, the threshold is set to its calculated optimal value, its bin score is calculated, and then both child bins are added to the queue $\mathbf{Q}$. On the other hand, if only one of the two child bins can have at least β_m labels, in lines 21 to 26 we obtain manual classifications for that bin, update the remaining budget and the score of that bin, and add it to $\mathbf{Q}$, while the other child bin (the one not having enough labels) is added to the final list of bins $\mathbf{B}$. If neither child bin can have at least β_m labels then in line 28 we add the parent bin $\mathbf{b}^p$ to the final list of bins $\mathbf{B}$, because having bins with very few labels (less than β_m) can result in overfitting.

Algorithm 7: Calculate optimal bin similarity thresholds - **CalcBestThres**()

Input: $\mathbf{b}^l, \mathbf{b}^r$ - Left and right child bins
Output: δ^l, δ^r - Optimal bin threshold pair

```
1  fn_l = GetFalseNeg(b^l, b^l.δ); fn_r = GetFalseNeg(b^r, b^r.δ)  // Get false negatives
2  f_t = |fn_l| + |fn_r|                  // Get the initial total false negative count
3  L = [(f_t, b^l.δ, b^r.δ)]  // List with bin thresholds and total false negative count
4  for fn ∈ fn_l ⊕ fn_r do    // Iterate through list of false negative record pairs
5      δ^l, δ^r = ShiftThresh(b^l, b^r, fn)          // Shift thresholds in child bins
6      fn_l = GetFalseNeg(b^l, δ^l); fn_r = GetFalseNeg(b^r, δ^r)  // New false negatives
7      if (|fn_l| > f_t) ∨ (|fn_r| > f_t) then  // One bin exceeds the false negative total
8          break                    // Stop shifting threshold in a given direction
9      else
10         L.add((|fn_l| + |fn_r|, δ^l, δ^r))    // Add thresholds and false negative count
11 δ^l, δ^r = GetMinFalseNegThres(L)                      // Get optimal thresholds
12 return δ^l, δ^r                                  // Return the optimal thresholds
```

Subsequent to processing all bins in $\mathbf{Q}$, we generate the filtered similarity graph of selected links (record pairs), $\mathbf{G}_f$, in lines 29 to 31 by looping over all bins in $\mathbf{b} \in \mathbf{B}$, and adding all record pairs with a pairwise similarity of at least the bin threshold $\mathbf{b}.\delta$ into the graph $\mathbf{G}_f$. Finally, in line 32 the filtered pairwise similarity graph $\mathbf{G}_f$ is returned.

6.2.3 Calculating Optimal Bin Similarity Thresholds

We now describe the functionality of the **CalcBestThresh**() function (used in line 16 in Algorithm 6), as outlined in Algorithm 7. The input to Algorithm 7 is a bin pair $\mathbf{b}^l$ and $\mathbf{b}^r$, and the function calculates a pair of optimal thresholds, δ^l and δ^r, which minimise the total number of false negatives across both bins. The algorithm starts with obtaining the lists of false negatives, $\mathbf{fn}_l$ and $\mathbf{fn}_r$, in the two bins, where true matching record pairs (as manually classified by the oracle) that have a similarity below the thresholds $\mathbf{b}^l.\delta$ and $\mathbf{b}^r.\delta$ are considered as false negatives. We assume that record pairs in a bin are sorted based on their similarities. In lines 2 and 3, we then calculate the initial total number of false negatives, f_t, and initialise a list $\mathbf{L}$ with a tuple made of f_t and the initial thresholds.

The loop starting in line 4 (with $\oplus$ representing list concatenation) then shifts thresholds for each false negative record pair fn in both child bins, where the function **ShiftThresh**() sets the threshold of one of the bins ($\mathbf{b}^l$ or $\mathbf{b}^r$) to the similarity value of fn. The threshold of the other child bin is adjusted such that the total number of record pairs with a similarity greater than the thresholds is unchanged. This ensures that we select $o_m \cdot \epsilon$ links at any time, despite the changing thresholds. The new thresholds δ^l and δ^r are returned by **ShiftThresh**(), and we then obtain the lists of false negatives $\mathbf{fn}_l$ and $\mathbf{fn}_r$ for δ^l and δ^r. In lines 7 and 8 we check if at least one of the bins has more false negatives than the original total false negative count, f_t. If

this is the case we end further shifting of thresholds because no more improvement can be gained (a threshold combination that results in one of the bins having more false negatives compared to the original cannot be improved). If the condition in line 7 is not met, in lines 9 and 10 we add the new total false negative count $|\mathbf{fn}_l| + |\mathbf{fn}_r|$ together with the new threshold pair δ^l and δ^r to the list $\mathbf{L}$. In line 11, we finally obtain the optimal bin threshold pair δ^l and δ^r that has a minimum total false negative count, and in line 12 we return this threshold pair.

Complexity analysis: We now conduct a complexity analysis of Algorithms 6 and 7. In Algorithm 6, the steps in lines 3 to 8 have $O(|\mathbf{E}|)$ time complexity since the initial bin contains all edges in the similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. The while loop starting at line 9 can iterate a maximum of $2 \cdot \beta_t - 1$ times in the worst case considering $\beta_m = 1$, since that would result in a number of $2 \cdot \beta_t - 1$ bins with one labelled record pair in each. All steps, except the **CalcBestThresh**() function within the while loop can be executed in constant or $O(|\mathbf{E}|)$ time.

Considering Algorithm 7 which outlines the **CalcBestThresh**() function, the steps in lines 1 to 3 have constant time complexity $O(1)$ considering set representation of manually classified record pairs. The loop starting at line 4 in Algorithm 7 iterates a maximum of β_t times, since only labelled false negative (fn) record pairs are considered. All steps within this loop can be executed in constant time whereas the step in line 11 has $O(\beta_t)$ time complexity given the list $\mathbf{L}$ can contain a maximum of β_t tuples. The time complexity of the filtered graph $\mathbf{G}_f$ generation step (lines 29 to 31) in Algorithm 6 is $O(\beta_t)$ assuming the worst case where $\beta_m = 1$. Therefore, the total time complexity of our filtering algorithm is $O(\beta_t^2 + \beta_t \cdot |\mathbf{E}|)$ as determined by the while loop in Algorithm 6 and the function **CalcBestThresh**() outlined in Algorithm 7.

6.2.4 Bin Scoring Functions

An important aspect of our recursive binning approach is the ordering of the queue $\mathbf{Q}$ based on the bin scores, $\mathbf{b}.s$, which determine how bins are being processed. Our aim is to calculate an optimal threshold for each bin such that the total number of false negatives is minimised before the budget is used up. We now describe four variations of the function **CalcScore**(). In all variations we only consider the record pairs manually classified by the oracle in a given bin, $\mathbf{b}.\mathbf{l}$.

1. **False negative count** ($score_{fn}$): With this approach we calculate the number of false negative record pairs contained in a bin $\mathbf{b}$, where a false negative is a pair that has been classified as a true match by the oracle and that has a similarity below the bin threshold $\mathbf{b}.\delta$. Using this scoring function means bins that contain more false negatives will be at the top of the queue $\mathbf{Q}$, and processed first.

2. **Bin recall** ($score_r$): With this approach we calculate the recall of bin $\mathbf{b}$ as the proportion of manually classified true matches with a similarity above $\mathbf{b}.\delta$ over all manually classified true matches in $\mathbf{b}$. With this scoring function we process bins in $\mathbf{Q}$ such that those bins with lowest recall are processed first. This

allows us to further explore bins that have fewer true positives and adjust their thresholds to improve their recall.

3. **Normalised false negative count** ($score_{nfn}$): This approach is similar to the $score_{fn}$ approach, except that we divide the false negative count by the bin size $|\mathbf{b}|$, to find the bins with the largest proportion of false negatives.

4. **Adjusted bin recall** ($score_{ar}$): This approach is similar to the $score_r$ function except that we adjust the original $score_r$ value by dividing it by the bin size $|\mathbf{b}|$. With this approach, larger bins that have a lower bin recall value will be processed first.

6.3 Experimental Evaluation

In this section, we present the results of the experimental evaluation we conducted to assess the linkage quality achieved with applying our proposed graph filtering technique. For evaluation, we used the real-world Isle of Skye (IoS) birth data set, the synthetic UK birth data set, and the real-world North Carolina Voter Registration data set (NCVR) which we presented in Section 2.5. For the birth data sets, we used the pairwise similarity graphs generated by comparing record pairs using *All* attribute values $\mathbf{G}_A$ as shown in Table 2.4 on page 30. This is because considering more attributes in the comparison results in more widely distributed pairwise similarities, which helps in effectively selecting record pairs with a higher overall similarity within each bin using our approach. We implemented our algorithms in Python 2.7, and all experiments were conducted on a server running Ubuntu 18.04 with 64-bit Intel Xeon 2.10 GHz CPUs and 512 GB of memory.

The pairwise similarity graphs $\mathbf{G}_A$ corresponding to the IoS and UK birth data sets contain approximately 5.4 million and 4.3 million record pairs (edges) respectively, as shown in Table 2.4, whereas the similarity graph corresponding to the NCVR data set contains approximately 34.6 million record pairs as shown in Table 2.6. By applying our graph filtering technique, we can significantly reduce the sizes of these graphs by retaining only o_m number of record pairs (the expected number of true matches), which is 17,613, 14,027, and approximately 7 million for the IoS, UK, and NCVR data sets, respectively, as shown in Table 2.1 on page 25. For evaluation we used the precision and recall measures which we discussed in Section 2.4.

For the data dimensions used for binning, we calculated time distances as the number of days between two birth records in IoS and UK, and the number of months between two voter records in NCVR, while we calculated geographical (space) distances using address geocoding [102] for IoS and UK, and the distances between zip-codes using a database of zip-code locations for NCVR, respectively. As illustrated in Figure 6.2, as baseline we explore a simple filtering approach using a global threshold for selecting the o_m record pairs with the highest similarity, assuming o_m was provided by a domain expert. We then investigate our proposed method of binning record pairs to help improve the quality of the filtered similarity graph.

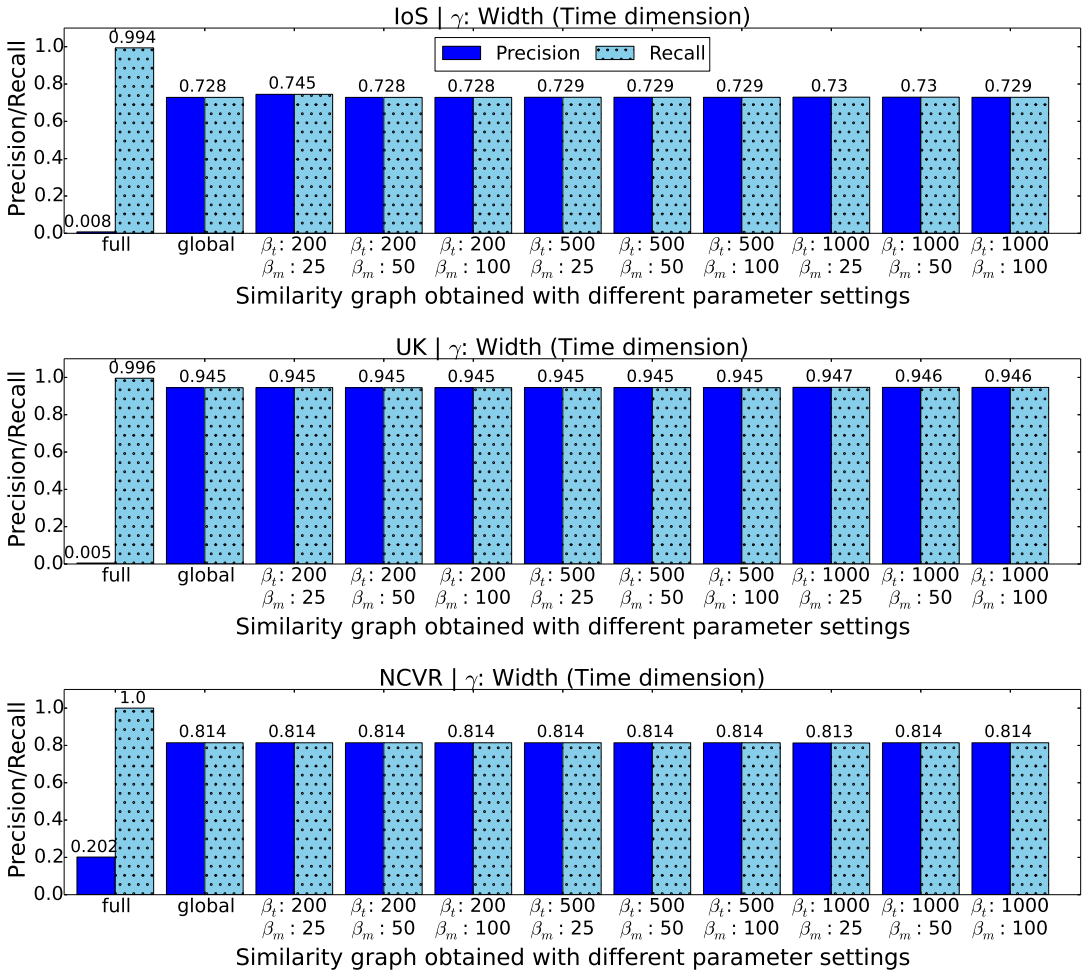

Figure 6.3: Graph filtering with equal width binning based on the time data dimension: Precision and recall results of the full similarity graph compared with the quality of the graphs filtered with a global threshold (top o_m links), or binwise thresholds for different total budgets β_t and manual classifications per bin β_m. Precision and recall results are the same for all the filtered graphs since the number of record pairs retained in the graph is equal to the number of true matches.

We explored the parameter settings $\beta_t = [200, 500, 1000]$ for total manual classification budget, and $\beta_m = [25, 50, 100]$ for the threshold number of manual classifications per bin. With a set of initial experiments we conducted, we found that retaining exactly o_m record pairs in the filtering by setting $\epsilon = 1$ produced better linkage results, and therefore present results obtained for $\epsilon = 1$. Furthermore, we explored the different score calculation methods (**CalcScore**()) described under Section 6.2.4.

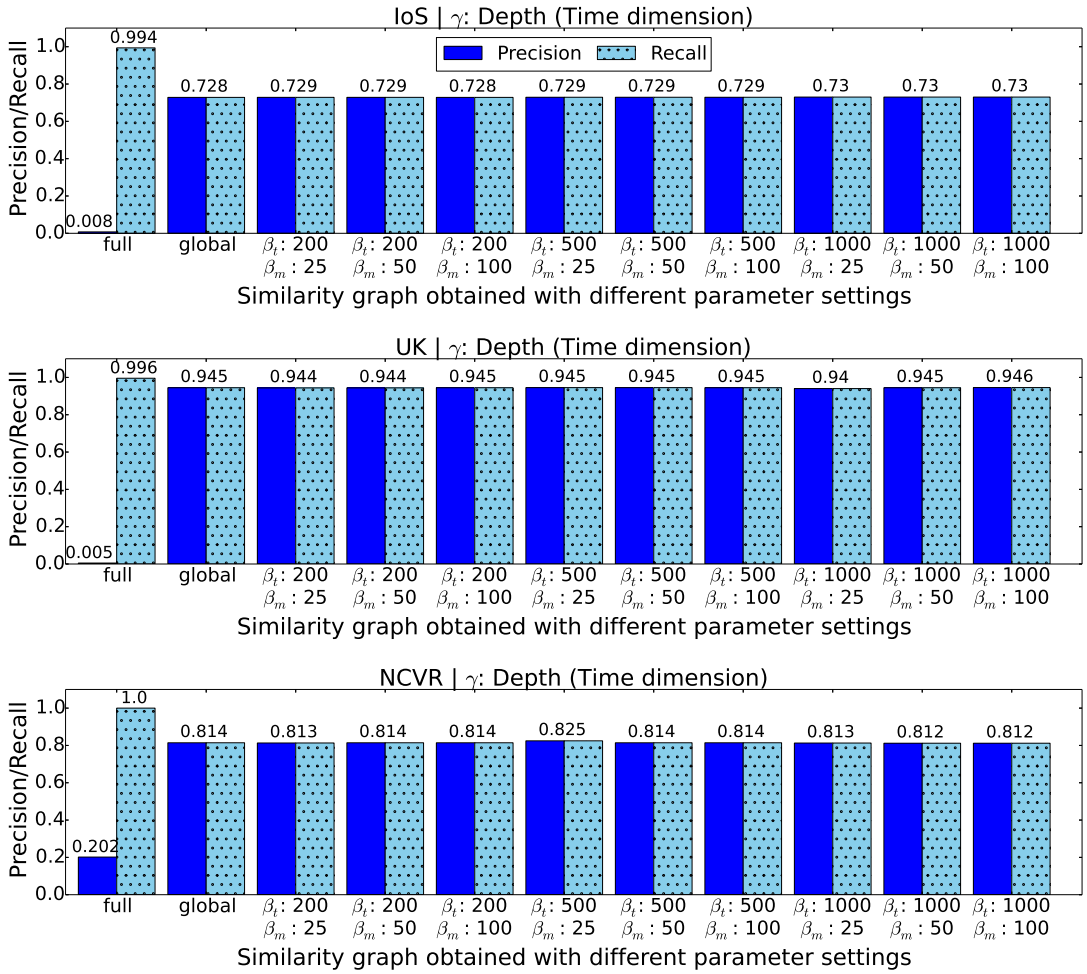

Figure 6.4: Graph filtering with equal depth binning based on the time data dimension: Precision and recall results of the full similarity graph compared with the quality of the graphs filtered with a global threshold (top o_m links), or binwise thresholds for different total budgets β_t and manual classifications per bin β_m.

6.3.1 Filtered Similarity Graph Quality

In Figures 6.3 to 6.6 we show the precision and recall results obtained for the original full similarity graph, compared with the quality of the filtered graphs obtained with applying a global threshold, as well as our binning method using different total budgets β_t and different manual classification thresholds per bin β_m. The filtered results were averaged across the score calculation methods described in Section 6.2.4 since they produced similar graph quality, with the adjusted bin recall method ($score_{ar}$) producing slightly better results. For all filtered graphs, both precision and recall values are the same because we limit the number of record pairs in the filtered graph to o_m, which results in the number of classified record pairs ($TP + FP$ in Equation 2.2)

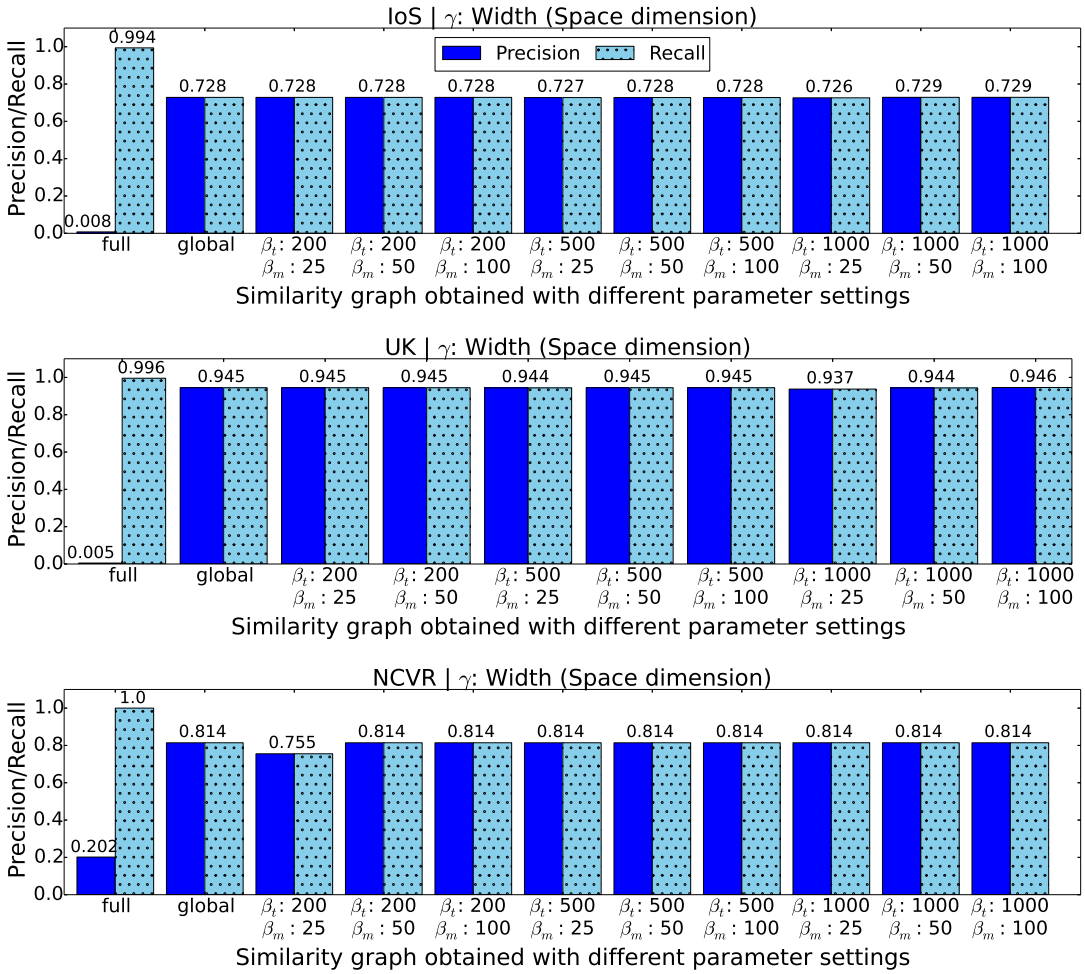

Figure 6.5: Graph filtering with equal width binning based on the space data dimension: Precision and recall results of the full similarity graph compared with the quality of the graphs filtered with a global threshold (top o_m links), or binwise thresholds for different total budgets β_t and manual classifications per bin β_m.

being the same as the number of true matches ($TP + FN$ in Equation 2.3). We can see that the precision of the filtered graphs are far better compared to the original graph (full), since our filtering approach was able to remove a large proportion of the true non-matching record pairs.

In Figures 6.3 and 6.4, we show the quality of the filtered graphs obtained with applying binning on the time data dimension, using equal width and equal depth binning respectively. For all three data sets, the graph quality has slightly improved compared to the baseline (global) at least for one parameter setting, except for NCVR with equal width binning on the time data dimension. Our binning approach has worked for the IoS and UK data sets on the time dimension, since for birth record pairs we find patterns such as no true matches between zero to nine months (an

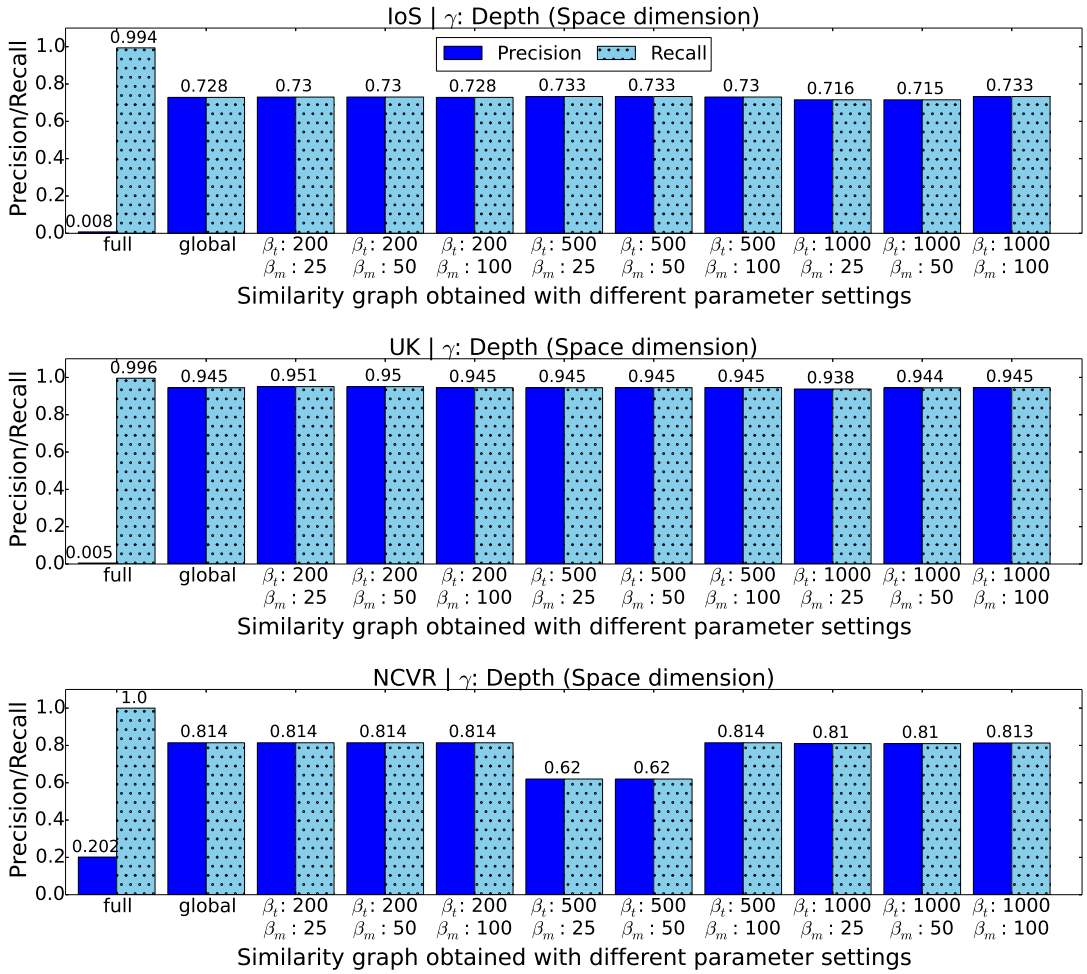

Figure 6.6: Graph filtering with equal depth binning based on the space data dimension: Precision and recall results of the full similarity graph compared with the quality of the graphs filtered with a global threshold (top o_m links), or binwise thresholds for different total budgets β_t and manual classifications per bin β_m.

impossible age gap for siblings). However, with NCVR, there are less distinct differences in the bins generated on the time dimension, whereas equal width binning has failed to capture the limited time related patterns available in this data set.

Figures 6.5 and 6.6 show the quality of the filtered graphs obtained with applying binning on the space data dimension, using equal width and equal depth binning respectively. While some graph quality improvements have been obtained for the IoS and UK data sets on the space data dimension with filtering compared to the baseline, considerable reductions in precision and recall can be seen for certain parameter configurations with the NCVR data set. In the historic IoS and UK birth data set, we see siblings being born in nearby geographic locations (often the same household). Therefore, true matches often have smaller distances and true non-matches larger

distances in the space dimension for the birth data sets. This argument often does not hold true for the more modern NCVR data set, where we see the same individual (corresponding to true matches) relocating in different counties. Because of this reason we get much worse graph quality for NCVR with filtering on the space dimension for certain parameter configurations, compared to the baseline.

6.3.2 Linkage Quality and Efficiency Improvement

In this section, we apply the graph clustering techniques (greedy, star, and robust graph clustering) which we proposed in Chapter 4 on the filtered graphs, and compare the quality of the resulting clusters with the linkage quality obtained with clustering the full similarity graph. We consider the best clustering parameters for the IoS and UK birth data sets as specified in Table 4.1, and the same parameters as for IoS on the experiments conducted with NCVR given both are real-world data sets. Since the aim of these experiments is to show the inefficiency of applying RL classification on large graphs, we used low similarity threshold values δ_s for the clustering. We set $\delta_s = 0.55$ for the IoS and UK birth data sets and $\delta_s = 0.45$ for the NCVR data set such that the majority of true matching record pairs were retained for clustering. We have only applied the greedy and star clustering approaches on the large NCVR data set because RL could not be executed in feasible time (within a month) by applying the robust graph clustering approach on the full NCVR similarity graph. Furthermore, for each data set we have considered the best filtered graph with binning on the time dimension (since that produced better results compared to binning in the space dimension) for a given total budget β_t, and binning method γ (either equal depth or equal width).

Figures 6.7 to 6.9 show the clustering quality achieved with the full and filtered pairwise similarity graphs. As shown in Figure 6.7, both the precision and recall significantly improve when clustering is applied on the IoS and UK filtered similarity graphs, whereas the precision considerably improves at the cost of a slight decline in recall for the NCVR filtered similarity graphs, compared to clustering with the full graph. For the IoS data set, the best clustering quality was achieved with β_t = 1,000 and γ set to equal width which has produced better quality compared to using a global threshold (the baseline). While the same configurations have resulted in an improvement in recall for the UK data set, the precision has declined compared to the baseline. For the NCVR data set the linkage quality results achieved with clustering the filtered graphs are similar as well as for the global threshold technique.

Figure 6.8 shows the linkage quality results obtained for the star clustering approach. Compared to applying star clustering on the full similarity graph, clustering the filtered graphs has improved both the precision and recall for the UK data set, whereas the precision has considerably improved at a slight decline in recall for the IoS data set. However, for the NCVR data set, the precision improvement has been slightly outweighed by the decline in recall. Applying greedy and star clustering on the filtered NCVR similarity graphs have therefore not yielded good results which is due to the quality of the graphs not being improved for the NCVR data set with the

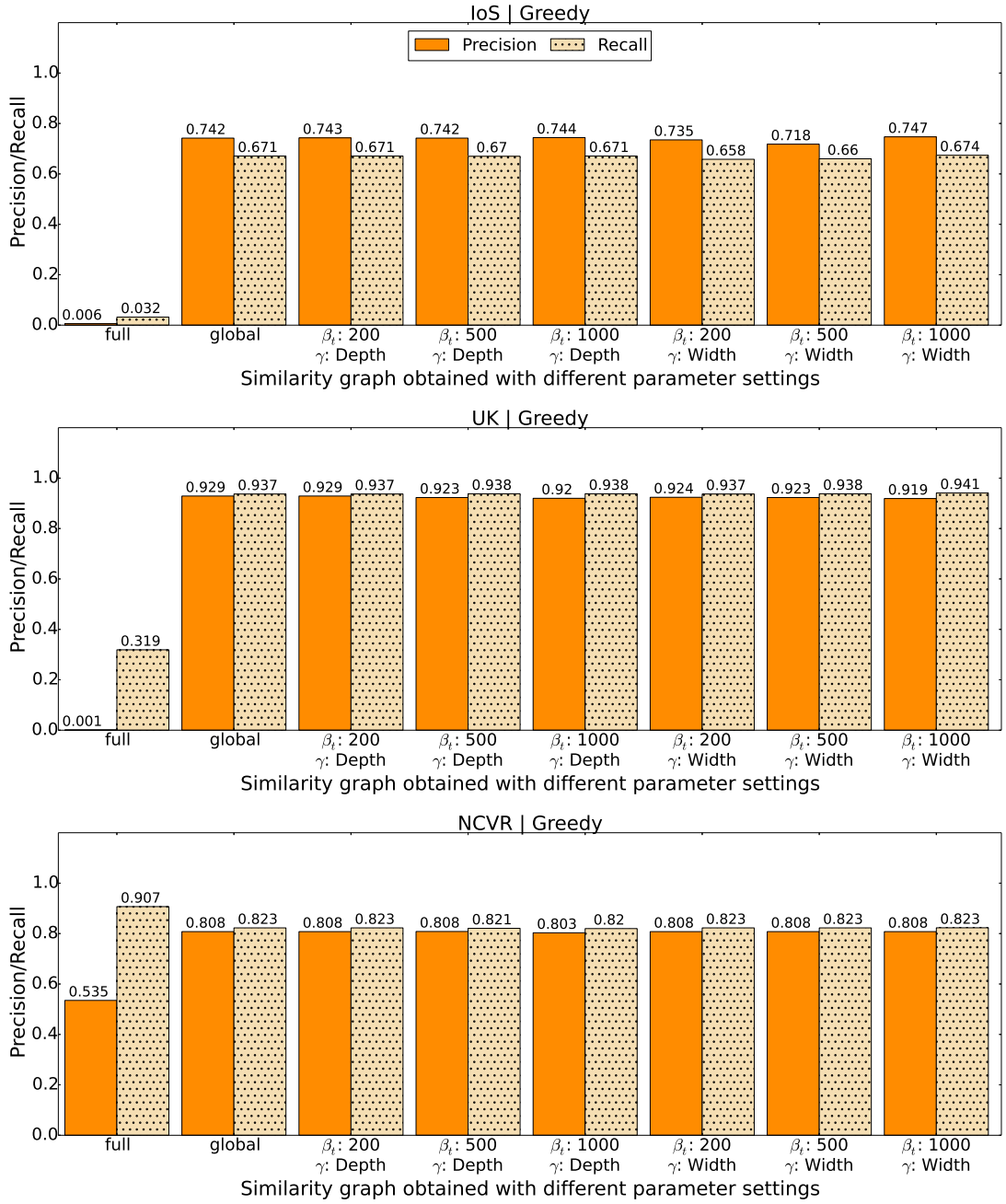

Figure 6.7: Precision and recall results obtained by applying greedy clustering on the full similarity graph, the graph filtered with a global threshold (top o_m links), or graphs filtered with equal depth and equal width binning using different total budgets β_t.

application of our binning approach, as we discussed in the previous section. The parameter configurations β_t = 200 and β_t = 1,000 using equal width binning have

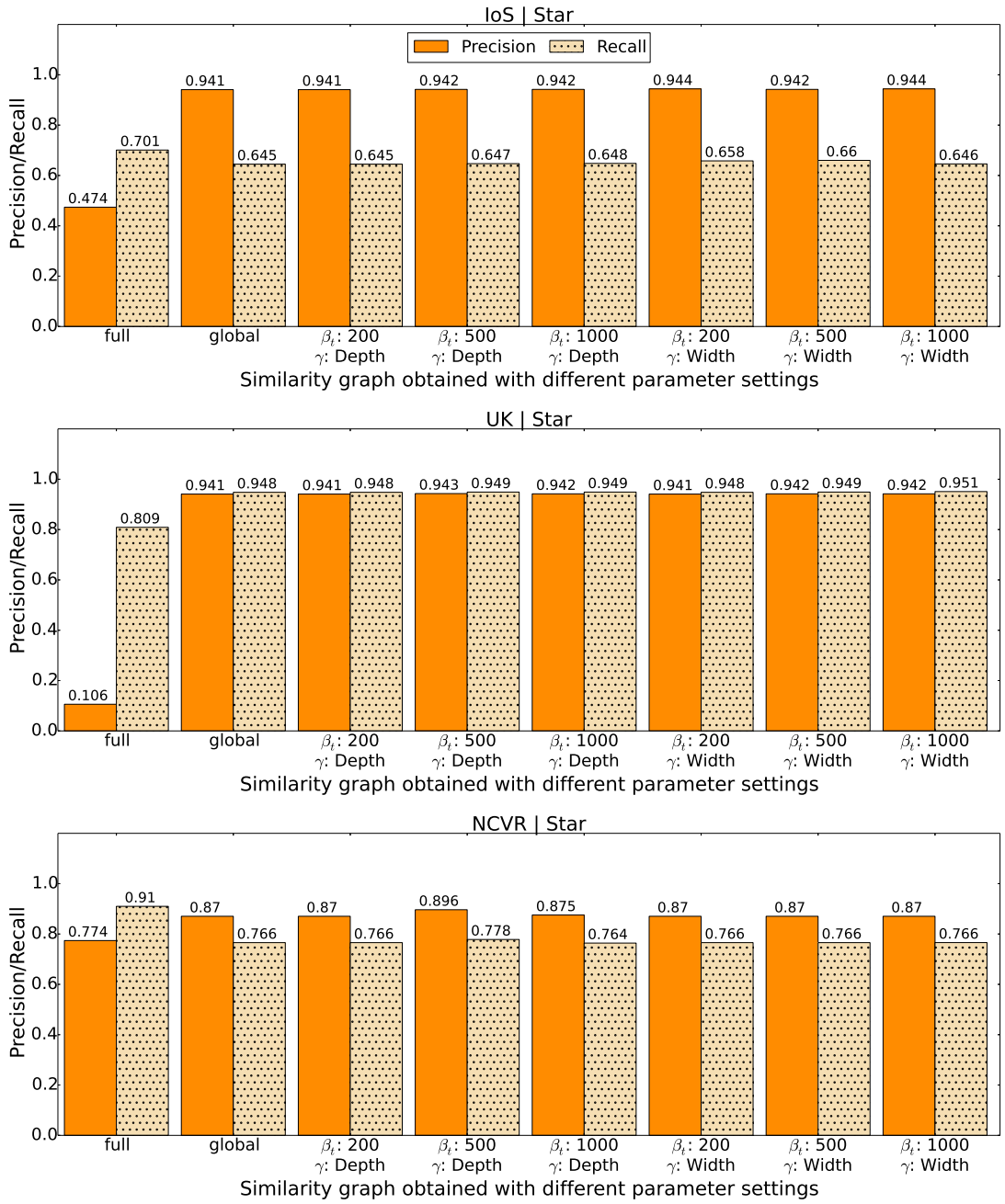

Figure 6.8: Precision and recall results obtained by applying star clustering on the full similarity graph, the graph filtered with a global threshold (top o_m links), or graphs filtered with equal depth and equal width binning using different total budgets β_t.

produced the best star clustering results for the IoS and UK data sets respectively, both of which exceed the corresponding baseline results achieved with the global threshold method.

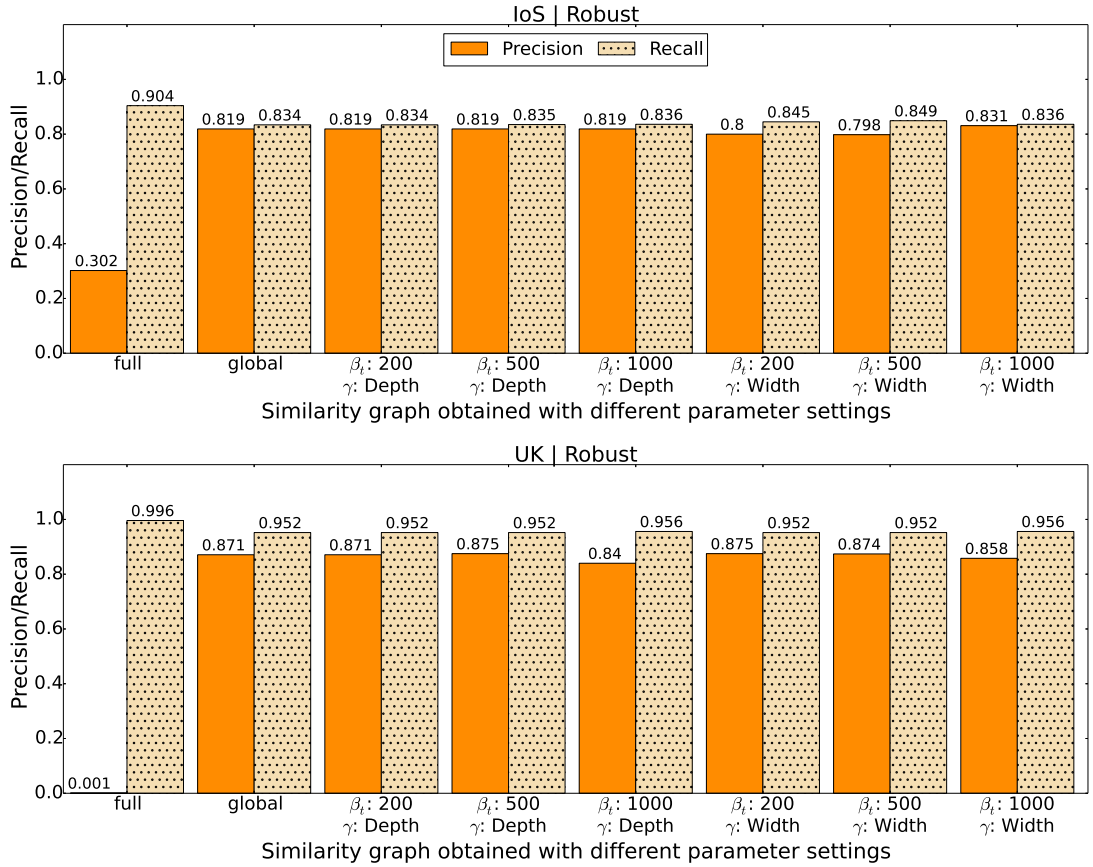

Figure 6.9: Precision and recall results obtained by applying robust graph clustering on the full similarity graph, the graph filtered with a global threshold (top o_m links), or graphs filtered with equal depth and equal width binning using different total budgets β_t.

Figure 6.9 shows the linkage quality results obtained for the robust graph clustering approach. For both the IoS and UK data sets, the precision has considerably improved at the cost of a slight decline in recall when clustering was applied on the filtered graphs, as compared to clustering the full similarity graph. The best linkage results were achieved with the parameter configurations β_t = 1000 using equal width binning for the IoS data set, whereas both β_t = 500 using equal depth binning, and β_t = 200 using equal width binning produced the best linkage results for the UK data set, which are better than the corresponding baseline (global) results. These clustering experiments further show that our active learning-based graph filtering approach can improve the linkage results when clear patterns along the binning dimensions are available, such as for the IoS and UK data sets. Furthermore, these results show that applying RL clustering on filtered graphs produces far better linkage quality compared to clustering the full similarity graphs.

Table 6.1: The run-times (in seconds) of graph clustering using the full similarity graph, graph filtered with a global threshold (top o_m links), or graphs filtered with equal depth (EDB) and equal width (EWB) binning using different total budgets β_t.

Graph	Greedy			Star			Robust	
	IoS	UK	NCVR	IoS	UK	NCVR	IoS	UK
Full	108.31	254,134.09	1,315.90	178.93	3,293.42	398.05	7,097.54	4,725.97
Global	0.66	0.48	286.22	1.12	0.41	247.35	1,920.05	126.31
β_t: 200, γ: EDB	0.62	0.48	286.24	1.11	0.40	257.55	1,889.20	124.47
β_t: 500, γ: EDB	0.65	0.48	288.90	1.07	1.06	251.41	1,835.67	125.25
β_t: 1000, γ: EDB	0.69	0.47	278.12	1.12	0.40	253.31	1,849.90	124.79
β_t: 200, γ: EWB	0.61	0.49	285.96	1.07	0.40	254.73	1,864.82	125.94
β_t: 500, γ: EWB	0.69	0.52	287.22	1.14	0.40	253.85	1,881.64	126.39
β_t: 1000, γ: EWB	0.64	0.44	285.95	1.13	0.39	242.94	1,848.45	127.94

As shown in Figures 6.7 to 6.9, we cannot see a significant variation in linkage quality with regard to the different parameter settings that we have explored for filtering. This indicates that our proposed filtering method is quite robust to parameter variations, and that regardless of the choice of parameter settings, the final linkage precision significantly improves at the cost of minor reduction in recall when a filtered graph is clustered, compared to applying clustering on the full graph.

Table 6.1 shows the run-times obtained with applying graph clustering on the full and filtered pairwise similarity graphs. For every data set and clustering algorithm, the RL classification time has been reduced an entire order of magnitude when clustering was applied on the filtered graphs compared to clustering the full similarity graphs. Therefore, applying graph filtering improves both the quality and the efficiency of RL. The run-times reported in Table 6.1 do not include the overhead of the filtering technique proposed in this chapter. The overhead of filtering for the IoS and UK data sets is approximately 250 seconds, and 2,500 seconds for the NCVR data set. Note that even though this overhead seems significant, given our filtering approach requires to be executed only once, and numerous computationally expensive clustering techniques can be applied on the filtered graphs multiple times, allocating time on our graph filtering step is justifiable and beneficial for the overall RL process.

6.4 Summary

In this chapter we have presented a novel active learning-based graph clustering technique which utilises a domain expert's knowledge of the approximate number of true matches o_m, and the distribution of pairwise similarities across a suitable data dimension (such as time or space) to effectively filter likely non-matches from a large pairwise similarity graph corresponding to a data set. In this approach, we apply iterative binning of record pairs (edges in a similarity graph) on a data dimension and retain the top record pairs in each bin (and a total of $o_m \cdot \epsilon$ across the bins, where ϵ is a multiplier for record pair selection) based on the assumption that record pairs with a higher similarity are more likely to be true matches.

We experimentally evaluated our proposed active learning-based graph clustering technique using two real-world and one synthetic data set. We observed significantly improved graph and RL clustering quality compared to the quality of the full graph, when our filtering method was applied on data sets with clear patterns along the chosen data dimensions. The RL efficiency too was improved considerably when clustering was applied on the filtered graphs compared to clustering the full graph. Therefore, applying graph filtering can improve both the linkage quality and efficiency when clear patterns in the data are available along data dimensions. Furthermore, we showed that the linkage quality obtained with filtering graphs using our binning approach was slightly better compared to filtering with a global threshold. Even though this justifies our proposed binning method, applying a global threshold is seemingly adequate when a slight reduction in linkage quality is acceptable. In the next chapter, we present a novel active learning-based method for conducting RL classification while improving the classification efficiency with filtering.

Chapter 7

Active Learning-based Record Linkage With Filtering

As we highlighted in Section 1.2, a lack of availability of ground-truth data in the form of true matching and non-matching record pairs, and the difficulty of scaling most existing algorithms to link large data sets due to quadratic record pair comparisons, are two of the key issues in the Record Linkage (RL) domain. When ground-truth data is unavailable, supervised classifiers cannot be trained to be utilised in RL projects. In this chapter, we propose a novel RL approach which uses an iterative *active learning* strategy which labels a small, selected number of record pairs for training such that supervised classifiers can be effectively used for RL. Furthermore, we incorporate record pair filtering into this active learning approach such that the record pairs with the most confident classifications in previous iterations are disregarded from future classification steps to improve RL efficiency.

In Section 7.1 we provide an introduction to conducting RL with active learning and highlight our contribution to the literature with the methods we propose in this chapter. Next, in Section 7.2 we describe our active learning-based technique for RL filtering and classification in detail. In Section 7.3 we experimentally evaluate our proposed method using four real-world and one synthetic data set, and finally, in Section 7.4 we conclude this chapter with a summary of our findings.

7.1 Introduction

Identifying records that refer to the same entity as conducted in RL requires the comparison of record pairs across the data sets to be linked [136]. However, as we discussed in Section 1.2, the sizes of most real-world databases have rendered such naïve comparison computationally infeasible [56]. Linking two databases containing a thousand records each results in a million record pairs, and with increasing database sizes the number of comparisons grows quadratically. Therefore, as we discussed in Section 2.3, different blocking and indexing techniques have been proposed to reduce the comparison space [138], where record pairs that are highly unlikely to be matches are discarded prior to the expensive comparison step. However, the majority of record pairs retained after having been compared are often non-matches

Table 7.1: The number of matching and non-matching record pairs (edges) in the pairwise similarity graph of each data set, with the approximate match to non-match ratio. For the IoS, Kilm, and UK birth data sets we have considered the pairwise similarity graphs generated by comparing *All* attribute values $\mathbf{G}_A$ (see Section 2.6).

Data set	Matches in graph	Non-matches in graph	Match to non-match ratio
IoS	40,993	5,332,505	1:130
Kilm	67,057	24,978,921	1:373
UK	21,922	4,246,421	1:194
NCVR	6,978,001	27,611,477	1:4
DBLP-ACM	2,220	10,143	1:5
DBLP-Scholar	5,347	23,360	1:4

due to the high class imbalance encountered in linkage applications [35, 138]. As a result, the pairwise similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ generated at the comparison phase (see Definition 2 on page 20) is also generally very large, and mostly contain non-matching record pairs. Table 7.1 shows the match to non-match record pair ratio in the similarity graphs of the data sets we use for evaluation in our thesis, where the number of matches are highly outweighed by the number of non-matches, especially for the birth data sets IoS, Kilm, and UK. Having a pairwise similarity graph $\mathbf{G}$ of considerable size subsequent to the comparison step significantly hinders the performance of classification or clustering methods applied on such similarity graphs as the final step in the RL process [58]. Even though many blocking techniques have been proposed to reduce the number of record pair comparisons, very limited research has explored how to reduce the number of record pairs resulting from the comparison step to enhance the efficiency of the classification step [125].

As we discussed in Section 1.2, another significant issue hindering the use of supervised linkage methods is the lack of ground-truth data for training supervised classification models [42], where ground-truth would consist of a set of record pairs where their true match status is determined by a human with expertise in the relevant domain. However, due to the sizes of real-world databases, the cost and time required for constructing a ground-truth set for even a single linkage application is substantial and often infeasible. As we discussed in Section 3.3, methods such as *active learning* have been developed to select a subset of record pairs for manual classification to be later utilised as a training set for classification tasks [5, 41, 144, 145, 155]. The class imbalance resulting from the number of true matches being highly outnumbered by the number of true non-matches, however, makes the process of selecting a high quality subset of record pairs for manual labelling challenging [145]. It is therefore important to develop techniques to sample a balanced subset of record pairs for labelling that better reflects the characteristics of the full comparison space.

As outlined in Figure 7.1 and described in detail in Section 7.2, we propose a novel active learning-based classification approach with filtering which addresses the two significant issues in RL highlighted above. The challenge of lack of ground-truth data for conducting supervised learning is addressed by an active learning

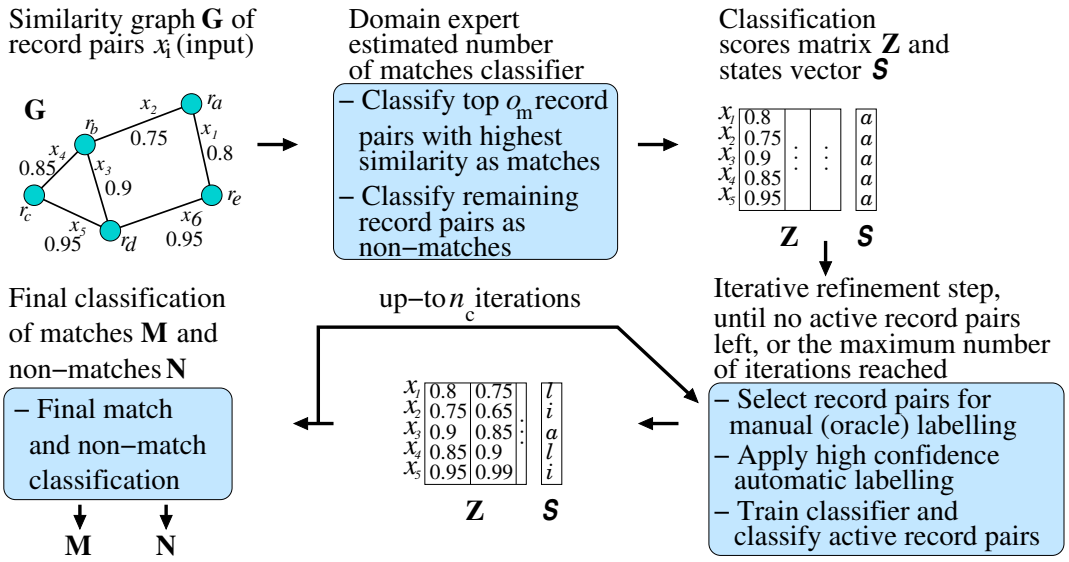

Figure 7.1: Overview of our active learning and filtering approach for efficient RL, as we describe in Section 7.2. Record pairs can be in the state active (a) and still need to be classified, manually labelled by the oracle (state ℓ), or inactive (state i) if they have been automatically labelled as a match or non-match with high confidence, and filtered to improve efficiency.

method that iteratively selects record pairs for manual labelling based on knowledge gained from previous rounds of active learning and automated labelling. Making such informed decisions in the link selection process helps build a training data set that is reflective of the features in the full data set. Furthermore, the selection of record pairs is conducted in a manner that helps deal with the class imbalance problem, by selecting an equal number of likely matches and non-matches.

Since the number of record pairs that can be manually labelled is limited (because this is generally a cumbersome and costly process), we introduce a method to automatically label high confidence match and non-match pairs. This automated labelling approach helps us obtain a larger training data set rather than being constrained by the limited number of manual labels obtained with the active learning approach only. To improve the efficiency of our approach we iteratively utilise this training data set to classify record pairs as matches and non-matches, while concurrently filtering out pairs that are automatically and manually labelled in each iteration. Furthermore, we use traditional machine learning classifiers rather than deep learning approaches due to the generally high time complexity of deep learning algorithms [10].

Even though the concept of combining manually and automatically labelled pairs is not novel [144], their limited application in the RL domain makes our proposed technique a valuable contribution. Furthermore, while we proposed an active learning based filtering technique in Chapter 6, here we develop an active learning strategy that combines filtering and classification steps, as shown in Figure 6.1.

7.2 Active Learning with Filtering

In this section, we describe our approach to active learning with filtering for RL in detail, as illustrated in Figure 7.1. We begin with an overview to provide the main ideas of our approach. In Section 7.2.2 we then describe the first phase of our approach where we obtain an initial classification outcome based on the expected number of matching record pairs. In Section 7.2.3 we discuss how we iteratively refine our classification, and in Section 7.2.4 we provide an overall algorithmic outline of our approach.

7.2.1 Overview

We assume a single (or multiple) data set(s) **D** is to be linked, where no ground-truth data in the form of known true matches and non-matches are available. Supervised learning algorithms can therefore not be applied, requiring either unsupervised clustering techniques [15, 58, 86, 154], or active learning [5, 41, 142, 145], the methodology we use here. Furthermore, we assume that a pairwise similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is generated as described in Definition 2, where the set of vertices, **V**, corresponds to records from a data set ($r_i = v_i$ where $r_i \in \mathbf{D}$ and $v_i \in \mathbf{V}$), and the set of undirected edges, **E**, between vertices represent the attribute value similarities calculated between them ($(r_i, r_j) \in \mathbf{E}$). These similarities can either be a single numerical value $s_{i,j}$ that encapsulates the overall similarity between two records, or they can be a vector of individual similarities $\mathbf{s}_{i,j}$ as calculated between the values of the compared attributes [35] (see Definitions 1 and 2).

In our approach, this similarity graph, **G**, is the main input from where we build a succession of classification models. We consider both selected record pairs for manual classification by a domain expert (named as the *oracle*), as well as an automatic classification of those record pairs where our approach provides a high confidence that the pair is a match or non-match, as we describe below.

Our approach has two phases. In the first we build an initial classifier based on the expected number of matches, o_m, as provided by a domain expert who can often estimate this number based on context related knowledge of the data. For example, when RL is conducted to identify sibling groups, the knowledge about the size distribution of families in a population can be used to estimate o_m [149]. On the other hand, when linking two product databases, where one-to-one matches are expected, o_m is limited by the number of records in the smaller of the two databases being linked.

In the second phase of our approach, we generate a series of increasingly more accurate classification models to iteratively refine the classification outcomes. A similar approach was proposed by Wang et al. [172] for image classification, which we adapt here for linking records. In each iteration we select a certain number of record pairs for manual classification by the oracle, while concurrently we automatically classify pairs as matches or non-matches that consistently have obtained high match or non-match classification scores over the last $k \geq 1$ iterations. This automatic clas-

sification provides both high quality classification results and it also substantially reduces the problem space by lowering the number of record pairs to be considered for classification in the following iterations.

The main data structure in our approach is a matrix $\mathbf{Z}$ that holds the classification scores for each record pair $(r_i, r_j) \in \mathbf{G}.\mathbf{E}$ over all iterations. This matrix has one row per record pair (r_i, r_j), with $n_r = |\mathbf{G}.\mathbf{E}|$ rows in total, and n_c columns, where n_c is the maximum number of iterations (including the first initial classification phase). For ease of presentation, we denote a record pair (r_i, r_j) by its row number x in matrix $\mathbf{Z}$ throughout this chapter. The matrix element $\mathbf{Z}[x, y]$ corresponds to the classification score the record pair x has obtained in iteration y. We denote row x in this matrix with $\mathbf{C}[x, :]$, where $1 \leq x \leq n_r$, and column y with $\mathbf{C}[:, y]$, where $1 \leq y \leq n_c$.

Each record pair can be in one of three states. At the beginning all record pairs are *active* (a). If a record pair was selected for manual classification we set its state to *oracle labelled* (ℓ), while if it was classified automatically we set the pair's state to *inactive* (i). We denote the vector used to store the states of record pairs as $\boldsymbol{\mathcal{S}}$, where $|\boldsymbol{\mathcal{S}}| = n_r$, and the entry for the record pair in row x in the matrix $\mathbf{Z}$ is indicated with $\boldsymbol{\mathcal{S}}[x]$, where $\boldsymbol{\mathcal{S}}[x] \in \{a, \ell, i\}$.

We also assume there is an oracle labelling budget (number of record pairs to be manually classified by the oracle) per iteration, β, and therefore a total budget of $\beta \cdot n_c$, where n_c is the maximum number of iterations. Our approach proceeds as long as there are active record pairs, which means $|\{x : \boldsymbol{\mathcal{S}}[x] = a, 1 \leq x \leq n_r\}| > 0$, and the maximum number of iterations n_c has not been exceeded.

7.2.2 Initial Classification Based on the Expected Number of Matches

As described in Section 7.2.1, the first phase of our approach is to build an initial classifier based on the expected number of true matches, o_m, as provided by a domain expert. Furthermore, our initial classification is based on the assumption that record pairs that have a higher pairwise similarity are generally more likely to be true matches. While this assumption does not necessarily hold for every record pair in graph $\mathbf{G}$, it is a common assumption used in RL [5, 125, 164]. We therefore use the pairwise similarity s_x (assumed to be normalised into $[0, 1]$) of each record pair x in $\mathbf{G}.\mathbf{E}$ as an approximation of the pair's match probability (the probability of a record pair corresponding to the same entity). The initial classifier generates the vector of such match probabilities, $\mathbf{p}$, where we set $\mathbf{p}[x] = s_x$, and assume $0 \leq s_x \leq 1$.

The initial classifier determines the threshold δ_p for classifying record pairs as matches and non-matches based on the estimated true match count o_m, and the assumption that record pairs with higher similarities are more likely to be true matches [5, 125, 164]. In this initial phase, we therefore select the top o_m record pairs with the highest values in $\mathbf{p}$ as matches, and set all others as non-matches.

We also initialise the vector $\boldsymbol{\mathcal{S}}$ to $\boldsymbol{\mathcal{S}}[x] = a$, where $1 \leq x \leq n_r$ to indicate that all record pairs are in the active state a. At the end of this first phase we now have two initial sets of record pairs labelled as matches and non-matches, respectively, which we aim to refine in the second phase.

7.2.3 Iterative Classification Refinement

In the second phase of our approach, we conduct the iterative refinement of the classification using an active learning approach. This iterative phase comprises of five steps as we describe in this section. In the first step we convert the match probabilities into *classification scores* to reflect the class (match or non-match) and the confidence of the classification. In the second step, the classification scores for each active record pair are averaged over $k \geq 1$ iterations, and these averages are then utilised in the third step to decide which record pairs to inactivate with automatic labelling. Subsequently, in step four, β record pairs are sampled to be manually classified by an oracle and to be used for training the next classifier C. Finally, in step five, this new classifier C is trained and record pairs in the active state are classified to obtain new match probabilities $\mathbf{p}$, where $\mathbf{p}[x]$ indicates the probability for record pair x to be a match, with $0 \leq \mathbf{p}[x] \leq 1$. This iterative refinement process continues until there are no more active record pairs left, $|\{x : \boldsymbol{\mathcal{S}}[x] = \mathscr{a}\}| = 0$, or the maximum number of iterations n_c is reached. We next describe each of these steps in more detail.

Step 1: Calculating Classification Scores

The classification score $\mathbf{Z}[x, y]$ reflects the likelihood for a record pair x to be either a match or a non-match in iteration y, based on the threshold δ_p where $0 \leq \delta_p \leq 1$. A positive $\mathbf{Z}[x, y]$ value indicates that a record pair x was classified as a match (if $\mathbf{p}[x] > \delta_p$), a negative value of $\mathbf{Z}[x, y]$ indicates that the pair was classified as a non-match ($\mathbf{p}[x] < \delta_p$), and a score of $\mathbf{Z}[x, y] = 0$ (if $\mathbf{p}[x] = \delta_p$) indicates an ambiguous classification outcome. We consider two methods to calculate a score $\mathbf{Z}[x, y]$ from a classifier probability $\mathbf{p}[x]$.

- *Without confidence scaling*: In this method we convert match probabilities into classification scores by setting $\mathbf{Z}[x, y] = \mathbf{p}[x]$ when $\mathbf{p}[x] > \delta_p$, $\mathbf{Z}[x, y] = -(1 - \mathbf{p}[x])$ when $\mathbf{p}[x] < \delta_p$, and $\mathbf{Z}[x, y] = 0$ when $\mathbf{p}[x] = \delta_p$. Here we assume that the probability of a non-match for record pair x is $1 - \mathbf{p}[x]$ because match and non-match probabilities must add to 1.
- *Applying confidence scaling*: The closer a match probability $\mathbf{p}[x]$ of a record pair x is to the classification threshold δ_p, the higher the ambiguity of its classification. When we calculate the classification scores without confidence scaling, this ambiguity of the classification is not evident. Furthermore, the ranges of the classifier match probabilities $\mathbf{p}[x]$ returned in different iterations may be different, potentially resulting in an implicit bias of the classification result to certain iterations.

  For example, if the match probabilities obtained in the first iteration are in the range $[0.4, 0.9]$, while in the second iteration they are in the range $[0.1, 0.8]$, then the final classification result (averaged over the two iterations, as we discuss below in Step 2) has an implicit bias to the results obtained in the first iteration due to it having higher probability values. To resolve this issue, we adapt the

confidence scaling technique as proposed by Primpeli et al. [142] (which we described in Section 3.3, and presented in Equation 3.2) as follows:

$$\mathbf{Z}[x,y] = \begin{cases} \frac{\mathbf{p}[x]-\delta_p}{\delta_p - min(\mathbf{p})}, & \text{if } \mathbf{p}[x] < \delta_p \\ \frac{\mathbf{p}[x]-\delta_p}{max(\mathbf{p})-\delta_p}, & \text{if } \mathbf{p}[x] > \delta_p \\ 0.0, & \text{if } \mathbf{p}[x] = \delta_p \end{cases} \tag{7.1}$$

When confidence scaling is applied as shown in Equation (7.1), we normalise score values into the $[-1, 1]$ range to adjust for the ambiguity of classification in different iterations. The $min()$ and $max()$ functions return the minimum and maximum values in the input list provided, respectively (here the minimum and maximum match probabilities of active record pairs in $\mathbf{p}$). A classification score $\mathbf{Z}[x, y]$ closer to 0 indicates higher ambiguity and a value closer to ± 1 indicates higher confidence of the classification outcomes for record pair x in iteration y.

Step 2: Weighted Average Score

We now describe how we calculate the vector of weighted average classification scores, $\mathbf{z}$, using the classification scores in the matrix $\mathbf{Z}$. This vector of averaged scores $\mathbf{z}$ is used to decide whether a record pair should be inactivated, and if its final classification should be as a match or a non-match. We consider the classification scores of the k most recent iterations when calculating these weighted averages. We use a linear discounting approach where the most recent score is given a higher weight compared to earlier scores. This is because we expect the classification models in latter iterations to have better predictive power since they have been trained with a larger and more refined training data set that contains more record pairs labelled by the oracle, as well as more inactive record pairs of improved quality.

Specifically, we calculate a vector of weights, $\mathbf{w}$, where we set $\mathbf{w}[j] = j$, with $1 \leq j \leq k$. The classification score of the earliest iteration among the iterations considered for the average score calculation will receive a weight of $\mathbf{w}[1] = 1$, with weights incremented by one for subsequent iterations as $\mathbf{w}[j+1] = \mathbf{w}[j] + 1$. Note that alternative discounting methods would be possible, such as setting each $\mathbf{w}[j]$ to half of $\mathbf{w}[j+1]$. This would however give too much weight to the classification score of the most recent iteration. In each iteration y, we then apply this weighted averaging approach on each active record pair x to calculate its overall score $\mathbf{z}[x]$, as shown in Equation (7.2). If less than k iterations have been performed ($y < k$), then the averaging calculations are only applied on these first y iterations.

$$\mathbf{z}[x] = \begin{cases} \frac{\sum_{j=1}^{k} \mathbf{w}[j] \cdot \mathbf{Z}[x, y-k+j]}{\sum_{j=1}^{k} \mathbf{w}[j]}, & \text{if } y \geq k \\ \frac{\sum_{j=1}^{y} \mathbf{w}[k-y+j] \cdot \mathbf{Z}[x, j]}{\sum_{j=1}^{y} \mathbf{w}[k-y+j]}, & \text{if } y < k \end{cases} \tag{7.2}$$

Note that $-1 \leq \mathbf{z}[x] \leq 1$ due to every score in the matrix $\mathbf{Z}$ being in the $[-1, 1]$ range.

Step 3: Automatic Record Pair Classification

The third step of the iterative refinement phase is the automatic classification (labelling) of record pairs where we are highly confident that they are either a match or a non-match. We automatically label and inactivate selected record pairs such that the efficiency of the linkage process is improved by not considering all inactive record pairs in subsequent iterations. We inactivate record pairs only when at least k iterations have elapsed by calculating the average of their classification scores across the k most recent iterations. We consider two methods to inactivate a record pair x based on its average overall score $\mathbf{z}[x]$.

- *Threshold*: In this method all record pairs in the active state ($\boldsymbol{\mathcal{S}}[x] = a$) with an absolute weighted average score $\mathbf{z}[x]$ (we consider the absolute value since $-1 \leq \mathbf{z}[x] < 0$ for potential non-matches) above a given threshold value δ_a are inactivated. The absolute $\mathbf{z}[x]$ value being above a given threshold δ_a indicates that the corresponding record pair has been classified as a match or a non-match with high confidence in the k most recent iterations.
- *Percentage*: A given percentage δ_b of record pairs in the active state with the highest absolute weighted average scores $\mathbf{z}[x]$ are made inactive in this method.

The inactivated record pairs with a positive weighted average score $\mathbf{z}[x] \geq 0$ are then automatically labelled as matches, while the inactivated record pairs with a negative weighted average score $\mathbf{z}[x] < 0$ are labelled as non-matches. Furthermore, the state of all inactivated record pairs is changed from active (a) to inactive (i) in the vector $\boldsymbol{\mathcal{S}}$ of states ($\boldsymbol{\mathcal{S}}[x] = i$). Note that we do not require the consistent classification of record pairs into a single class in the last k iterations for their inactivation because the classifiers in early iterations tend to make erroneous classifications. Therefore, the weighted average of scores across k iterations, as given in Equation (7.2), is a more apt measure for selecting the record pairs to be inactivated.

Step 4: Record Pair Sampling for Classifier Training

Subsequent to the inactivation step, active record pairs ($\boldsymbol{\mathcal{S}}[x] = a$) are sampled to be manually classified by the oracle, and inactive record pairs ($\boldsymbol{\mathcal{S}}[x] = i$) are sampled to train a next classifier C together with the oracle labelled record pairs. We consider the following two variations of sampling methods, where in both β is the oracle labelling budget for one iteration.

- *Random*: With this method, $\beta/2$ record pairs in the active state are chosen randomly from the potential matches (record pairs where their $\mathbf{z}[x] \geq 0$) and $\beta/2$ from the potential non-matches (where $\mathbf{z}[x] < 0$) in the active state. Because no manual labelling is required for the sampled inactive record pairs (that have been classified automatically as we described above), without costs we can sample more such inactive record pairs into the training set. Assuming $\mathbf{o}$ to be the set of manually labelled record pairs through all iterations (where $|\mathbf{o}| = \beta \cdot y$), we calculate the sampling budget for the inactive set as a multiplier ϵ of the number of oracle labelled record pairs, and then randomly select $|\mathbf{o}| \cdot \epsilon/2$ record

pairs from each of the automatically labelled set of matches and non-matches in the inactive state.

- *Extreme*: With this method, the most ambiguous β record pairs in the active state are selected for manual classification. Higher ambiguity is indicated by weighted average scores $\mathbf{z}[x]$ being closer to 0. We select an equal number of $\beta/2$ record pairs from either sides of the $\mathbf{z}[x] = 0$ boundary, such that both highly ambiguous potential matches and potential non-matches are manually labelled [41]. When sampling inactive record pairs for training the classifier, we select record pairs with the highest absolute weighted average scores $\mathbf{z}[x]$ being closest to 1. These are the match and non-match record pairs with the most confident automated labelling. As with the random sampling method described above, the sampling budget for the inactive set is calculated as a multiplier ϵ of the number of oracle labelled record pairs. We therefore select the $|\mathbf{o}| \cdot \epsilon$ most confident record pairs in the inactive state, allocating an equal budget of $|\mathbf{o}| \cdot \epsilon/2$ to automatically labelled matches and non-matches.

Step 5: Classifier Training and Record Pair Classification

The final step in the iterative refinement phase is to train a classifier C with the sampled inactive record pairs as well as the oracle labelled pairs, and subsequently to classify all record pairs x which are still in the active state ($\boldsymbol{\mathcal{S}}[x] = \mathscr{a}$) to obtain their new match probabilities $\mathbf{p}[x]$. These will then be used in the next iteration as described above.

7.2.4 Algorithmic Outline

Algorithm 8 outlines our proposed active learning based RL technique with filtering. As main input, the algorithm takes $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, an undirected pairwise similarity graph, and a supervised classification algorithm, C. The parameters provided to Algorithm 8 are the confidence scaling flag f (*True* or *False*) we described in Step 1, the record pair inactivation method m_i (*Threshold* or *Percentage* based) as described in Step 3, and the sampling method m_s (*Random* or *Extreme*) that identifies how to sample record pairs for oracle labelling and training the classifier C, as we described in Step 4. The parameter $k\,(k \geq 1)$ specifies the number of past iterations to take into consideration when deciding which record pairs to set to inactive state. The maximum number of iterations to consider and the budget allocated per iteration are given by n_c and β, respectively, whereas the sampling ratio for deciding the number of inactive record pairs to use for training the classifier C is given by ϵ. The number of matches as estimated by the domain expert for the initial classification phase is specified by o_m. The final output of the algorithm are the sets of classified matching and non-matching record pairs $\mathbf{M}$ and $\mathbf{N}$.

In line 1 of the algorithm, we initialise a classification score matrix $\mathbf{Z}$ with $n_r = |\mathbf{G}.\mathbf{E}|$ rows and n_c columns, as described in Section 7.2.1. The classification score obtained for a given record pair x in a certain iteration y is stored in the matrix cell $\mathbf{Z}[x, y]$, where $1 \leq x \leq n_r$ and $1 \leq y \leq n_c$. Next, in line 2, the function

Algorithm 8: Record linkage with active learning and filtering (RALF)

Input: **G** - Undirected pairwise similarity graph
C - Supervised classifier
f - Confidence scaling flag (*True* or *False*)
m_i - Method to inactivate record pairs (*Threshold* or *Percentage*)
m_s - Method to sample record pairs (*Extreme* or *Random*)
k - Number of most recent iterations to consider for inactivation
n_c - Maximum number of iterations
β - Budget per iteration
ϵ - Inactive sampling ratio
o_m - Estimation of the expected number of true matches by domain expert
Output: **M**, **N** - Classified match and non-match record pairs

1 $\mathbf{Z} = \textbf{InitScoreMatrix}(n_r = |\mathbf{G}.\mathbf{E}|, n_c)$ // Initialise the score matrix
2 $\mathbf{p}, \delta_p = \textbf{ApproxMatchClass}(\mathbf{G}, o_m)$ // Get initial classification
3 $y = 1; \boldsymbol{\mathcal{S}}[x] = \mathscr{a}, 1 \leq x \leq n_r$ // Initialise iteration counter and score vector
4 **while** $y \leq n_c \wedge |\{x : \boldsymbol{\mathcal{S}}[x] = \mathscr{a}\}| > 0$ **do** // While iterations / active pairs
5 $\quad \mathbf{z} = \textbf{CalcScore}(\mathbf{Z}, \mathbf{p}, \delta_p, f, y, k)$ // Calculate the scores and averages
6 $\quad$ **if** $y \geq k$ **then**
7 $\quad\quad \textbf{SetInactive}(\mathbf{Z}, m_i, \mathbf{z}, \boldsymbol{\mathcal{S}})$ // Inactivate selected record pairs
8 $\quad \mathbf{o} = \textbf{Oracle}(\{x : \boldsymbol{\mathcal{S}}[x] = \mathscr{a}\}, m_s, \mathbf{z}, \beta, \boldsymbol{\mathcal{S}})$ // Oracle labelling of active pairs
9 $\quad \mathbf{D}_t = \textbf{GetTrainData}(\mathbf{o}, m_s, \mathbf{z}, \{x : \boldsymbol{\mathcal{S}}[x] = \mathscr{i}\}, |\mathbf{o}| \cdot \epsilon)$ // Get training data
10 $\quad C.train(\mathbf{D}_t)$ // Train the classifier
11 $\quad \mathbf{p}, \delta_p = C.classify(\{x : \boldsymbol{\mathcal{S}}[x] = \mathscr{a}\})$ // Classify active record pairs
12 $\quad y = y + 1$ // Increment the iteration counter
13 $\mathbf{M}, \mathbf{N} = \textbf{GetFinalClass}(\mathbf{Z}, \mathbf{z}, \boldsymbol{\mathcal{S}})$ // Get final matches and non-matches
14 **return** **M**, **N** // Return the final matches and non matches

ApproxMatchClass() conducts the initial classification of record pairs in **G** based on the estimated number of true matches o_m, as we described in Section 7.2.2. This function returns a list of match probabilities, **p**, containing one probability for each record pair in the graph **G**, and a threshold probability δ_p for classifying record pairs as matches and non-matches. In line 3 the current iteration y is initialised to 1, and the score vector $\boldsymbol{\mathcal{S}}$ is initialised by setting all record pairs to active state ($\boldsymbol{\mathcal{S}}[x] = \mathscr{a}$).

The iterative active learning based classification phase described in Section 7.2.3 starts at line 4. This phase continues until the maximum number of iterations n_c is reached, or there are no active record pairs left, identified by $|\{x : \boldsymbol{\mathcal{S}}[x] = \mathscr{a}\}| = 0$.

In line 5, using the function **CalcScore**(), the classification scores for iteration y are calculated and stored in the matrix **Z**, as we described in Step 1, using the match probabilities **p**, the threshold value δ_p, and the confidence scaling flag f. Furthermore, the weighted average scores $\mathbf{z}[x]$ are calculated for all record pairs, where we consider the classification scores obtained across up to the k most recent iterations, as we discussed in Step 2.

Subsequently, the record pairs in **Z** are automatically labelled and inactivated according to the inactivation method m_i, and the weighted average scores in **z**, as we discussed in Step 3. This record pair inactivation is conducted if at least k iterations have elapsed (lines 6 and 7), whereas the states of inactivated record pairs are

changed from active a to inactive i in vector $\boldsymbol{\mathcal{S}}$ ($\boldsymbol{\mathcal{S}}[x] = i$). In lines 8 and 9, we then sample record pairs for manual classification and for training the classifier, as we described in Step 4, based on the weighted average scores of record pairs in $\mathbf{z}$. In line 8 the oracle is given a sample of β active record pairs (with $\{x : \boldsymbol{\mathcal{S}}[x] = a\}$), where the sampling method is identified by m_s. The states of the record pairs labelled by the oracle are changed to *oracle labelled* ($\boldsymbol{\mathcal{S}}[x] = \ell$), and the set of record pairs labelled by the oracle across all iterations $\mathbf{o}$ is returned. Then, in line 9, we sample $|\mathbf{o}| \cdot \epsilon$ record pairs from the inactive set ($\{x : \boldsymbol{\mathcal{S}}[x] = i\}$) using the same sampling method m_s. These sampled inactive record pairs, together with all oracle labelled record pairs $\mathbf{o}$ are used in line 10 to train the classifier C. Next, in line 11, record pairs in the active state are classified using the trained classifier C, which returns a new vector of match probabilities $\mathbf{p}$ and a match probability threshold δ_p. The iteration counter y is incremented by one in line 12.

The final classification results are determined for all record pairs in matrix $\mathbf{Z}$ in line 13. Each record pair x in the active state ($\boldsymbol{\mathcal{S}}[x] = a$) with a positive weighted average score ($\mathbf{z}[x] \geq 0$), together with the record pairs automatically and manually labelled as matches among the inactive and oracle labelled record pairs, respectively, are added to the final match set $\mathbf{M}$. The remaining record pairs in the matrix $\mathbf{Z}$ are added to the final non-match set $\mathbf{N}$, and these two final sets are returned in line 14.

Complexity analysis: We now describe the complexity of our proposed method. Since we consider the classification algorithm C as a black box, we exclude the complexity of the training and classification functions of C from our analysis. For ease of presentation, we will refer to the number of record pairs in the active state as n_a where $n_a = |\{x : \boldsymbol{\mathcal{S}}[x] = a\}|$. The initial classification step (line 2 in Algorithm 8) has a complexity of $O(n_r \cdot log(n_r))$, with $n_r = |\mathbf{G}.\mathbf{E}|$, because we can use a heap-sort algorithm [91] to sort the edges $\mathbf{G}.\mathbf{E}$ in the similarity graph to obtain the o_m record pairs with the highest similarities that are expected to correspond to matches, as estimated by the domain expert.

For each iteration, the classification score calculation step has a complexity of $O(n_a)$, where we conduct a linear scan through the active record pairs in $\boldsymbol{\mathcal{S}}$. For record pair inactivation using the *Threshold* based technique, the complexity is $O(n_a)$, whereas for the *Percentage* based technique the complexity is $O(n_a \cdot log(n_a))$ because this technique requires sorting of the active record pairs. Similarly, record pair sampling for oracle labelling and classifier training has a time complexity of $O(n_a)$ if the *Random* sampling technique is used, and $O(n_a \cdot log(n_a))$ if *Extreme* sampling is used. Since the iterative phase of our approach is mostly dependent on the number of active record pairs, n_a, if these record pairs are inactivated rapidly then the efficiency of the algorithm improves.

The worst case time complexity of the algorithm considering a maximum of n_c iterations is $O(n_r \cdot log(n_r) + n_c \cdot n_a \cdot log(n_a))$. Given $n_r \gg n_c$ and $n_r \geq n_a$, the overall complexity of our approach is dominated by the initial classification phase, whereas the iterative refinement phase will be dependent upon the complexity of the classifier C used in lines 10 and 11 of Algorithm 8.

7.3 Experimental Evaluation

We evaluated our novel active learning with filtering-based RL approach (which we abbreviate as RALF) using the real-world bibliographic data sets DBLP-ACM and DBLP-Scholar, the real-world birth data set IoS, the synthetic birth data set UK, and the real-world voter data set NCVR, which we described in Section 2.5. We use the large NCVR data set only to evaluate the efficiency of our approach compared to employing fully supervised classification methods. For the birth data sets, we use the pairwise similarity graphs generated by comparing *All* attribute values $\mathbf{G}_A$ (see Section 2.6 on page 30) since they contain the maximum number of features for a classifier C to learn from.

We conduct experiments with various parameter settings on the remaining four data sets. Ground-truth data is available for all five data sets in the form of true matching record pairs as assessed by domain experts in the relevant fields, and we use the number of true matches in these ground-truth data sets as the estimation of the expected number of true matches, o_m (see Table 2.1 on page 25), in the initial classification phase of our approach, as we discussed in Section 7.2.2.

We implemented our algorithm in Python 2.7, and all experiments were conducted on a server running Ubuntu 18.04 with 64-bit Intel Xeon 2.10 GHz CPUs and 512 GB of memory. The classification algorithms C we used are a Decision Tree (Tree), Logistic Regression (Reg), Support Vector Machines (SVM), and AdaBoost (Ada) [20], as provided by the Scikit-learn [139] Python machine learning library. We used the default parameter settings for all classifiers, and for repeatability we set the *random_state* parameter (which identifies how record pairs would be shuffled) to 0.

As we described in Section 7.2.4, we set the confidence scaling flag f to *True* or *False*, the record pair inactivation method m_i to *Threshold* or *Percentage*, and the record pair sampling method m_s to either *Extreme* or *Random*. We set the number of most recent iterations to consider for inactivation as $k \in [1, 5, 10, 15]$, whereas the maximum number of iterations n_c we set to 10 or 20. The oracle labelling budget per iteration was set to $\beta \in [20, 50, 100]$, and inactive sampling ratio as $\epsilon = [0, 1, 2]$, where $\epsilon = 0$ implies that no inactive (automatically classified) record pairs were used to train the classifier C. Based on n_c and β, we evaluated total budgets $\beta \cdot n_c$ as 200, 500, 1000, and 2000. Furthermore, for the inactivation method $m_i = Threshold$, we set the threshold values $\delta_a \in [0.99, 0.95, 0.9, 0.85, 0.8]$, while for $m_i = Percentage$, we set the percentage values $\delta_b \in [0.005, 0.01, 0.015, 0.02]$.

For evaluating the linkage quality of our active learning-based filtering approach, we use the measures of precision (P) and recall (R) which we described in Section 2.4. We also use the recently proposed F^*-measure (see Equation 2.5) which can be used as an alternative to the F-measure as we discussed in Chapter 4 in page 77.

7.3.1 Linkage Quality With Different Parameter Settings

We first discuss how the linkage quality achieved with our proposed technique varies when different parameter settings are used.

Table 7.2: Comparison of the parameter settings for sampling, inactivation, and confidence calculation.

Data set	Linkage quality	Sampling method m_s		Inactivation method m_i		Confidence scaling f	
		Random	Extreme	Threshold	Percentage	True	False
DBLP ACM	P	0.95 ± 0.1	0.87 ± 0.2	0.89 ± 0.2	0.93 ± 0.1	0.91 ± 0.2	0.90 ± 0.2
	R	0.97 ± 0.0	0.85 ± 0.3	0.91 ± 0.2	0.92 ± 0.2	0.92 ± 0.2	0.91 ± 0.2
	F^*	0.92 ± 0.1	0.73 ± 0.3	0.81 ± 0.3	0.85 ± 0.2	0.84 ± 0.3	0.82 ± 0.3
DBLP Scholar	P	0.94 ± 0.1	0.97 ± 0.1	0.95 ± 0.1	0.96 ± 0.1	0.95 ± 0.1	0.96 ± 0.1
	R	0.96 ± 0.1	0.18 ± 0.3	0.55 ± 0.5	0.58 ± 0.5	0.58 ± 0.5	0.56 ± 0.5
	F^*	0.90 ± 0.1	0.15 ± 0.3	0.51 ± 0.4	0.55 ± 0.4	0.54 ± 0.4	0.52 ± 0.4
IoS	P	0.70 ± 0.3	0.90 ± 0.2	0.89 ± 0.2	0.68 ± 0.4	0.79 ± 0.3	0.80 ± 0.3
	R	0.84 ± 0.2	0.20 ± 0.2	0.51 ± 0.4	0.53 ± 0.4	0.53 ± 0.4	0.50 ± 0.4
	F^*	0.59 ± 0.3	0.14 ± 0.1	0.44 ± 0.4	0.28 ± 0.3	0.38 ± 0.3	0.35 ± 0.3
UK	P	0.72 ± 0.3	0.88 ± 0.3	0.88 ± 0.3	0.69 ± 0.4	0.79 ± 0.3	0.81 ± 0.3
	R	0.93 ± 0.1	0.68 ± 0.1	0.80 ± 0.2	0.81 ± 0.2	0.81 ± 0.2	0.80 ± 0.2
	F^*	0.66 ± 0.3	0.57 ± 0.2	0.69 ± 0.2	0.52 ± 0.3	0.61 ± 0.3	0.62 ± 0.3
Averages	F^*	**0.76**	0.40	**0.61**	0.55	**0.59**	0.58

- *Quality Analysis for Functional Parameters*: Table 7.2 shows a comparison of the linkage quality achieved for different values for sampling method m_s, inactivation method m_i, and confidence scaling method f. The presented quality values are averages and standard deviations across different parameter settings, with the opposing parameter value ignored. For example, when calculating the averages and standard deviations for *Random* we ignored all results obtained with the *Extreme* parameter setting.

  While the *Random* sampling method has performed better than *extreme* sampling with regard to both precision and recall on the DBLP-ACM data set, *Extreme* sampling has achieved slightly better precision and much worse recall than the *Random* method for the remaining data sets. With further investigation, we were able to identify that for data sets DBLP-Scholar, IoS and UK, the average recall with *Extreme* sampling is very low due to early termination of our algorithm when $k > 1$. With $k > 1$, no record pairs would be inactivated in the first iteration itself, and as a result, the next classifier $= C$ has to be trained only using the oracle labelled record pairs. When the *Extreme* method is used to sample record pairs, depending upon the distribution of the similarities between records, there is a possibility of sampling record pairs from a single class. Since the classifier C cannot be trained with data from a single class (match or non-match) only, our algorithm terminates after only one iteration. The average F^* values in Table 7.2 show that *Random* is superior to *Extreme* sampling.

  For the two inactivation methods (m_i) considered, the *Percentage* method produces better precision and recall on average for the smaller bibliographic data sets, whereas the *Threshold* method achieves better results for the IoS and UK data sets with considerably better precision and slightly lower recall only.

  For all data sets except DBLP-ACM, recall improves and precision declines slightly when confidence scaling is applied ($f = True$) compared to not apply-

ing confidence scaling, whereas with DBLP-ACM both measures show a slight improvement with confidence scaling. The F^* values however have improved for all except the UK data set when confidence scaling was applied, which indicates that the improvement in recall surpasses the slight decline in precision in most experiments.

In summary, as shown with the highlighted average F^* values in Table 7.2, *Random* sampling is better than *Extreme* sampling, *Threshold* inactivation is superior to *Percentage* based inactivation, and applying confidence scaling ($f = True$) produces slightly better linkage results than when no scaling is applied.

We also analysed the trade-off between the efficiency and the quality of our algorithm considering the larger IoS and UK data sets with regard to the choice of functional parameter settings. We did not consider the DBLP data sets because they are too small in size to obtain a proper understanding of the algorithm efficiency. For both the IoS and UK data sets, *Random* sampling indicated a five-fold increase in run-time compared to *Extreme* sampling, the *Threshold* inactivation method was approximately twice as efficient as the *Percentage method*, and the choice between applying or disregarding confidence scaling did not show a considerable difference in the run-time. *Extreme* sampling takes less time since it leads to early termination of the algorithm, and therefore, it is not necessarily better than the *Random* approach with regard to efficiency.

- *Quality Analysis for Other Parameters*: Figure 7.2 shows how the linkage quality of our active learning technique changes with varying parameter values. As shown in Figure 7.2 (a), the performance of the AdaBoost, SVM, and Decision Tree classifiers on the four data sets are quite similar, whereas the Logistic Regression classifier has the worst performance. Figure 7.2 (b) shows an upward trend in F^* results obtained with increasing oracle budgets β and ratio values ϵ, since when both β and ϵ are large we have more manually and more automatically labelled record pairs for training the classifier C. For a given oracle budget and increasing values of ϵ, however, the quality does not seem to improve much which is potentially caused by increased possibility of making errors in the automatic labelling when more record pairs are labelled automatically.

  The quality as measured with F^* improves for increasing values of k (the number of iterations to consider for inactivating record pairs) as shown in Figure 7.2 (c). This is to be expected because with larger k we take the classification outcome of more classifiers into account when deciding which record pairs to automatically label and inactivate, thereby improving the reliability of our decisions. Similarly, as shown in Figure 7.2 (d), with larger oracle budgets $\beta \cdot n_c$, the classifier has more manually labelled data to learn from, thus improving the linkage quality. However, the run-time of the algorithm also increases with the total oracle budget. This is because we calculate the total budget as a multiplication of the budget per iteration, β, and the total number of iterations, n_c. Therefore, a larger total oracle budget $\beta \cdot n_c$ means that the algorithm executes a higher number of iterations, which takes more time.

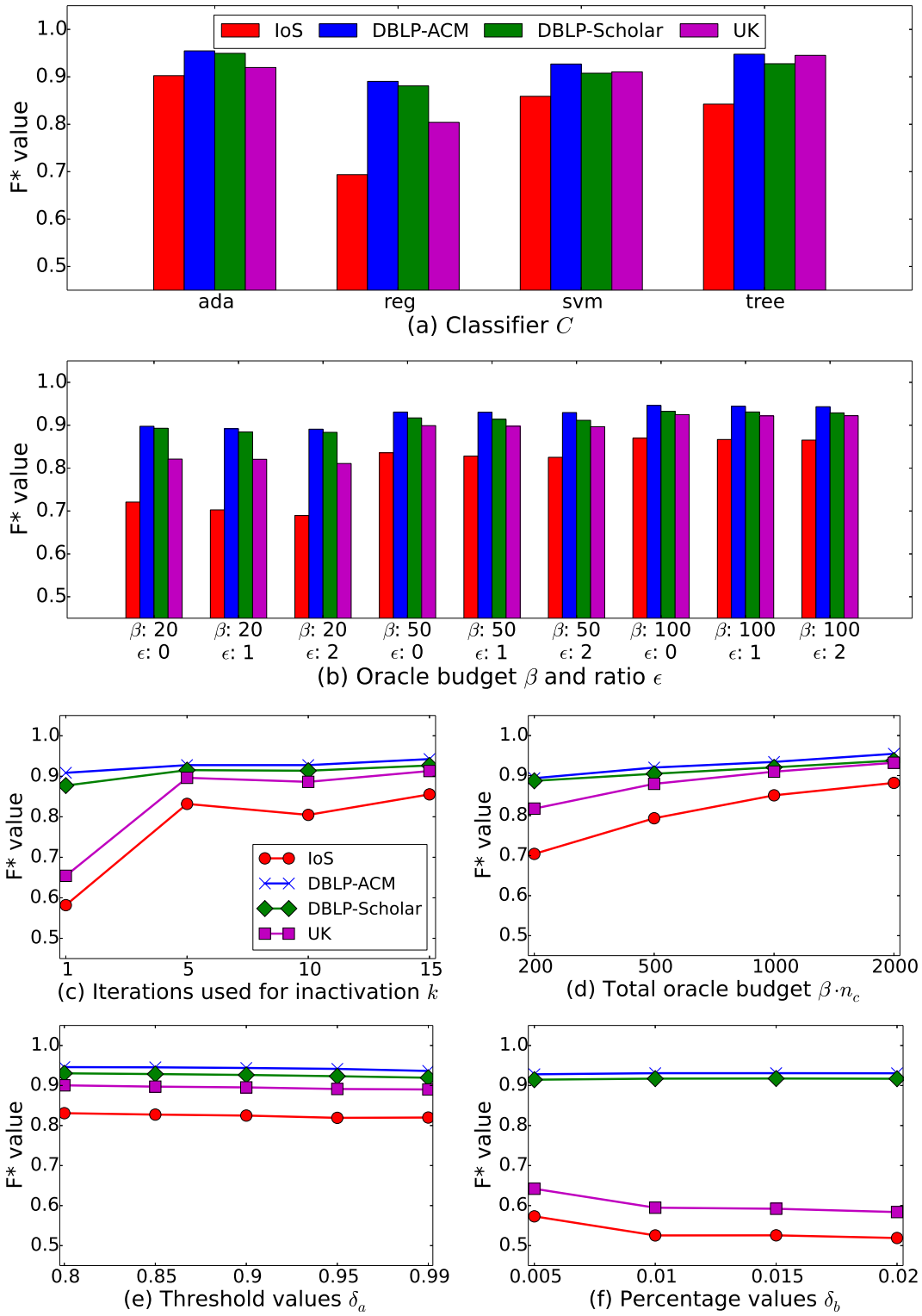

Figure 7.2: F^* values obtained with different parameter settings.

Table 7.3: Comparison of the linkage quality (F^*) of our proposed RALF technique with state-of-the-art deep learning and active learning baselines. The oracle labelling budgets for the active learning techniques are shown within brackets. For the DBLP-Scholar data set $F^* = 0.95$ was obtained with a budget of 1,000 for RALF method.

Data Set	Meduri et al. [116]	Mudgal et al. [120]	Magellan [103]	Christen et al. [41]	RALF
DBLP-ACM	**0.98** (260)	0.97	0.97	0.92 (1,000)	**0.98** (1,000)
DBLP-Scholar	**0.98** (1,770)	0.90	0.86	0.90 (1,000)	**0.98** (2,000)

The last two plots in Figures 7.2 (e) and 7.2 (f) indicate how linkage quality changes with different thresholds δ_a when record pairs are inactivated using the *Threshold* technique, and different percentage values δ_b when the *Percentage* inactivation method is used, respectively. As can be seen, the quality does not change much for varying δ_a due to an improvement in precision and decrease in recall as δ_a is increased. Changing δ_b also does not affect linkage quality of the smaller DBLP data sets. However, linkage quality does decline with increasing values of δ_b for the larger IoS and UK data sets because of a reduction in precision resulting from the automatic labelling of too many record pairs.

For all the results presented in this section, we assumed a perfect oracle with 100% accuracy in conducting manual labelling. With only 90% oracle accuracy, the F^* values are reduced by 14.5% on average. We also experimented with two values, 10 and 20, for the maximum number of iterations n_c, and found the F^* results to improve by 8% on average for larger values of n_c.

7.3.2 Linkage Quality Comparison With State-of-the-art Techniques

Table 7.3 shows the linkage quality achieved with our proposed RALF technique, and existing state-of-the-art active learning based linkage approaches by Meduri et al. [116] and Christen et al. [41], deep learning based RL techniques proposed by Mudgal et al. [120], and Magellan, a renowned machine learning based framework for RL proposed by Konda et al. [103]. We have used the DBLP data sets as commonly used for evaluating linkage quality in the corresponding papers.

We were able to achieve the best average quality of $F^* = 0.98$ with Decision Tree-based classification and $k = 15$, using a total oracle labelling budget $\beta \cdot n_c$ of 1,000 for DBLP-ACM and 2,000 for DBLP-Scholar. With $\beta \cdot n_c = 1,000$ for DBLP-Scholar, we obtained an average quality of $F^* = 0.95$. We were able to surpass the linkage quality achieved with all state-of-the-art techniques, while being comparable with the performance of the active learning framework proposed by Meduri et al. [116]. On average, we have been able to improve linkage quality by an average 6.7% compared to these baselines. The full data set was used for training with Magellan and the framework proposed by Mudgal et al., whereas for the active learning technique proposed by Christen et al. [41], an oracle labelling budget of 1,000 was used. Meduri et al. [116] used a budget of 260 for DBLP-ACM and 1,770 for DBLP-Scholar, whereas

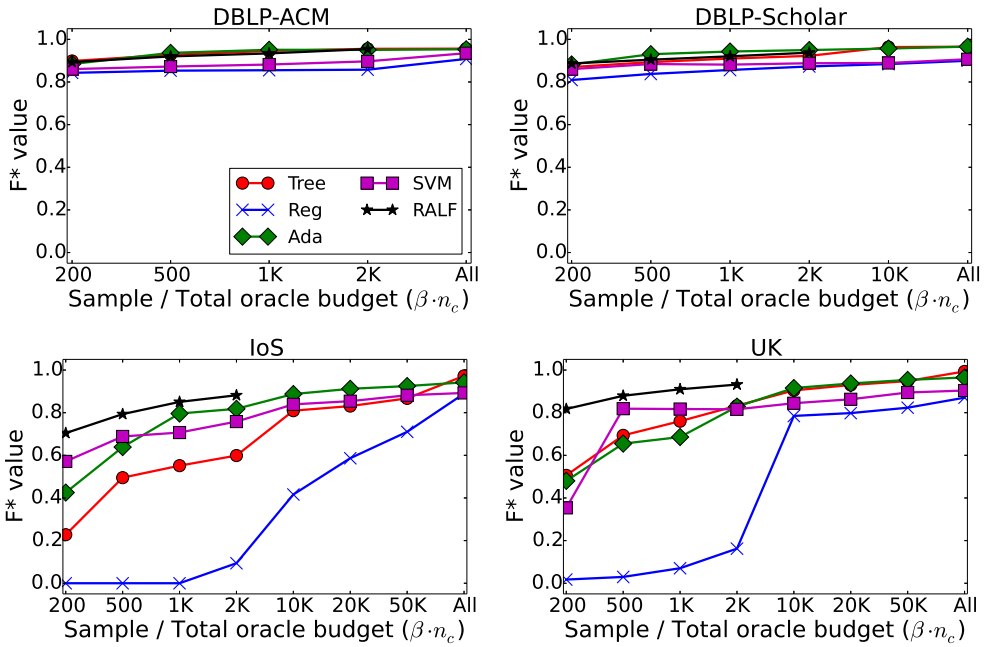

Figure 7.3: The F^* results obtained with different supervised classification algorithms with different training sample sizes, and our proposed active learning technique with different total oracle labelling budgets $\beta \cdot n_c$.

a Random Forest classifier with 20 decision trees produced the best results in their active learning framework. Considering the labelling budgets, the method proposed by Meduri et al. [116] is still slightly better than our RALF approach.

7.3.3 Linkage Quality and Efficiency Comparison With Supervised Classification Techniques

In this section we first compare the linkage quality of RALF and supervised classification techniques, with the aim of justifying using our RALF approach rather than applying supervised classification with a randomly sampled set of labelled record pairs for training. As shown in Figure 7.3, we increased the training sample size for supervised classifiers from 200 to the full data set size, where record pairs are randomly sampled such that an equal number of matches and non-matches are selected for training, except in the instance where training is conducted on the full data set. We consider total budgets $\beta \cdot n_c$ of 200, 500, 1,000 and 2,000 for oracle labelling in our RALF method, and report the average quality obtained for each total budget.

When applied on the DBLP data sets, our RALF method is either on par with, or only marginally surpasses the linkage quality achieved with supervised classifiers. The reason for this is due to the DBLP data sets being clean, structured, and relatively small in size. However, for the larger IoS and UK data sets which lack in data quality, the improvement in linkage quality achieved with our RALF method is higher when compared to using supervised learning techniques. The linkage quality

Table 7.4: Time taken in seconds to run fully supervised (FullSup) classification and our proposed RALF technique, and the percentage run-time reduction.

Data	IoS			UK			NCVR		
Set	FullSup	RALF	Reduction	FullSup	RALF	Reduction	FullSup	RALF	Reduction
Ada	1,998	986	50.65%	1,119	803	28.24%	14,994	9,525	36.47%
SVM	116,876	753	99.36%	55,462	580	98.95%	2,592,000	6,747	99.74%

we have been able to achieve with only 2,000 oracle labels has been surpassed by the supervised techniques only when much larger sets of labelled record pairs, or the full data sets, were used for training. This implies that for large and dirty data sets (which is often the nature of real-world databases), randomly sampled training data is inadequate for achieving high linkage quality. Rather, we need a careful selection strategy for deciding which record pairs to use for classifier training as done in our RALF method. The quality improvements seen for the IoS and UK data sets in Figure 7.3 justify our approach to selecting record pairs for manual and automated labelling based on a weighted average score value, and the method we adopt for subsequent training data selection.

Table 7.4 shows a comparison of the average time taken to run RALF and the time taken with the supervised classification techniques. We only compare our method with AdaBoost and SVM since they are the best performing supervised algorithms on the larger data sets. Furthermore, we disregard the smaller DBLP data sets, since linking those does not consume much time, whereas we consider a very large data set NCVR with 34.5 million record pairs (as we showed in Table 2.6 in page 31), together with IoS and UK. While we consider the average time across all parameter settings for IoS and UK, for NCVR we only consider $k = 5$ and maximum iterations $n = 10$, since high linkage quality could be achieved with these settings in less time.

As shown in Table 7.4, the RALF method is significantly more efficient than running supervised classification on the full data sets, for both the AdaBoost and SVM algorithms and especially with SVM. This is due to the training complexity of SVM being exponential (between $O(n^2)$ and $O(n^3)$) by the number of training record pairs [22]. On average, our RALF method has achieved an efficiency improvement (reduction in run-time) of 28% to 50% for AdaBoost, and above 98% for SVM, compared to running fully supervised classification with AdaBoost and SVM.

7.4 Summary

In this chapter, we have presented a novel active learning-based RL technique with filtering, where we propose a novel strategy to select record pairs to conduct manual labelling, and utilise an iterative classification method to obtain a larger training data set. Ours is the first method to propose a record pair filtering strategy based on active learning for improving the efficiency of the classification step in the RL process.

With an empirical evaluation conducted on four real-world and one synthetic data set, we have shown that our proposed technique is on average 28% to 99% more efficient than using complex supervised classification techniques for RL. Furthermore, we showed that our method outperforms state-of-the-art active learning, deep learning, and machine learning linkage approaches by an average 6.7% with regard to linkage quality. Our experiments also showed that our proposed method is more suitable for linking large data sets which lack in data quality.

As future work, we hope to explore whether the linkage quality can be further improved by classifying an equal number of matches and non-matches in the automatic labelling step, rather than using a threshold-based or percentage-based technique, which can be affected by the class imbalance. Furthermore, as future work we intend to experiment with a Random Forest classifier, given the active learning method proposed by Meduri et al. [116] slightly outperforms our approach with regard to the number of labelled record pairs when a Random Forest classifier is used. In the next chapter, we discuss limitations in existing evaluation approaches for assessing group RL techniques and propose a novel evaluation measure for group RL.

Chapter 8

An Evaluation Technique for Group Record Linkage

As we discussed in Section 2.3, group Record Linkage (RL) methods, as opposed to traditional pairwise RL techniques, have recently gained popularity due to their applicability in linking groups of entities, such as individuals in a household or publications by the same author. However, as we highlighted in Section 1.2, limited research has been conducted on the suitability of existing evaluation measures to assess group RL methods. In this chapter we discuss why existing evaluation measures may produce ambiguous results, and we propose a novel evaluation measure for assessing group RL techniques.

In Section 8.1 we provide an overview of the issue of using existing evaluation methods on group RL with an example, and in Section 8.2 we present our proposed new measure. Next, in Section 8.3 we apply this novel evaluation measure on three group RL techniques and compare the results with the assessments made based on existing RL evaluation measures. Finally, in Section 8.4 we provide a summary of this new evaluation measure and conclude this chapter.

8.1 Introduction

While with traditional RL approaches the aim is to identify individual records referring to the same entity, the purpose of applying group RL methods is to identify related sets of entities [134]. Graph-based clustering [86, 153] approaches are commonly applied to tackle the problem of group RL, where a similarity graph is initially created representing records as vertices and the pairwise similarities as the weights of the edges connecting vertex pairs, as we described in Definition 2. As we showed in Chapter 4, a graph-based clustering technique applied on such a similarity graph aims to cluster the densely connected areas of the graph (where there are many vertices connected by edges indicating that these records belong to the same group of entities), whereas sparsely connected or unconnected vertices represent entities which do not belong to that particular group [153].

As automated linkage techniques are now increasingly being used across many domains [17], one crucial question is how well do such techniques perform - i.e.

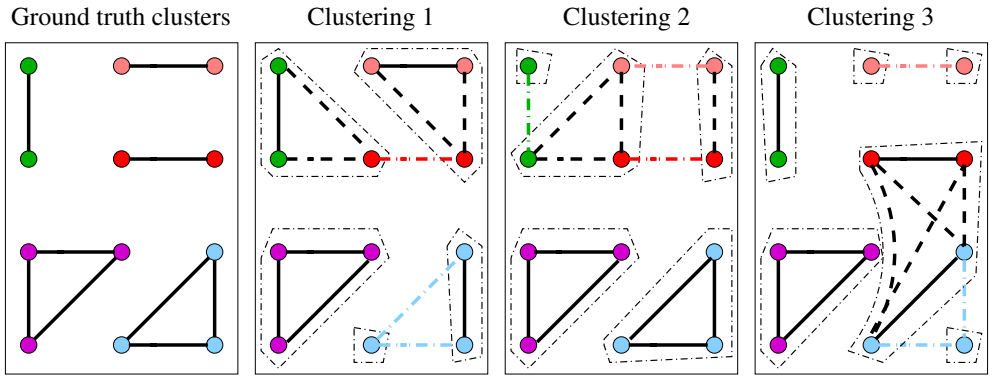

Figure 8.1: Examples of different cluster predictions. Node colors represent the five true clusters, solid edges show true matches (correctly predicted links), dotted black edges show wrong matches (incorrectly predicted links), and dotted coloured edges show missed matches (true matches not linked in the prediction).

how accurate are the links identified by these techniques? In order to calculate a numerical linkage accuracy measure, ground-truth data in the form of true matches (pairs of records believed - with a predetermined level of certainty - to refer to the same entity) and non-matches (pairs of records believed - with a predetermined level of certainty - to refer to different entities) need to be available.

However, the evaluation of the quality of clustering approaches for group RL is not a straightforward undertaking. The reason for this is that some predicted clusters might only be partially correct (a cluster might contain some correct links and some wrong links). This can make the identification of which ground-truth cluster is represented by which predicted cluster difficult.

As an example, when bundling birth records by the same parents, each cluster is supposed to represent the children of one mother and father. However, a clustering algorithm might generate some predicted clusters that are only partially correct. For instance, the true sibling group $\{r_a, r_b, r_c\}$ might be split into two clusters $\{r_a, r_b\}$ and $\{r_c, r_d\}$, where birth record r_d by other parents was wrongly linked to record r_c.

So far, most researchers working on group RL (or clustering) problems have adopted the traditional classification evaluation measures of precision and recall, which we formally defined in Section 2.4. Precision is calculated as the ratio of how many of the predicted links between records are in fact true matches (i.e. seen in the ground-truth matches), while recall is calculated as the ratio of how many of the true matches were correctly predicted as matches by the classification algorithm. Both these measures however are based on the evaluation of links between individual records (record pairs) rather than clusters of records.

Despite the widespread use of these two measures to evaluate the quality of clustering and group RL results, obtaining the same precision and recall values for different clustering results does not necessarily reflect linkage outcomes of comparable quality, as we describe later in this section. For traditional pairwise RL, precision and recall are suitable measures, assuming one is interested in the quality of the

predicted links between individual records [35, 85]. However, for group RL these measures do not provide detailed enough information about the predicted clusters. Let us explain this limitation of precision and recall with an example.

In Figure 8.1, we show a simple ground-truth clustering of five entity groups (clusters) in the left-most plot, and three different cluster prediction outcomes in the other three plots. In this example, we assume the five ground-truth clusters (two made of three records and three made of two records) were created manually by an experienced domain expert whose linkage outcome can be seen to be correct with high confidence. The three clustering outcomes 1, 2 and 3, on the other hand, are the linkage outcomes generated by three different automated RL clustering algorithms. Each ground-truth cluster in Figure 8.1 is represented using a different vertex colour. In each of the three predicted clustering results, we can see that:

- the number of true matches (true positives as defined in Section 2.4) is 6,
- the number of false matches (false positives) is 4, and
- the number of missed true matches (false negatives) is 3.

Therefore, all three of these very contrasting clustering results obtain the same precision, P, and recall, R, values of $P = 6/10 = 0.6$ and $R = 6/9 = 0.667$, respectively. However, the three clustering results generated by the algorithms are all very different from one another. Depending upon the use of these linked data, for example in a public health or social science research study, one or the other of these three clustering outcomes might be more useful. For example, a health researcher who is interested in studying siblings of larger families would likely prefer clustering 2, where two of the three clusters of size three are correct (whereas for clustering 1 only one of the three clusters of size three is correct, and for clustering 3 there is a large wrong cluster of size four). This example shows how misleading, in the domain of group RL, the use of link-based evaluation measures such as precision and recall can sometimes be, and that the comparison of different linkage methods based on precision and recall might not be suitable.

As we presented in Equations 3.4 and 3.5, Hassanzadeh et al. [86] proposed two measures named clustering precision (CPr) and penalised clustering precision (PCPr) for evaluating group RL algorithms. CPr reflects the average number of record pairs which are correctly grouped together in a single cluster. The PCPr method is the same as CPr but it penalises algorithms that generate fewer or a larger number of clusters compared to the ground-truth cluster count. A major drawback of these two evaluation measures, however, is the possibility of mapping several predicted clusters to a single ground-truth cluster. For example, assume a ground-truth cluster $\{r_a, r_b, r_c, r_d\}$ was split into two in the prediction as $\{r_a, r_b\}$ and $\{r_c, r_d\}$. In the evaluation with CPr and PCPr, both clusters $\{r_a, r_b\}$ and $\{r_c, r_d\}$ are considered as correct groupings with respect to the ground-truth cluster, which means that a single ground-truth cluster is mapped to two different predicted clusters. Such a one-to-many mapping between ground-truth and predicted clusters is unacceptable in the

group RL domain, since it is incorrect to interpret two different groups as a single group (such as two predicted families as referring to one true family).

To address the problem of the lack of suitable linkage evaluation measures for group RL, we propose a novel method for evaluating the quality of the clusters generated in a linkage process, which assesses records (rather than links) according to how correctly they have been grouped when compared to the ground-truth clusters. We aim to answer the question "which linkage method has generated clusters that are closest to the ground-truth". We also aim to resolve the issue of one-to-many and many-to-many mappings between ground-truth clusters and predicted clusters as done in the group RL evaluation measure proposed by Hassanzadeh et al. [86]. Since our proposed evaluation method relies on the linkage outcome alone, it is applicable for assessing any group RL method regardless of the sources being multiple data sets or a single data set.

8.2 Proposed Evaluation Method

Assuming a linkage of large data sets, our proposed clustering quality evaluation method considers how individual records have been allocated into predicted groups/clusters (the result of a clustering algorithm), with respect to how they appear in the ground-truth clusters. Each ground-truth cluster contains true matching record pairs that were manually identified by a domain expert.

We follow a standard RL process as we described in Section 2.3 where we block a pre-processed and cleaned data set(s), and compare record pairs within these blocks. We then generate a pairwise similarity graph **G** as detailed in Definition 2, on which a clustering algorithm such as those described in Chapter 4 is applied to conduct group RL. While the clustering technique itself is considered to be a *black box* (meaning that the functionality of the clustering algorithm is of disinterest to us) we assume that the used technique results in non-overlapping predicted clusters, where each cluster is assumed to represent a single entity group. Some of these clusters are singletons (contain a single record) whereas others contain several records. The union of all predicted clusters contains all records in the data set(s) on which RL was conducted.

8.2.1 Record-based Cluster Evaluation

As we highlighted earlier in this chapter, traditionally the precision and recall measures (or measures derived from the precision and recall such as the *F*-measure and the area under the precision-recall curve or AUPRC) are used to assess the correctness of the compared record pairs [35, 85]. The precision and recall are calculated as detailed in Section 2.4 by categorising record pairs as true positives, false positives, true negatives and false negatives, based on how record pairs appear in the ground-truth clusters and the predicted clusters.

As shown in Figure 8.1, precision and recall are not suitable for evaluating group RL methods that generate clusters of records because they can produce ambiguous results. They are based on the classification of links, but not of records. A user who

Table 8.1: Classification of records into categories for our evaluation measure.

Category	**Description**
Correct singleton (SS)	Records which appear as singletons in both the ground-truth and the predicted clusters.
Wrongly grouped singleton (SG)	Records which appear as singletons in the ground-truth but were assigned to a group of records in the prediction.
Exact group match (GG_E)	Records contained in a predicted cluster that exactly matches a ground-truth cluster (i.e. each record in the predicted cluster appears in the same ground-truth cluster, and vice versa), where the size of the cluster is larger than one.
Majority group match (GG_M)	A majority group match occurs when at least 50% of the records in a predicted cluster (with more than one record) come from a single ground-truth cluster. For this classification, the best representative predicted cluster of a ground-truth cluster (which contains at least two records from the ground-truth cluster) must be identified. For a majority group match, all the records which appear in both the ground-truth cluster and predicted cluster are assigned to category GG_M, while all other records are classified either into category GS or GG_W.
Minority group match (GG_m)	A minority group match is similar to a majority group match, however less than 50% of the records in a predicted cluster come from the corresponding ground-truth cluster.
Wrongly assigned member (GG_W)	These are all the records from a ground-truth cluster (with more than one record) which appear in a predicted cluster (with more than one record) different to the majority or minority group match. That is, once we find the best representative cluster for a given ground-truth cluster, all the records which appear in a predicted cluster other than the representative cluster are assigned to this category.
Missed group member (GS)	These are the records which appear in a group in the ground-truth, and singletons in the prediction.

employs several RL clustering methods and wishes to find the best such method (or the best setting of parameters when using only one linkage method) therefore cannot make a clear decision based on precision and recall only.

To resolve this issue, our proposed method is based on classifying records instead of links for evaluation. Prior to record classification, we find the predicted cluster which best represents each ground-truth cluster. Then each record from a ground-truth cluster which appears in the corresponding best representative predicted cluster is considered a correct classification whereas the other records are considered to be misclassified. While this new evaluation measure is not necessarily a replacement for existing evaluation measures, it avoids the ambiguities caused by measures such as precision and recall.

We now describe our proposed clustering quality evaluation method in detail. We assume that a pairwise similarity graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is generated as described in

Table 8.2: Confusion matrix for the seven categories described in Table 8.1

	True Singleton	**True Group**
Predicted Singleton	SS	GS
Predicted Group	SG	GG_E, GG_M, GG_m, GG_W

Definition 2, by comparing record pairs in a data set $\mathbf{D}$ to be linked. Note that $\mathbf{V}$ denotes the vertices in the graph (all records in $\mathbf{D}$) and $\mathbf{E}$ denotes the set of edges (the similarities calculated between records in $\mathbf{D}$ as explained in Definition 1). After applying a clustering technique on graph $\mathbf{G}$, the predicted clusters contain all the records from $\mathbf{D}$, where each cluster may be a singleton (one record) or a group of two or more records. The ground-truth clusters too may be singletons or contain several records.

We now classify each record in the ground-truth into one of seven categories, based on how they have been clustered by an algorithm. The seven categories are described in Table 8.1[1]. This classification of records into seven categories based on their clustering can also be represented in an error or confusion matrix as shown in Table 8.2, where the vertical columns show the true status of records (if they are a singleton or part of a cluster/group of more than one record), while the rows show the way records are predicted (again as singletons or parts of a group of records).

Identifying records which belong to categories *correct singleton* (SS), *wrongly grouped singleton* (SG), *missed group member* (GS) and *exact group match* (GG_E) is straightforward. It can be accomplished by a single scan over the set of ground-truth clusters and the set of predicted clusters to identify all singletons in either, as well as all exactly matching groups. However, to identify records which belong to categories *majority group match* (GG_M), *minority group match* (GG_m), and *wrongly assigned member* (GG_W), we first require to do a mapping between ground-truth and predicted clusters such that the best representative prediction for a ground-truth cluster is identified. Subsequent to this mapping, we can identify whether each record in the ground-truth cluster appears in the correct predicted cluster or not. The reason for this requirement is that each predicted cluster can only represent one ground-truth cluster but not several. For example, if we consider clustering sibling birth records (children of the same parents), each predicted cluster can only represent the births by one mother and father; it is not possible that two predicted clusters represent the same parents.

The cluster mapping is conducted as follows. For each ground-truth cluster $\mathbf{c}_g \in \mathbf{C}_g$ (which is a group, $|\mathbf{c}_g| > 1$) with no exact match in the prediction, we identify all candidate predicted clusters $\mathbf{C}_{p'} \subseteq \mathbf{C}_p$ in which more than one record from the ground-truth cluster appear. Predicted clusters with more than one record from the ground-truth cluster alone are considered as candidates, because a predicted cluster containing just one record from the ground-truth does not contain a single true link, and is therefore inadequate to become the best representative cluster. For example,

[1]The category naming format indicates whether a record is a singleton or belongs to a group in the ground-truth and prediction respectively (for example SG refers to a Singleton record in the ground-truth allocated to a Group in the prediction).

if the ground-truth cluster $\mathbf{c}_g = \{r_a, r_b, r_c\}$ was split as $\{r_a\}$, $\{r_b\}$, and $\{r_c\}$ in the prediction, it would be incorrect to identify any one of the predicted clusters to be a representation of the ground-truth cluster, because none of the true links (record pairs), (r_a, r_b), (r_b, r_c), or (r_a, r_c) are included in the prediction. Next, we calculate the similarity of each ground-truth cluster $\mathbf{c}_g \in \mathbf{C}_g$, with each candidate predicted cluster $\mathbf{c}_{p'} \in \mathbf{C}_{p'}$, using the Jaccard similarity [35] and the true link similarity which are defined as:

- **Jaccard similarity** ($s_J = |\mathbf{c}_g \cap \mathbf{c}_{p'}|/|\mathbf{c}_g \cup \mathbf{c}_{p'}|$): The ratio between the records common to both the ground-truth and candidate predicted cluster, and the total number of records in the union of the two clusters. The Jaccard similarity always returns a similarity between 0 and 1.

- **True link similarity** ($s_T = |\mathbf{c}_g \cap \mathbf{c}_{p'}|$): The number of records common to both the ground-truth and predicted cluster. This gives a positive integer similarity.

We use Jaccard similarity s_J because of its capability of rewarding the number of records in the candidate predicted cluster which are from the ground-truth cluster, and penalising the records which are missed or added to the wrong predicted cluster. In an instance where the Jaccard similarities are equal for two cluster pairs, we prefer to map the cluster pair with the larger number of records first by selecting the pair with the higher true link similarity, s_T.

Once the similarity is calculated between each ground-truth cluster, $\mathbf{c}_g \in \mathbf{C}_g$, and the candidate predicted clusters $\mathbf{C}_{p'} \subseteq \mathbf{C}_p$, the cluster pairs are sorted in descending order of their similarities. Cluster mapping is done in a greedy manner, where the most similar clusters are mapped first. In case a ground-truth cluster is split equally into several clusters, only one is mapped to the ground-truth cluster. For example, if the ground-truth cluster $\mathbf{c}_g = \{r_a, r_b, r_c, r_d\}$ is split into two predicted clusters $\{r_a, r_b\}$ and $\{r_c, r_d\}$, only one of the two would be mapped to $\{r_a, r_b, r_c, r_d\}$. Once a ground-truth or predicted cluster is mapped, it is removed from the similarity list to ensure we obtain a one-to-one mapping, such that a predicted cluster represents at most one ground-truth cluster.

This process results in finding the best representative cluster for each ground-truth cluster. However, some of the ground-truth clusters may not have a corresponding best match due to complete cluster splitting (each record in the ground-truth cluster appears in a separate cluster in the prediction) or due to a candidate predicted cluster being mapped to a different ground-truth cluster. The greedy algorithm always ensures that a ground-truth cluster is mapped to the largest candidate predicted cluster (cluster with most matches), for as long as the corresponding predicted cluster is not already matched to another ground-truth cluster. For example, a ground-truth cluster $\{r_a, r_b, r_c, r_d, r_e\}$ with predictions $\{r_a, r_b, r_c\}$ and $\{r_d, r_e\}$ is guaranteed to be mapped to the larger predicted cluster $\{r_a, r_b, r_c\}$. However, if we have two ground-truth clusters $\{r_a, r_b, r_c, r_d, r_e\}$ and $\{r_f, r_g, r_h\}$ with a common largest predicted cluster $\{r_a, r_b, r_c, r_f, r_g\}$, this predicted cluster would only be mapped with the ground-truth cluster $\{r_a, r_b, r_c, r_d, r_e\}$ because it has higher cluster similarity.

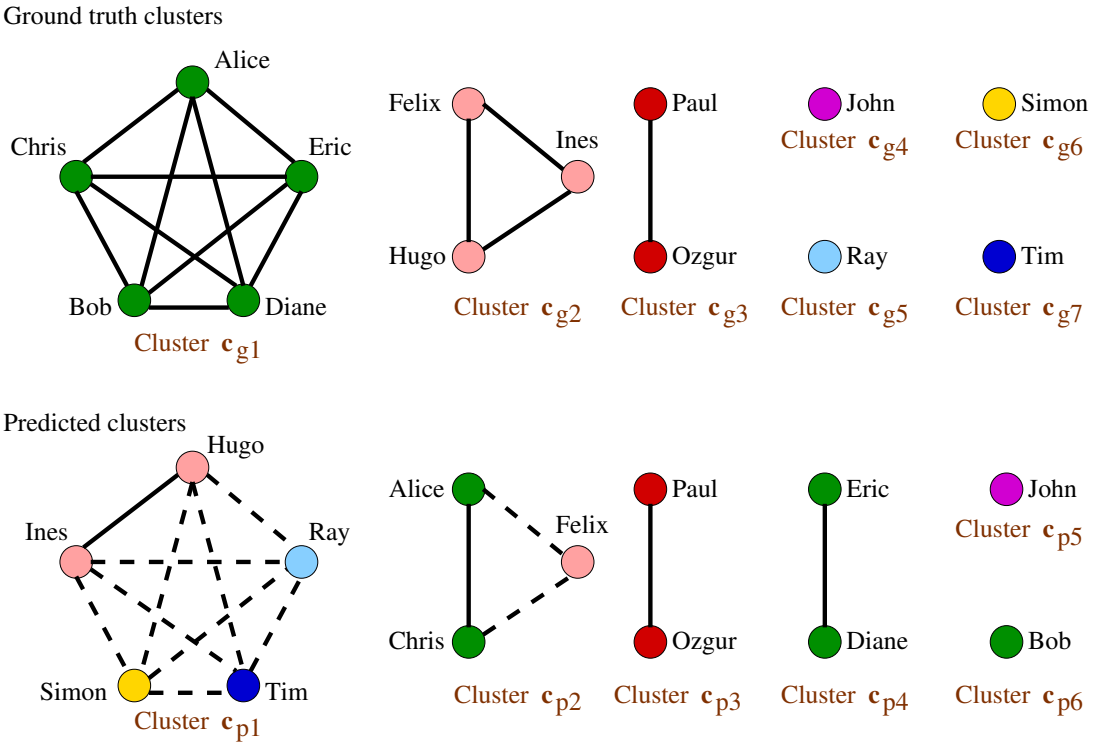

Figure 8.2: An example set of ground-truth clusters (sibling groups) reflecting the births by the same parents, and the linkage result when the corresponding birth records are clustered using a group record linkage technique. Vertices of the same colour identify records belonging to the same group (ground-truth cluster), whereas correct links are identified by solid lines and wrong links by dashed lines.

The best representative cluster is labelled as a majority or minority group match, GG_M or GG_m, based on its record composition, as described in Table 8.1. Once the records belonging to categories GG_M and GG_m are identified, all the records from the ground-truth clusters $\mathbf{c}_g \in \mathbf{C}_g$ which belong to neither of these categories, nor the GS category, are classified into the wrongly assigned members category GG_W (as described in Table 8.1).

8.2.2 Example Cluster Evaluation

We will now illustrate our proposed evaluation measure considering an example bundling (clustering) of sibling birth records (children born to the same parents). A singleton represents an only child in a family. Each child birth record in the clusters predicted by an automated linkage method is classified into one of the seven categories as specified above.

Figure 8.2 shows the birth bundling example, where there are seven ground-truth clusters and six predicted clusters. We will describe the classification of records with respect to each ground-truth cluster and the corresponding predicted clusters

containing the records from the ground-truth cluster. In Figure 8.2, correct links are marked with solid lines, whereas wrong links are shown with dashed lines, and records belonging to a single ground-truth cluster are shown in the same colour.

Let us first consider the ground-truth cluster (sibling group) $\mathbf{c}_{g1}$ containing birth records corresponding to Alice, Bob, Chris, Diane, and Eric in Figure 8.2. Among the predicted clusters, we initially find all the candidate clusters where records Alice, Bob, Chris, Diane, and Eric appear, which are $\mathbf{c}_{p2}$ and $\mathbf{c}_{p4}$ (cluster $\mathbf{c}_{p6}$ does not qualify as a candidate due to having no true links from $\mathbf{c}_{g1}$). Among these, the ground-truth cluster $\mathbf{c}_{g1}$ has the highest similarity ($s_J = 0.4$ and $s_T = 2$) with the predicted cluster $\mathbf{c}_{p4}$. Therefore, we select $\mathbf{c}_{p4}$ as the best representative cluster of the sibling group $\mathbf{c}_{g1}$, and classify records Diane and Eric to be majority group matches (GG_M) since all records in the predicted cluster $\mathbf{c}_{p4}$ come from the ground-truth cluster $\mathbf{c}_{g1}$. Record Bob is classified as a missed group member (GS), whereas records Alice and Chris are classified as wrongly assigned members (GG_W).

Let us now consider the ground-truth cluster $\mathbf{c}_{g2}$ with birth records corresponding to Felix, Hugo, and Ines, as shown in Figure 8.2. It has one candidate predicted cluster $\mathbf{c}_{p1}$ (with $s_J = 0.33$ and $s_T = 2$), whereas cluster $\mathbf{c}_{p2}$ is not considered a candidate due to containing no true links from $\mathbf{c}_{g2}$. Therefore, the predicted cluster $\mathbf{c}_{p1}$ represents the sibling group $\mathbf{c}_{g2}$, and records Hugo and Ines are classified as minority group matches (GG_m) because less than 50% of records in $\mathbf{c}_{p1}$ come from the ground-truth cluster $\mathbf{c}_{g2}$. Record Felix is classified as a wrongly assigned member (GG_W).

As illustrated in Figure 8.2, the ground-truth cluster $\mathbf{c}_{g3}$ with records Paul and Ozgur appears as it is in the prediction as well (cluster $\mathbf{c}_{p3}$). Therefore, both records Paul and Ozgur are classified as exact group matches (GG_E). Furthermore, record John appears as a singleton both in the ground-truth (cluster $\mathbf{c}_{g4}$) and the prediction (cluster $\mathbf{c}_{p5}$). Therefore, this record is classified as a correct singleton (SS). All the other three records corresponding to Ray, Simon, and Tim appear as singletons in the ground-truth (clusters $\mathbf{c}_{g5}$, $\mathbf{c}_{g6}$, and $\mathbf{c}_{g7}$), but were assigned to a group in the prediction (cluster $\mathbf{c}_{p1}$). Therefore, these three records are classified as wrongly grouped singletons (SG).

8.2.3 Area Under the Curve

Most clustering algorithms have a variety of parameters that can be set by users, and based on these parameter settings different clustering results will be generated. One parameter common to most clustering algorithms is the minimum similarity δ_s to consider between records such that the edge between the records is included in the similarity graph $\mathbf{G}$ to be clustered [86, 122, 123, 153]. As a result, for different such similarities (or different other parameter settings), different clustering outcomes for the seven categories described in Table 8.1 will be obtained. Since it is difficult to individually analyse seven different evaluation values for each parameter setting per clustering algorithm, it is worth to explore a method of summarising these values such that the relatively better algorithm can be chosen for a RL application.

The Area Under the Curve (AUC) [84] is a frequently used measure for summarising the linkage quality results over a range of parameter settings in RL algorithms. For example, the Area Under the Precision-Recall Curve (AUPRC), which we described in Section 2.4, summarises the precision and recall values obtained across different parameter settings in a RL algorithm. Similarly, we propose the following approach to calculate the AUC for each of the seven categories. The values obtained for the seven categories are initially normalised such that for a given clustering algorithm, and a given parameter configuration, $f_n(GG_E) + f_n(GG_M) + f_n(GG_m) + f_n(SS) + f_n(GS_W) + f_n(GS) + f_n(SG) = 1.0$, where $f_n()$ is a normalisation function. These normalised proportions of records corresponding to the seven categories are then plotted against the similarity threshold values δ_s, subsequent to which the AUC for each plot is calculated.

For a high quality RL approach, the AUC values of correct singleton (SS), exact group match (GG_E), majority group match (GG_M), and minority group match (GG_m) categories should be higher whereas the AUC values of the other categories should be lower. The differences of such AUC values allow us to describe how much better one clustering technique is over another. We therefore use a simple AUC average calculated as follows, which rewards AUC values for categories SS, GG_E, GG_M, and GG_m and penalises AUC values for the other three categories:

$$AUC_{avr} = \frac{f_a(GG_E) + f_a(GG_M) + f_a(GG_m) + f_a(SS)}{4} - \frac{f_a(GS_W) + f_a(GS) + f_a(SG)}{3}, \quad (8.1)$$

where function $f_a()$ returns the AUC of a given category. Note that even though we have used equal weights for each category in Equation 8.1, it is possible to assign different weights to give more importance to certain types of categorisations (such as assigning a higher weight to GG_E compared to GG_m) depending on the linkage requirement.

8.3 Experimental Evaluation

In this section, we experimentally evaluate our proposed evaluation measure to assess its suitability in the context of group RL. Note that we cannot assess how good or bad this evaluation measure is (just as we cannot determine the quality of existing measures such as precision and recall), but rather aim to show its robustness and lack of ambiguity compared to existing RL evaluation measures.

We assessed our novel proposed evaluation technique by applying it on the linkage results obtained with the three graph clustering methods we proposed in Chapter 4, namely greedy clustering, star clustering, and robust graph clustering. These clustering algorithms were applied on the real-world Isle of Skye (IoS) and synthetic UK birth data sets which we described in Section 2.5, to conduct birth bundling of siblings (linking birth records of children by the same parents). As highlighted in Table 4.1 on page 80, the best linkage results were obtained with the pairwise similarity graph generated considering the attributes *Parent names only* ($\mathbf{G}_N$) for the IoS

data set and the graph considering *All* attributes ($\mathbf{G}_A$) for the UK data set. Therefore, we used the $\mathbf{G}_N$ and $\mathbf{G}_A$ graphs for the IoS and UK data sets respectively, for the experimental results presented in this section. Furthermore, the algorithm specific parameters were set as shown in Table 4.1 for each data set since they produced the best linkage results.

Figure 8.3 shows the precision-recall (PR) curves (where we show the precision and recall values obtained for different similarity thresholds δ_s as described in Section 2.4), the F^*-measures (described in Section 2.4), and the penalised clustering precision (PCPr) values (described in Section 3.5) obtained for the greedy, star and robust graph clustering approaches, for the IoS and UK birth data sets. For the IoS data set, the star and robust graph clustering approaches have clearly performed better than the greedy algorithm as indicated by all three measures. The PR curve for the greedy algorithm indicates a rapid decline in both precision and recall for decreasing similarity thresholds δ_s, which is due to many non-matching record pairs being grouped together and true matches being missed with the vertex selection method (m_n) *Next* (which is the best parameter configuration for the greedy algorithm) at lower similarity thresholds.

In Table 8.3 we show the area under the PR curves (AUPRC) which shows robust graph clustering as the best algorithm. However, as per the F^* and PCPr plots in Figure 8.3, the star clustering algorithm has performed slightly better than the robust graph clustering method which is somewhat contradicting. For the UK data set, the PR plots in Figure 8.3 and the AUPRC values in Table 8.3 indicate that star and robust graph clustering produced better linkage results compared to greedy clustering. However, as per the F^* plot in Figure 8.3, the greedy clustering method has performed better than robust graph clustering for the UK data set, whereas the PCPr plot does not clearly distinguish among the different clustering algorithms' performance. This shows the possibility for ambiguities to occur among various existing evaluation measures.

Figure 8.4 shows the plots for our novel cluster evaluation method for the three clustering techniques. The normalised proportions of the seven categories from Table 8.1 are shown against the similarity threshold δ_s used in each clustering algorithm to filter record pairs. As we described previously, for better clustering results the values of SS, GG_E, GG_M, and GG_m should be higher whereas the values of SG, GS, and GG_W should be lower.

According to Figure 8.4, the proportion of records in the GG_M category is consistently larger than the proportion of records in the GG_W category for both star and robust graph clustering approaches for the IoS data set. The greedy algorithm has a much higher GG_W proportion for the majority of similarity threshold values. Furthermore, the GG_E proportion is relatively higher for star and robust graph clustering compared to the greedy technique. Note that the highest value of GG_E is obtained at similarity threshold 0.95 for all three clustering algorithms, which is complementary to the results shown in Figure 8.3 for the IoS data set. These results show that as per our evaluation method, star and robust graph clustering outperform the greedy method for IoS, which agrees with the conclusion made based on existing cluster

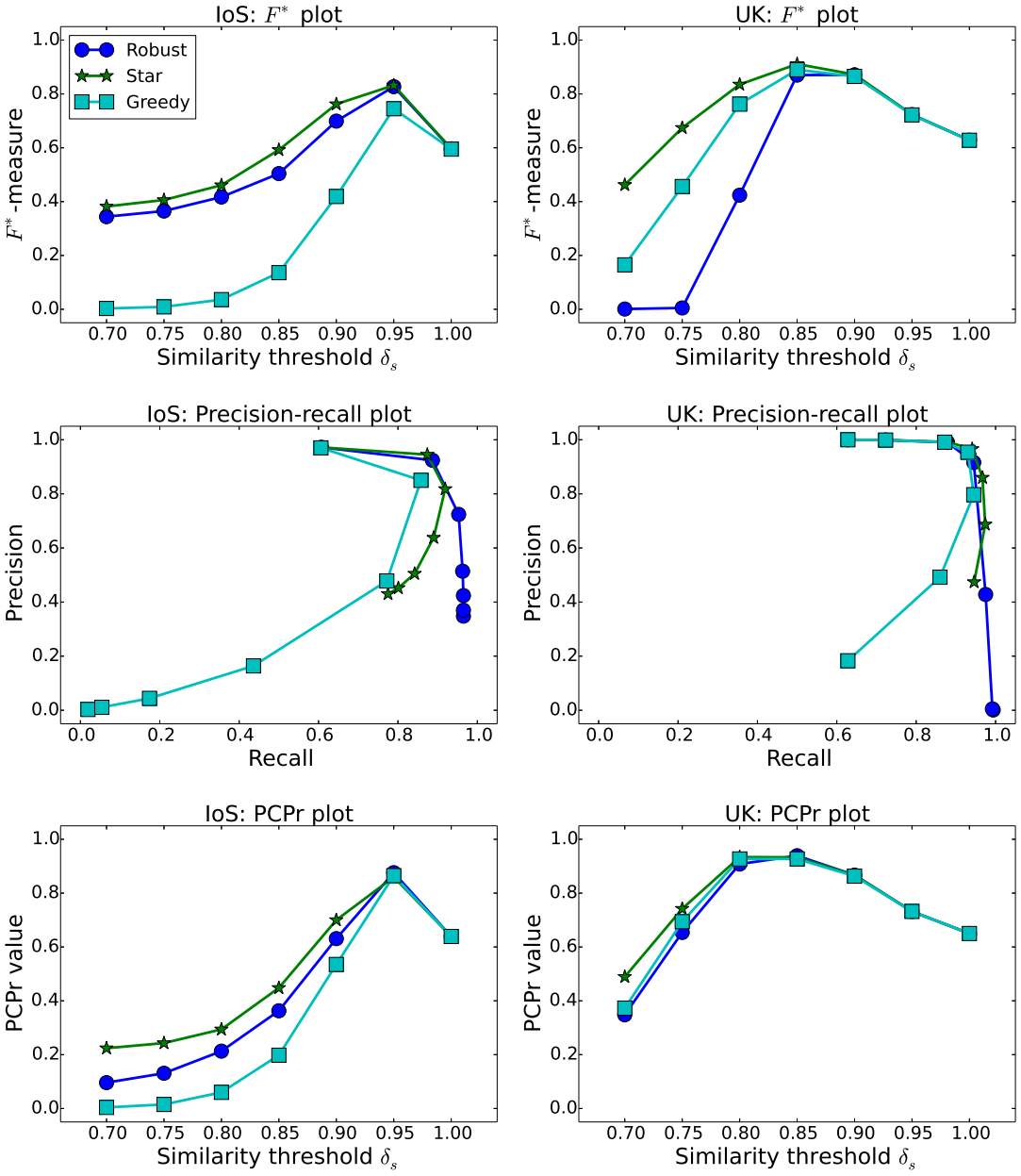

Figure 8.3: The precision-recall (PR), F^*, and penalised clustering precision (PCPr) plots corresponding to the linkage results obtained by conducting record linkage on the IoS (left) and UK (right) birth data sets using the greedy, star, and robust graph clustering approaches.

evaluation measures as shown in Figure 8.3. For the UK data set, the novel evaluation plots shown in Figure 8.4 have only minor variations across the three algorithms. The most noteworthy difference is that robust graph clustering is producing more incorrect clusters (GG_W) followed by greedy clustering for lower thresholds.

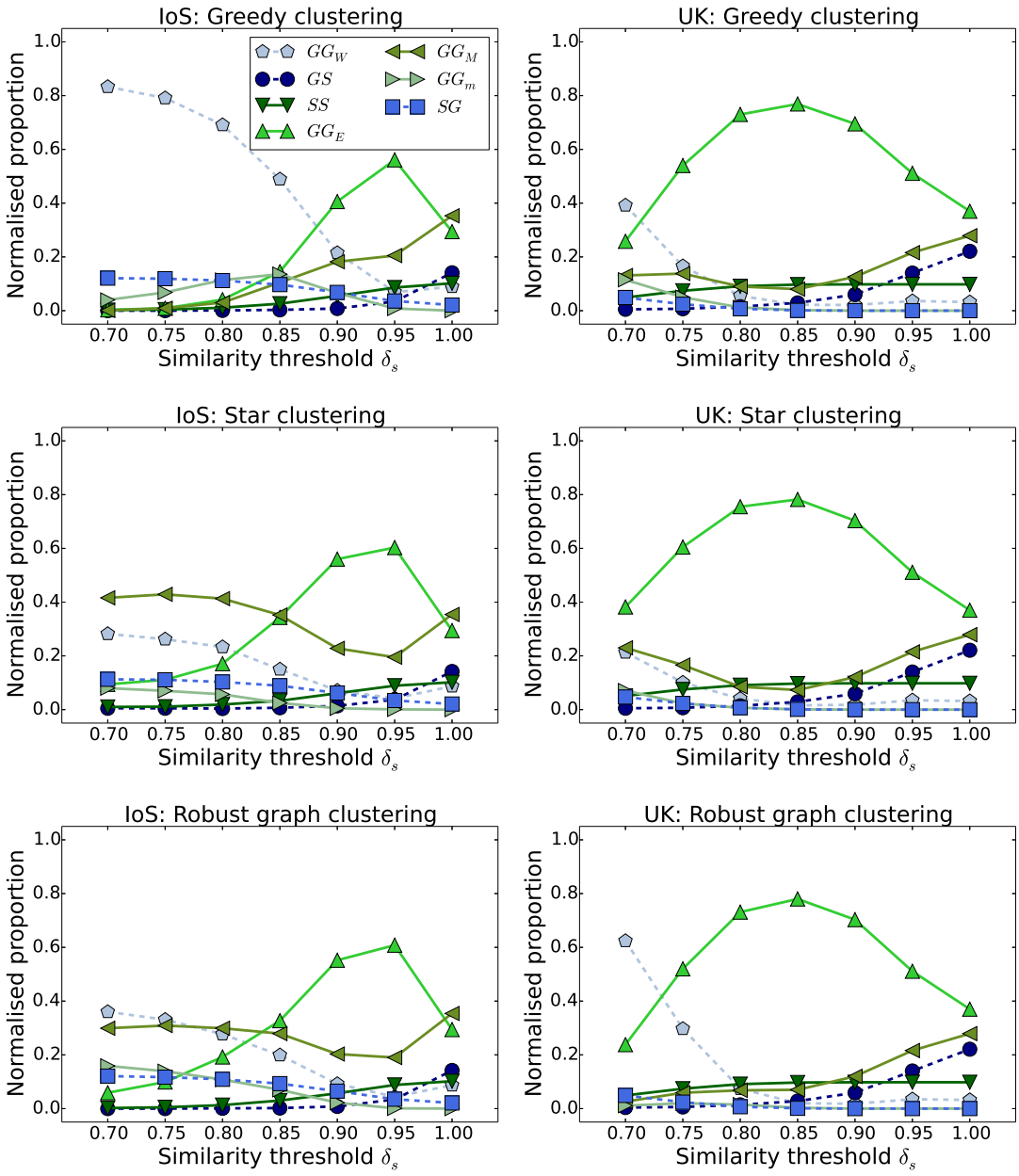

Figure 8.4: Plots for new evaluation measure corresponding to the linkage results obtained by conducting record linkage on the IoS (left) and UK (right) birth data sets using the greedy, star, and robust graph clustering approaches.

We further compare the new evaluation results with existing evaluation results in Table 8.3, where we present the area under the curve (AUC) of PR (based on the PR curves in Figure 8.3), the AUC of our new cluster evaluation plots (based on the plots in Figure 8.4) and a simple averaging of the AUC values across the seven categories AUC_{avr} as defined in Equation 8.1. For the IoS data set, the best clustering algorithm

Table 8.3: Area under the curve (AUC) values for the three clustering techniques with the best value(s) highlighted in each column per each data set.

Data set	Algorithm	Area under the curve (AUC)								Average
		AUPRC	GG_E	GG_M	GG_m	SS	GG_W	GS	SG	(AUC_{avr})
IoS	Greedy	0.69	0.22	0.12	**0.07**	0.04	0.45	0.02	0.08	-0.071
	Star	0.81	**0.33**	**0.33**	0.03	0.04	**0.16**	0.02	0.08	**0.096**
	Robust	**0.90**	**0.33**	0.27	**0.07**	0.04	0.19	0.02	0.08	0.081
UK	Greedy	0.85	0.59	0.14	**0.02**	0.09	0.09	0.06	0.01	0.157
	Star	**0.95**	**0.62**	**0.15**	0.01	0.09	**0.06**	0.06	0.01	**0.174**
	Robust	**0.95**	0.59	0.11	0.01	0.09	0.13	0.06	0.01	0.133

as per our evaluation measure AUC_{avr} is star, followed by robust graph clustering. This conclusion complements the F^* and PCPr values shown in Figure 8.3 but slightly varies from the assessment made based on the AUPRC values which shows robust graph clustering as the best performer.

For the UK data set, the star algorithm is again shown as the best performer followed by the greedy clustering approach as per our evaluation measure AUC_{avr}. While this outcome is again consistent with the assessment made based on the F^*-measure, it is different from the conclusion made based on the AUPRC values which show the robust graph clustering algorithm to be better than the greedy approach. This shows that our novel evaluation approach complements the F^*-measure but can sometimes produce different results to AUPRC. Furthermore, as opposed to existing evaluation measures, our proposed evaluation approach clearly shows which algorithms have generated the largest number of correct record groupings.

8.4 Summary

In this chapter, we have presented a novel evaluation measure for assessing group RL techniques. We showed how existing evaluation measures such as precision and recall can produce ambiguous results in a group RL context due to considering record pairs in the evaluation rather than considering the correct assignment of records into groups. Our proposed evaluation measure mitigates this issue by classifying records into seven categories which reflects how correctly records have been clustered by a group RL algorithm compared to the ground-truth clusters. We conducted experiments to compare our novel evaluation measure with existing evaluation measures.

The empirical evaluation was performed based on the group RL results obtained by applying three clustering algorithms on one real-world and one synthetic data set. The assessment of these linkage results using our evaluation method was consistent with the assessment made based on the F^*-measure. Furthermore, unlike with precision and recall values, the proposed method does not provide ambiguous results. That is, the proposed method ensures that identical evaluation results are obtained for two different linkage approaches, if and only if they both have generated identical cluster predictions.

In the next chapter, we address a different issue pertaining to RL which is the privacy concerns associated with publishing linked data sets. We propose a novel graph anonymisation technique which can be used in the RL context to ensure the privacy of linked data sets without compromising the interpretability of linkage results.

Chapter 9

Graph Data Anonymisation for Record Linkage

As we discussed in Section 1.2, ensuring the privacy of sensitive linked data is crucial in Record Linkage (RL) applications. Even though Privacy-Preserving Record Linkage (PPRL) is beyond the scope of this thesis (as we highlighted in Section 1.5), since we aim to link population data which often contain sensitive information about people, it is still important to explore alternative methods of anonymising linked population data. In this chapter, we propose a novel method of anonymising graph data which is applicable in the RL context. Our proposed anonymisation technique can be employed not only in RL applications but in other domains where data can be represented using graphs.

In Section 9.1 we provide a concise introduction to the topic of graph anonymisation and highlight the necessity for developing novel anonymisation techniques in application domains such as RL. Next, in Section 9.2, we describe our proposed graph anonymisation technique in detail, whereas in Section 9.3 we provide a brief overview of the our web tool which is based on the graph anonymisation technique proposed in this chapter. In Section 9.4 we then evaluate our anonymisation method using real-world and synthetic population data, and finally in Section 9.5, we conclude this chapter with a summary of our findings.

9.1 Introduction

Representing databases as graphs is often necessary in modern data analysis tasks due to many databases having inter-relationships among their records [24]. For example, a social network data set would represent individuals as vertices and their relationships as edges, whereas in the RL context, several census data sets may be connected with edges to show the records that potentially belong to members of the same family (such as siblings) [43]. The types of data which require a graph representation are often sensitive (such as population data that represent real people) and therefore cannot be shared publicly [96]. This requires the anonymisation of a graph such that sensitive data cannot be re-identified, while the structure of the graph, the relationships between vertices and their attributes, are being preserved.

As we discussed in Section 3.6, anonymisation of graph data is a topic that has been explored by several previous studies [63, 174]. However, these studies focus on protecting a data set against privacy attacks [46, 188] and therefore often compromise the structure of the original graph by removing or adding edges and/or vertices that are vulnerable to re-identification due to their unique characteristics. Furthermore, the data sets resulting from existing anonymisation techniques do not necessarily have to be interpretable by humans. Rather, the aim of these techniques is to anonymise a graph such that identifying the real-world entities represented by vertices in the graph is made difficult, while it is still possible to conduct analysis on the anonymised graph with suitable machine learning algorithms [46, 188].

Therefore, existing graph anonymisation tools are not useful in generating an anonymised, human interpretable version of a given sensitive graph data set. Such an anonymised human interpretable data set is however important to allow inspection of the data set in the context of transparency of how a machine learning algorithm performs on that data set, or to allow a data set to be published for educational purposes. Furthermore, compromising the graph structure is particularly detrimental in RL applications since the structure of the graph (reflected by the edge weights and the neighbourhood of vertices) is often used to determine the likelihood for record pairs to refer to the same entity [148, 154].

In this paper, we propose a novel graph anonymisation technique, and present our related web tool DOYEN[1] (**D**ata an**O**n**Y**miser for s**EN**sitive Graph Data) which is based on our anonymisation method. This method generates an anonymised version of a sensitive graph data set while maintaining its graph structure by replacing the sensitive attribute values of vertices with values from a public lookup table using a cluster-based mapping technique. The initial implementation of the DOYEN tool anonymises family data where graph connectivity represents sibling relationships. Such family data is required for social science studies and for the reconstruction of (historical) populations [17]. We anticipate that our graph anonymisation method would be instrumental in mitigating hindrances to such research work due to the inability of publishing sensitive graph data. In the next section we describe the steps in our anonymisation approach.

9.2 Mapping-based Graph Data Anonymisation

Our proposed technique anonymises a given sensitive input graph data set by replacing (mapping) sensitive attribute values with values from a public lookup table. It also maintains the structure of the graph and the similarities between its vertices by conducting attribute value replacement in a way that preserves the similarities between vertices.

[1] https://dmm.anu.edu.au/doyen/

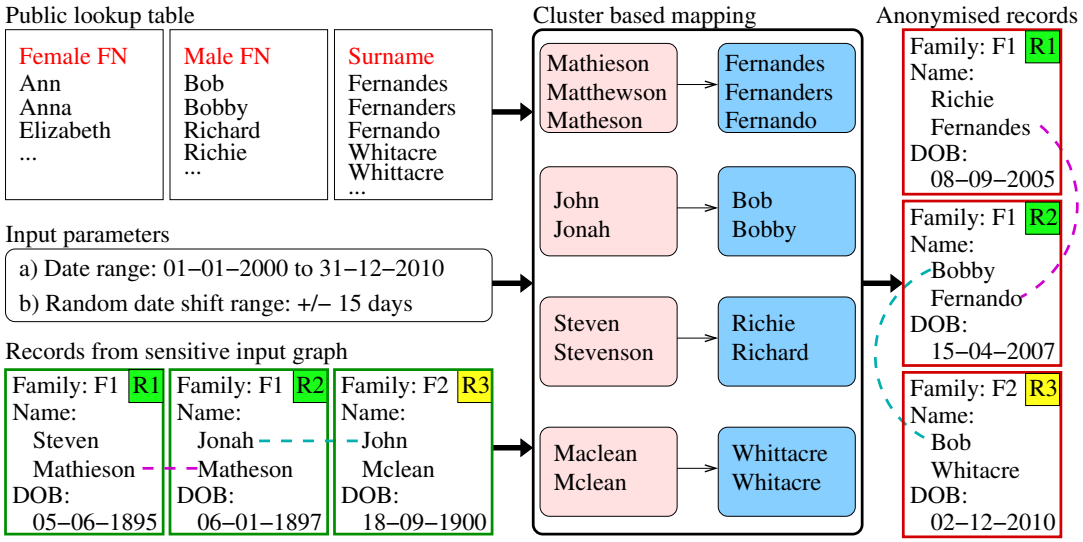

Figure 9.1: Overview of our anonymisation technique.

9.2.1 Method Overview

Figure 9.1 provides a high level overview of our anonymisation method. The sensitive graph data set and a lookup table with attribute values extracted from a public data source are given as input to this technique. As an example we show a family graph data set where siblings have the same family identifier and colour, and highly similar first name and last name pairs (where similarity is calculated with an approximate string similarity function [128] as we described in Section 2.3) are shown as edges (dashed lines). Our approach first clusters the sensitive attribute values from the input graph data set and a public lookup table separately, and then maps the generated clusters of sensitive values to the clusters generated from a lookup table using a mapping approach as we describe in Section 9.2.2.

For each vertex in the input graph data set an anonymised vertex is then generated by replacing each sensitive attribute value with the corresponding mapped value from a public lookup table. If the graph data contains date values, as shown in our example in Figure 9.1, they are anonymised by shifting dates within a user specified range as we discuss in Section 9.2.3.

9.2.2 Cluster-based Attribute Value Mapping and Anonymisation

We now describe our anonymisation approach for sensitive attribute values. Assume we have a sensitive input data set $\mathbf{D}_s$ containing records $r \in \mathbf{D}_s$ that represent entities, and an external lookup table $\mathcal{T}_l$ of attribute values. The data set $\mathbf{D}_s$ can be represented as a graph $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$, where a vertex in $\mathbf{V}_s$ represents a record $r_i \in \mathbf{D}_s$, and an edge in $\mathbf{E}_s$ corresponds to the pairwise attribute similarity of the record pair (r_i, r_j). Such similarities, as calculated by comparing attribute values (as

Algorithm 9: Cluster-based attribute value mapping

Input: $\mathbf{D}_s$ - Sensitive input data set
$\mathbf{A}$ - List of sensitive attributes
$\mathcal{T}_l$ - Attribute value lookup table
w - Weight assigned to attribute value similarities in cluster comparison
Output: $\mathcal{T}_m$ - Attribute value mapping table

1 $\mathcal{T}_m = \{\}$ // Initialise empty attribute value mapping table
2 **for** $a_i \in \mathbf{A}$ **do** // Iterate over sensitive attributes
3 $\quad \mathbf{C}_s = \textbf{GetClusters}(\mathbf{D}_s.\mathbf{a}_i)$ // Cluster sensitive input attribute values
4 $\quad \mathbf{C}_l = \textbf{GetClusters}(\mathcal{T}_l.\mathbf{a}_i)$ // Cluster lookup attribute values
5 $\quad$ **for** $\mathbf{c}_j \in \mathbf{C}_s$ **do** // Iterate over sensitive attribute value clusters
6 $\quad\quad \mathbf{c}'_j = \textbf{BestMatch}(\mathbf{c}_j.\mathbf{s}, \mathbf{c}_j.\mathbf{l}, |\mathbf{c}_j.\mathbf{v}|, \mathbf{C}_l, w)$ // Find best match
7 $\quad\quad \mathbf{C}_l.remove(\mathbf{c}'_j)$ // Remove selected cluster from lookup clusters
8 $\quad\quad \textbf{MapValues}(\mathcal{T}_m, \mathbf{c}_j.\mathbf{v}, \mathbf{c}'_j.\mathbf{v})$ // Map attribute values in clusters
9 **return** $\mathcal{T}_m$

we described in Definition 1 on page 19), are often used to show the strength or importance of relationships in graph data [180]. We refer to the list of sensitive attributes in $\mathbf{D}_s$ as $\mathbf{A} = \{a_1, \ldots, a_n\}$, and the values from each sensitive attribute a_i in the input data set and the lookup table as $\mathbf{D}_s.\mathbf{a}_i$ and $\mathcal{T}_l.\mathbf{a}_i$, respectively.

Assuming that the sensitive attributes $\mathbf{A}$ have been used to calculate the pairwise similarities between records in $\mathbf{D}_s$, we need to ensure that these similarities are maintained in the anonymised data set we generate. This means that we need to retain the similarity structure of $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$ in the generated anonymised graph $\mathbf{G}_a = (\mathbf{V}_a, \mathbf{E}_a)$ which represents the anonymised data set $\mathbf{D}_a$. To achieve this goal, we use a one-to-one cluster mapping approach where we map an attribute value cluster from the sensitive input data set $\mathbf{D}_s$ to an attribute value cluster from the public lookup table $\mathcal{T}_l$ such that the intra-cluster similarities are highly similar across the two clusters.

Algorithm 9 outlines the initial steps of our anonymisation technique, where we map attribute values in the sensitive input data set $\mathbf{D}_s$ to attribute values in a public lookup table $\mathcal{T}_l$ using a clustering-based approach. The input to the algorithm includes the sensitive graph data set $\mathbf{D}_s$, the list of sensitive attributes $\mathbf{A}$ (such as names and addresses of people), a public lookup table $\mathcal{T}_l$ which contain values that attributes $a_i \in \mathbf{A}$ could assume, as taken from an external publicly available source, and a weight w ($0 \leq w \leq 1$) to assign in the cluster comparison.

The algorithm starts by initialising an empty table $\mathcal{T}_m$ to hold the final attribute value mappings. In lines 2 to 4, the algorithm iterates over the sensitive attributes $a_i \in \mathbf{A}$, and clusters the corresponding attribute values in the sensitive input data set $\mathbf{D}_s.\mathbf{a}_i$ and the lookup table $\mathcal{T}_l.\mathbf{a}_i$. Next, in lines 5 and 6, we iterate over the sensitive attribute value clusters $\mathbf{c}_j \in \mathbf{C}_s$ from the input data set $\mathbf{D}_s$ and find the best matching attribute value cluster from the lookup attribute value clusters $\mathbf{C}_l$. For a given sensitive attribute value cluster $\mathbf{c}_j$, we obtain its sorted sensitive attribute

values $\mathbf{c}_j.\mathbf{v} = [v_1, v_2, \ldots v_m]$, the vector of pairwise similarities of attribute values in cluster $\mathbf{c}_j.\mathbf{s} = [s_{v_1,v_2}, \ldots, s_{v_1,v_m}, s_{v_2,v_3}, \ldots, s_{v_{m-1},v_m}]$, and the attribute value length vector $\mathbf{c}_j.\mathbf{l} = [|v_1|, |v_2|, \ldots, |v_m|]$. The best matching cluster for $\mathbf{c}_j$ is identified with the function **BestMatch**(), which takes as input the pairwise similarities vector $\mathbf{c}_j.\mathbf{s}$, the vector of attribute value lengths $\mathbf{c}_j.\mathbf{l}$, the number of attribute values in cluster $|\mathbf{c}_j.\mathbf{v}|$, the set of lookup attribute value clusters $\mathbf{C}_l$, and a weight w. The function **BestMatch**() finds the most similar lookup cluster $\mathbf{c}'_j$ to the sensitive attribute value cluster $\mathbf{c}_j$ from the set of lookup clusters $\mathbf{C}_l$ (which are of the same size as $\mathbf{c}_j$) using Euclidean distances [35] calculated between their similarity vectors and their attribute value length vectors. A weight of w and $1 - w$ is assigned to the similarity vectors and the length vectors, respectively, in the Euclidean distance calculation. The cluster $\mathbf{c}'_j \in \mathbf{C}_l$ with the minimal weighted Euclidean distance between similarity and length vectors is selected to be mapped to cluster $\mathbf{c}_j$. If $\mathbf{C}_l$ does not contain clusters of size $|\mathbf{c}_j.\mathbf{v}|$ then subsets of clusters from $\mathbf{C}_l$ which are larger than $|\mathbf{c}_j.\mathbf{v}|$ are considered.

In line 7, we remove the selected cluster $\mathbf{c}'_j$ from $\mathbf{C}_l$ to obtain unique cluster mappings. In line 8, we then map the sorted attribute values in $\mathbf{c}_j.\mathbf{v}$ to the corresponding values in cluster $\mathbf{c}'_j.\mathbf{v}$, such that each attribute value from data set $\mathbf{D}_s$ has a unique mapping to a value in the lookup table $\mathcal{T}_l$. These unique mappings are stored in an attribute value mapping table $\mathcal{T}_m$ which we return in line 9.

In the second step of our anonymisation technique, an anonymised graph data set $\mathbf{D}_a$ is generated for the sensitive input graph data set $\mathbf{D}_s$ in the following manner.

For each record from the input data set $r_i \in \mathbf{D}_s$, we create a synthetic record $r'_i \in \mathbf{D}_a$, and for each sensitive attribute value in r_i, we replace the original attribute value with the corresponding mapped attribute value from $\mathcal{T}_m$.

Complexity analysis: We now describe the complexity of our proposed cluster-based attribute value mapping algorithm. In line 2 of Algorithm 9, the for loop executes $|\mathbf{A}|$ times. Out of the cluster generation steps in lines 3 and 4, the time complexity of generating connected component clusters for lookup tables $\mathcal{T}_l$ is dominant, given that $\mathcal{T}_l$ is much larger than the sensitive input data set $\mathbf{D}_s$. Therefore, considering only line 4, the time complexity of cluster generation is $O((\mathcal{T}_l.\mathbf{a}_i)^2)$, which is the total number of edges across unique attribute values for a given attribute $\mathbf{a}_i \in \mathbf{A}$. The for loop in lines 5 to 8 has a maximum time complexity of $O(|\mathbf{C}_s| \cdot |\mathbf{C}_l)|$ since it searches for the best matching pairs among all sensitive and lookup table attribute value cluster pairs. Therefore, the overall time complexity of Algorithm 9 is $O(|\mathbf{A}| \cdot ((\mathcal{T}_l.\mathbf{a}_i)^2 + |\mathbf{C}_s| \cdot |\mathbf{C}_l)|))$.

9.2.3 Generating Anonymised Date Values

Given that many data sets have dates associated with their records (such as dates of birth or dates of hospital admission), our method also provides an anonymisation approach for date values while maintaining the temporal distances across connected records. Prior to date anonymisation, the records in $\mathbf{D}_s$ are grouped such that related records are contained in a single group. These groupings reflect records that represent a related group of entities, such as the siblings of the same family. Subsequent to

grouping records, our anonymisation method sorts the date values associated with the records in a group.

Our method allows the user to define a minimum (d_{min}) and a maximum (d_{max}) date range within which they want the earliest date value from a specific group to appear. Thus, for the earliest date d_1 from a record group, we create a corresponding date d'_1 where $d_{min} \leq d'_1 \leq d_{max}$. Then, the remaining date values in the record group are shifted by $d'_1 - d_1$ days and each newly generated date value, excluding the earliest date d'_1, is randomly shifted by $\pm\Delta d$ days such that any temporal constraints across the generated date values are maintained. For example, if $\mathbf{D}_s$ contains birth records of sibling groups, then the temporal constraints of the birth dates would reflect that it is not possible for two births by the same mother to be less than nine months apart unless they are twins [123]. Let us consider the following example for further clarification. Assume we have a sibling group of three children with birth dates $d_1 = 01/01/1991$, $d_2 = 23/12/1991$, and $d_3 = 05/10/1994$. Given $d_{min} = 01/01/2000$ and $d_{max} = 01/01/2001$, assume we randomly generated an anonymised date for d_1 within the $[d_{min}, d_{max}]$ range as $d'_1 = 09/07/2000$, which means that the original value has been shifted by $3,477$ days (i.e. $d'_1 - d_1 = 3,477$). We therefore shift dates d_2 and d_3 by $3,477$ days to obtain $d'_2 = 30/06/2001$ and $d'_3 = 12/04/2004$. Assuming $\pm\Delta d = 10$, we obtain the final anonymised date values $d'_1 = 09/07/2000$, $d'_2 = 20/06/2001$ (by subtracting 10 days), and $d'_3 = 22/04/2004$ (by adding 10 days). We use these anonymised date values in the synthetic records $r'_i \in \mathbf{D}_a$.

9.3 Web Tool Demonstration

We developed a web tool named DOYEN[2] which is based on the anonymisation technique presented in this chapter. The initial implementation of the DOYEN tool demonstrates sensitive graph data anonymisation using two synthetic input data sets which we generated based on two real-world historical birth data sets. These example data sets contain name and address variations to help demonstrate the capability of DOYEN to anonymise a graph while maintaining its similarity structure. Furthermore, they contain several twin births as well as missing values for the last names of fathers and children, as seen in the original birth data sets. A lookup table containing values for the sensitive attributes first name, last name, and address, was generated using the publicly available North Carolina Voter Registration (NCVR) data set (which we described in Section 2.5) as well as Australian addresses. In this section, we provide a brief overview of the DOYEN tool's interface.

Figure 9.2 shows the input screen of the DOYEN web tool. The two buttons *Example 1* and *Example 2* will load one of the input graph data sets and suitable parameter values. A sample of the input data set can be viewed by clicking on the *View Input Data Sample* button, whereas the full data set can be downloaded as well. The four parameters to be specified are:

[2] https://dmm.anu.edu.au/doyen/

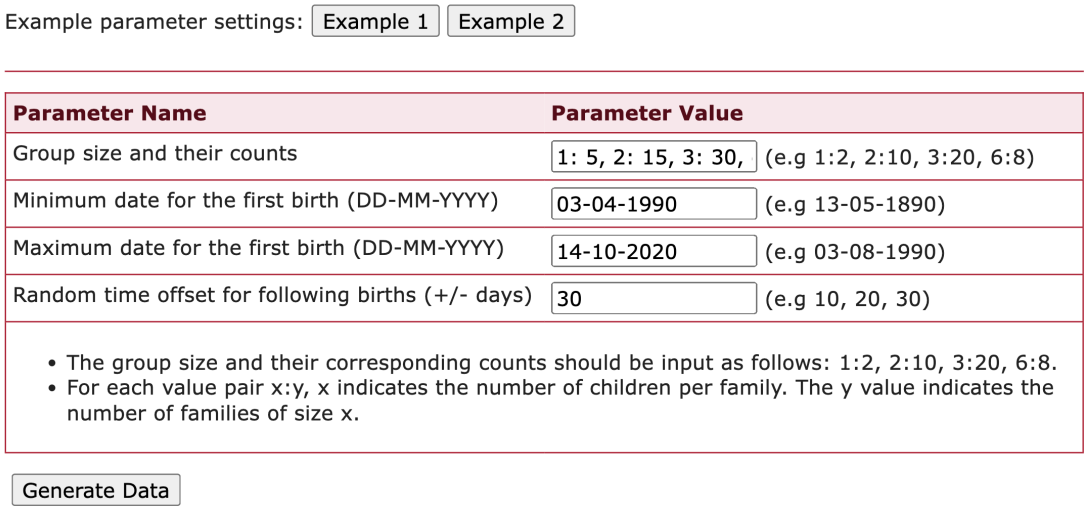

Figure 9.2: Input screen of the DOYEN web tool.

- **Group size and their counts**: This parameter allows control of the number of families of a given size that are to be generated. When a sensitive graph data set is loaded, this field is automatically populated with the family size distribution of the loaded input data set. The user can change the values as desired. However, the current implementation restricts the user to specifying only up to a maximum number of families as appearing in the input data set, for a given family size. That is, if there are ten families (clusters) of size five in the input data set, then currently the user can only generate a maximum of ten families of size five.

- **Minimum / Maximum dates for the first birth**: For each family in the anonymised data set, the first birth record in the family is expected to have a birth date within (including) the given minimum and maximum date range (d_{min} and d_{max}), where $d_{min} < d_{max}$ as we described in Section 9.2.3.

- **Random time offset**: This is the $\pm\Delta d$ time range which is used to further shift (randomly perturb) the dates of birth in each anonymised synthetic family (except the first date) as we also described in Section 9.2.3.

The user has the flexibility of editing the values with which the parameter fields are automatically populated after an example data set has been loaded. When the *Generate Data* button is clicked, the anonymisation and data generation process is executed in the back end. The tool internally calculates the string similarity of attribute values by first applying a blocking technique [35]. We use Soundex encoding for names and Locality Sensitive Hashing for addresses, as we discussed in Section 2.3.2. We then apply a pairwise string similarity calculation method, where we use Jaro-Winkler for names and the Dice coefficient based approach for addresses which we discussed in Section 2.3.3 [128].

Sample sensitive input data

Rec ID	Fam ID	Birth date	Child FN	Child LN	Mother FN	Mother LN	Father FN	Father LN	Address	Gender
R8	F7	10/1/1762	macdiarmid	mcmillan	pegy	mcmillan	lauchlin	mcmillan	monkstadt	m
R9	F7	5/11/1764	duncan	mcmillan	pegy	mcmillan	lauchlin	mcmillan	monkstodt	m
R10	F8	26/10/1766	murdo	ohare	frisken	macdougal	forbes	ohare	parish schoolhouse	m
R11	F8	14/9/1768	willie		frisken	macdougal			parish schoolhouse	m
R12	F9	20/10/1782	olive	mcnair	sibbilla	mcnair	armadale	mcnair	dunalister cottage	f
R13	F9	12/10/1783	mate	mcnair	sibbilla marjorie	mcnair	armadale	mcnair	dunalister cottage	f
R14	F10	21/7/1764	millan	richmond	josephina	richmond	mitchel	richmond	stein waternish	m
R15	F10	11/5/1766	jesse	richmond	josephin	richmond	mitchell macphie	richmond	stein waternish	f
R16	F11	10/10/1789	angus	macinnnes	marrion	macinnnes	betsy	macinnnes	culnaknock	m
R17	F11	2/9/1791	angus	macinnnes	marrion	macinnnes	betsy	macinnnes	culmaknock	m

Sample generated anonymised data

Rec ID	Fam ID	Birth date	Child FN	Child LN	Mother FN	Mother LN	Father FN	Father LN	Address	Gender
R8	F7	29/10/1999	orpheus	murdock	wren	murdock	mustapha	murdock	naremburе	m
R9	F7	12/9/2002	fayes	murdock	wren	murdock	mustapha	murdock	naremburn	m
R10	F8	18/9/1990	geeta	ergle	irvin	leadbeter	baye	ergle	sydney uni	m
R11	F8	27/8/1992	quill		irvin	leadbeter			sydney uni	m
R12	F9	16/8/2018	theldora	meyreles	anthosha	meyreles	ronreicus	meyreles	coranba	f
R13	F9	2/9/2019	sudhaben	meyreles	anthosha quantiara	meyreles	ronreicus	meyreles	coranba	f
R14	F10	27/10/2006	mulchandbhai	boufedji	nastasia	boufedji	gartrel	boufedji	st ruth	m
R15	F10	7/8/2008	omara	boufedji	nastasha	boufedji	gartrell trossie	boufedji	st ruth	f
R16	F11	8/11/2007	aliniswe	tagliatela	urshula	tagliatela	jamesrobert	tagliatela	tubbumurra	m
R17	F11	20/10/2009	aliniswe	tagliatela	urshula	tagliatela	jamesrobert	tagliatela	tubbamurra	m

Figure 9.3: Sample of the synthetic sensitive input data set and the anonymised data set as generated by the DOYEN web tool.

Subsequently, the attribute value clusters are identified with the components clustering approach [86] (discussed in Section 3.1) with a similarity threshold of 0.8 [35] which we set after conducting experiments with different similarity thresholds. Next, all attribute value clusters from the input graph are mapped to clusters from the lookup table using the Euclidean distance vector similarity measure, as we described in Algorithm 9. Since the vector of similarities between attribute value pairs in clusters ($\mathbf{c}_j.\mathbf{s}$) is more important than the vector of attribute value lengths ($\mathbf{c}_j.\mathbf{l}$), we assign a relatively higher weight w ($w > 0.5$) to $\mathbf{c}_j.\mathbf{s}$ and a weight of $1 - w$ to $\mathbf{c}_j.\mathbf{l}$ when cal-

culating the overall Euclidean distance between sensitive and lookup attribute value clusters.

After the attribute value mapping is completed, the DOYEN tool generates the anonymised graph data set by replacing the attribute values of each record in the input data set with the corresponding mapped attribute values, and by shifting the dates of birth as we described in Section 9.2.3. Once the anonymised data is generated, the user can view a sample of the attribute value mappings. Furthermore, the user can view a sample of the anonymised data set created by DOYEN, or download the full data set. Figure 9.3 shows a sample of the sensitive input data set and the corresponding anonymised records as generated by DOYEN. Notice how, for example, the address values 'monkstadt' and 'monkstodt' (in R8 and R9) with a high pairwise string similarity have been replaced with values 'narembure' and 'naremburn' which have a similar pairwise similarity.

9.4 Evaluation

In this section we present the results we obtained by applying our graph anonymisation technique on the real-world Isle of Skye (IoS) and the synthetic UK birth data sets which we discussed in Section 2.5. We set the parameter configurations of our algorithm as specified in the previous section. After conducting an initial set of experiments, we assigned weights $w = 0.75$ to the vector of similarities $\mathbf{c}_j.\mathbf{s}$, and $w = 0.25$ to the length vector $\mathbf{c}_j.\mathbf{l}$ in the cluster pair comparison.

Table 9.1 shows how our anonymisation method maps sensitive attribute values (from the IoS and UK data sets) to public attribute values from a lookup table, such that the pairwise similarities and attribute value lengths are maintained. For example, if we consider the female (F) first name mappings for the IoS data set, notice how the names 'kate' and 'katew' with shorter string lengths have been mapped to similarly short string values 'verla' and 'verle', whereas longer names 'catharine' and 'catherene' are mapped to relatively longer public attribute values 'nicholett' and 'nicholette' respectively, while maintaining pairwise similarities.

To illustrate the quality of the anonymised graphs generated by our anonymisation technique, Figure 9.4 shows the distribution of the pairwise similarity for record pairs from the IoS and UK data sets. We have used a randomly sampled subset of 1 million record pairs including all true matching pairs from each data set for plotting. For each record pair, the similarity is calculated using the Jaro-Winkler similarity measure on names and the Dice coefficient similarity measure on addresses [128] (described in Section 2.3.3), followed by an averaging of these values. As can be seen from this figure, the similarity distribution of both the sensitive input values and the generated anonymised values are highly similar for both the IoS and UK data sets. This shows the capability of our anonymisation method to anonymise sensitive graph data, while maintaining its structure, as reflected by these pairwise similarities.

Table 9.1: Sample of sensitive to public female (F) first name, male (M) first name, last name, and address attribute value mappings conducted by our graph data anonymisation technique.

		Sensitive attribute values	**Mapped public attribute values**
IoS	**First name (F)**	kate, katew, katie, kattie, katty, katy, keith, ketie, ketty, kitty	verla, verle, verlee, verley, verlie, verly, viral, viril, virl, virlee
		marian, marion, maron, marrion, mary-ann, mary-anna, mary-anne, maryann, mearon	gevaun, gevonne, gevony, giovana, giovanna, giovanni, giovannia, giovannie, gvonnia
		catharine, catherene, catherin, catherine, catherinw, cathrine	nicholett, nicholette, nicoletta, nicolette, nicollette, nikoletta
	First name (M)	john, johnie, johnina, johnnie, johny, joney	vasil, vasile, vasili, vasiliy, vasily, vasyl
		lachlaen, lachlain, lachlan, lachlean, lachlen, lachlin, lauchlan, lauchlen, lauchlin	octave, octavia, octavian, octaviano, octavio, octavion, octavious, octavis, octavius
		torquel, torquell, torquhol, torquihol, torquil, torquill	jefferies, jefferson, jeffres, jeffreyj, jeffreys, jeffries
	Last name	macbride, maciver, macpherson, mciver, mcpherson	maitland, mathlin, matilainen, mattlin, matulonis
		macinnes, macinnews, macinnis, macinnnes, mackenzie, mcinnes, mcinnis, mcinnnes, mckenzie	gallimore, galmore, gillmer, gillmore, gilmartin, gilmer, gilmore, gilmour, gollmar
		macgillivray, macgillvray, macgilvray, mcgillivray, mcgillvray	eisenbarth, eisenbath, eisenbeis, eisenbraun, eisenhofer
	Address	monkstadt, monkstodd, monkstodt, monstodd	mac gregor, macgregor, mc gregor, mcgregor
		garrylapin, garylapin, gerrylapen, gerrylapin, gerycapin, gerylapen, gerylapin	stathfield, strahtfield, straithfield, stratfield, strathfie, strathfield, strathield
		aird bernisdale, airdbernisdale, aridbernisdale	o halloran hill, o'halloran hill, ohalloran hill
UK	**First name (F)**	emili, emilu, emily	hazel, hazely, hazle
		elane, elen, elena, ell3n, ell4n, ellan, ellem, ellen	weili, wila, wiley, willa, willia, willie, willo, willow
		henrietta, henriette, henriettia	hildagard, hildegard, hildegarde
	First name (M)	dafid, dav8d, david	pablo, pavel, pavlo
		elias, elijah, elijsh, elisha, eliza, elizah, ellis, ellix	zabian, zabion, zavan, zaveon, zavian, zavion, zevin, zubin
		evelen, evelin, evelyn	korben, korbin, korbyn
	Last name	mackill, maskell, maxwell, mixell	ragsdale, rasdall, rosdahl, rosdol
		barmes, barmison, barnes, barnest, barnez, brines, burns	huckaba, huckabee, huckaby, huckoby, huispe, husby, huseby
		handman, hindman, hinton	laviner, leviner, libner
	Address	old row eawtehstall, old row rawtdnstall, old row rawtenstall	huie rd olin nc 28660, rash rd olin nc 28660, reece rd olin nc 28660
		rise vale street, rose street, rose vale st5eet, rose vale street, rose vale strset, roze street	penannt hills, penant hills, pennant hills, pennatn hills, pennenat hills, pennent hills
		laubd street, laund dtreet, laund street	banks meadow, banksmeadow, banksmeadows

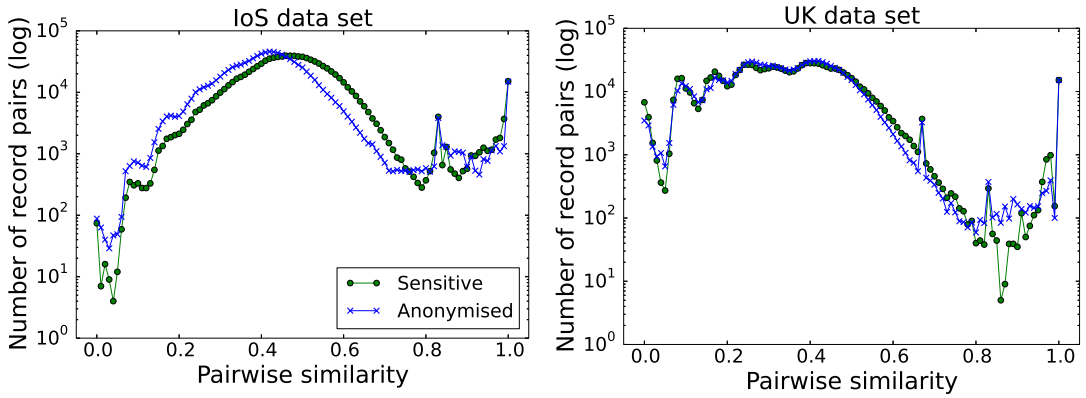

Figure 9.4: Comparison of pairwise record similarities of the sensitive input and the generated anonymised graph data sets.

9.5 Summary

In this chapter we have presented a novel graph anonymisation technique. We discussed how existing graph anonymisation methods are often inapplicable in the RL context since they compromise the structure and the human understandability of the original sensitive data to ensure privacy. In contrast, our proposed anonymisation method preserves the structure of the graph while ensuring the human interpretability of the graph data. The initial step in our anonymisation approach maps attribute values in a sensitive graph data set to attribute values from a public lookup table, whereas in the second step these mapped values are used to anonymise each record in the sensitive data set. We also proposed a method to anonymise date values since many sensitive population data sets contain a date attribute.

We evaluated our anonymisation method using one real-world and one synthetic birth data set. The results showed that our proposed method successfully preserves the structure of the original graph data as reflected by similar pairwise similarity distributions before and after the anonymisation. While in this approach we considered an attribute value mapping method to anonymise all sensitive string values, for attributes such as addresses it is possible to apply alternative methods such as perturbing address geocodes within a specific radius [157]. We aim to consider such alternative anonymisation techniques as future work. In the next chapter, we conduct an overall experimental evaluation of all the RL techniques we proposed from Chapters 4 to 9, using a new real-world data set.

Chapter 10

Overall Experimental Evaluation

In this chapter, we discuss how our techniques proposed in this thesis fit together, and individually evaluate all these techniques using a real-world birth data set which we did not use for evaluation in the previous chapters. The aim of conducting a new set of experiments is to establish the applicability of our methods for real-world Record Linkage (RL) projects, and further verify the conclusions we made in Chapters 4 to 9. In Section 10.1 we highlight the cohesion among our proposed methods, and discuss how they can be used to develop a comprehensive RL framework. Next, in Sections 10.2 to 10.7, we evaluate each of our proposed methods using a new real-world data set, and finally, in Section 10.8, we conclude this chapter with a summary of our findings.

10.1 Introduction

In this thesis, we have proposed solutions to the problems and limitations in the RL domain which we highlighted in Section 1.2. Since so far we discussed them independently, we now provide a concise overview of how our techniques can be combined to develop an end-to-end RL framework.

Figure 10.1 shows how the different methods we proposed in each chapter (highlighted in red) fit into the overall RL process. We assume that a population data set containing sensitive attribute values $\mathbf{D}_s$ is to be linked. We further assume that a pairwise similarity graph $\mathbf{G}_s = (\mathbf{V}_s, \mathbf{E}_s)$ has been generated as we showed in Definition 2, where vertices $v_i \in \mathbf{V}_s$ represent records in the data set to be linked $r_i \in \mathbf{D}_s$ (i.e. $v_i = r_i$), and edges represent compared record pairs $(r_i, r_j) \in \mathbf{E}_s$. Considering that the population data set $\mathbf{D}_s$ contains sensitive information, the corresponding similarity graph $\mathbf{G}_s$ also contains sensitive data. Therefore, as shown in Figure 10.1, we first anonymise this sensitive similarity graph $\mathbf{G}_s$ using the graph anonymisation technique we proposed in Chapter 9, which results in an anonymised similarity graph $\mathbf{G}_a = (\mathbf{V}_a, \mathbf{E}_a)$. We use this anonymised pairwise similarity graph in the remaining steps of the RL process.

Next, we apply the active learning-based filtering approach which we proposed in Chapter 6, on this anonymised similarity graph $\mathbf{G}_a$ to generate a filtered graph $\mathbf{G}_f = (\mathbf{V}_f, \mathbf{E}_f)$ by removing likely non-matching record pairs from the set of edges

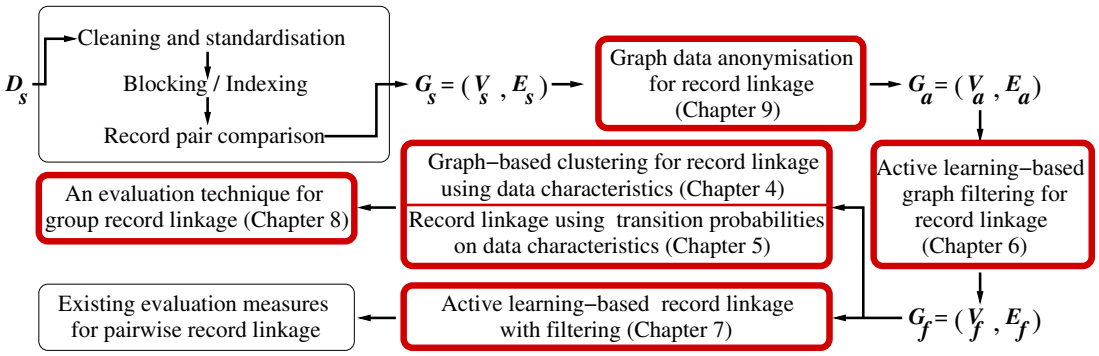

Figure 10.1: Flow diagram indicating how our proposed techniques (highlighted in red) can be integrated to conduct end-to-end record linkage.

$\mathbf{E}_a$. Filtering the anonymised similarity graph in this manner enhances the efficiency of the subsequent classification step. One of the RL classification / clustering approaches which we proposed in Chapters 4, 5 or 7, where the former two employ unsupervised clustering techniques incorporating data characteristics, and the latter employs an active learning method for record pair classification, can then be applied on this filtered, anonymised graph $\mathbf{G}_f$. Finally, the linkage results we obtain with applying the unsupervised clustering approaches (Chapters 4 and 5) can be assessed with the novel group RL evaluation technique we proposed in Chapter 8. Since the active learning-based classification approach we developed in Chapter 7 conducts pairwise classification of records, but not groups of records, existing evaluation measures such as precision and recall can be used to evaluate those results.

Next, in Sections 10.2 to 10.7 we present the further experimental evaluations conducted using the real-world Kilmarnock (Kilm) birth data set which we discussed in Section 2.5. As shown in Tables 2.1 and 2.4 on pages 25 and 30, respectively, the Kilm data set and its pairwise similarity graphs are much larger than the other two birth data sets (IoS and UK) which we used for evaluation in Chapters 4 to 9. Furthermore, the Kilm data set contains many missing values and errors, while the frequency distribution of attribute values are very skewed (similar to the IoS data set) as we discussed in Section 2.5. Therefore, the Kilm data set can be considered more challenging compared to the IoS and UK data sets given its larger size and lower data quality. We conducted all experiments on a server running Ubuntu 18.04 with 64-bit Intel Xeon 2.10 GHz CPUs and 512 GB of memory.

10.2 Graph-based Clustering Using Data Characteristics

In this section we evaluate our proposed graph-based clustering techniques (greedy, star, and robust) using data characteristics such as temporal and spacial information, which we presented in Chapter 4. Similar to the experiments we conducted in Chapter 4, we applied temporal constraints based on temporal data characteristics

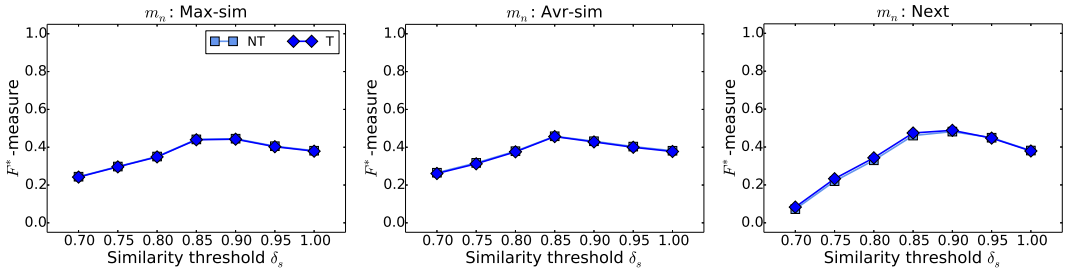

Figure 10.2: Greedy clustering results obtained with (T) and without (NT) temporal constraints: Average of F^* values obtained with different similarity graphs, shown for different similarity thresholds δ_s, and different next vertex selection methods m_n for the Kilm data set.

on the Kilm data set as well. Furthermore, as done in Chapter 4, we evaluated the linkage quality using the F^*-measure and the average of percentage improvements and declines in precision and recall when we applied temporal constraints compared to not using temporal constraints. In addition, we present precision-recall plots in this chapter to provide a better understanding of the behaviour of our proposed algorithms.

Figure 10.2 shows the linkage quality achieved with applying the greedy clustering approach discussed in Section 4.3.1 on page 64, on the Kilm birth data set. With the greedy clustering approach an average percentage precision improvement of 9.64% was achieved at the cost of a 1.78% decline in recall. As implied by these results and the F^* values shown in Figure 10.2, the precision improvement / decline achieved with applying temporal constraints (T) was approximately equal to the decline / improvement in recall and therefore the overall linkage quality was not significantly improved by applying temporal constraints in greedy clustering. This is similar to our observation with linking the IoS (Isle of Skye) and UK data sets using greedy clustering as discussed in Chapter 4.

Figure 10.3 shows the linkage quality achieved when the Kilm data set was linked using the star clustering approach discussed in Section 4.3.2 on page 68. Similar to our observations with the evaluations on the real-world IoS data set in Chapter 4, the F^* values have improved for Kilm with the application of temporal constraints (T) compared to not applying them (NT). With the Kilm data set, a significant average precision improvement of 58.77% was achieved at the cost of a small 1.89% decline in recall when temporal constraints were applied. Based on these results, we can show that star clustering using data characteristics performs well on real-world data sets.

In Figure 10.4, we show the F^* values obtained with conducting RL using the robust graph clustering approach. Given that this clustering technique is computationally more expensive compared to greedy and star clustering approaches, we only executed experiments for the parameter configurations e_b =*Strong* (base cluster generation with *Strong* links only) and e_m =*Norm with WeakHigh* (base clusters merged using both *Norm* and *WeakHigh* edges) since they produced better linkage results for both IoS and UK data sets as we showed in Chapter 4. Similar to our observations

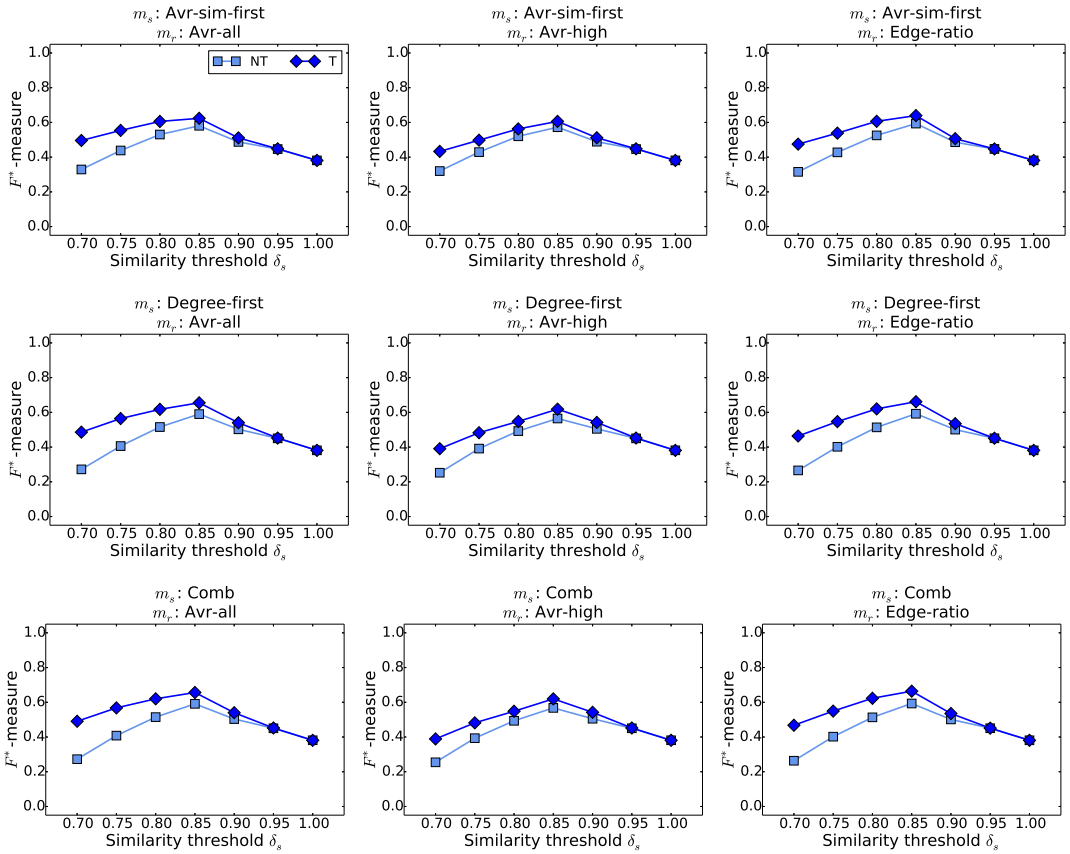

Figure 10.3: Star clustering results obtained with (T) and without (NT) temporal constraints: Average of F^* values obtained with different similarity graphs, shown for different similarity thresholds δ_s, different vertex sorting methods m_s, and different overlap cluster resolving methods m_r for the Kilm data set.

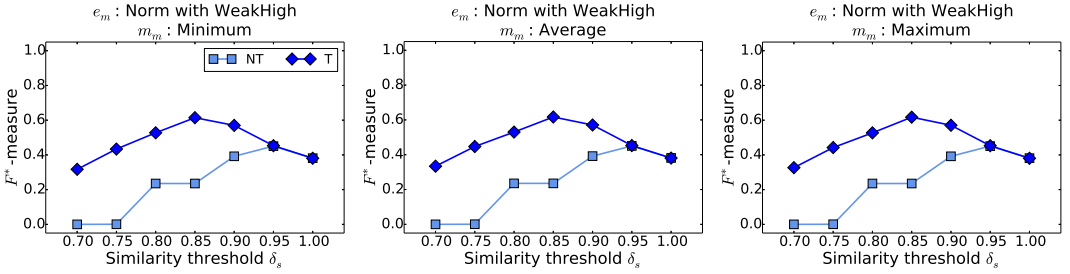

Figure 10.4: Robust graph clustering results obtained with (T) and without (NT) temporal constraints: Average of F^* values obtained with different similarity graphs, shown for different types of edge combinations to merge clusters e_m, and different cluster similarity calculation methods m_m for the Kilm data set.

in Chapter 4, for the real-world IoS data set, a considerable quality improvement is indicated for the Kilm data set when both *Norm* and *WeakHigh* edges are used with temporal constraints. With robust graph clustering, over 30-fold average precision

Table 10.1: The parameter configurations that produced the best F^* value for the Kilmarnock birth data set for each clustering algorithm.

Algorithm	F^*	**Best parameter configurations**		
		G	δ_s	Algorithm specific parameters
Greedy	0.90	$\mathbf{G}_N$	0.95	Temporal / Non-temporal, m_n = Next
Star	0.91	$\mathbf{G}_N$	0.95	Temporal, m_s = Degree-first, m_r = Avr-all
Robust	0.91	$\mathbf{G}_N$	0.95	Temporal, m_m = Average

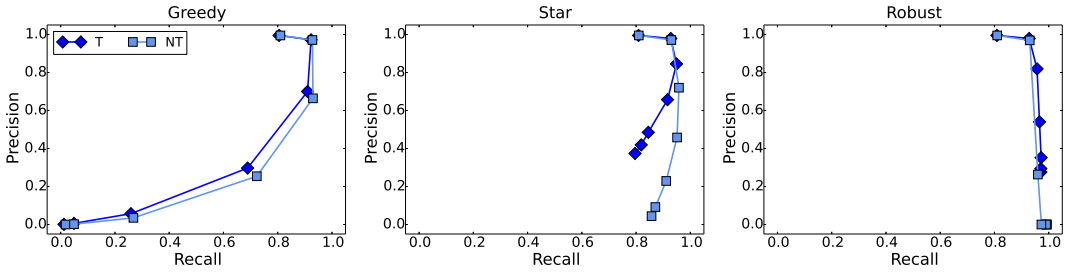

Figure 10.5: The precision-recall (PR) curves obtained for the best parameter configurations for the Kilm data set which consist of the similarity graph $\mathbf{G}_N$ (*Parent names only* attribute value comparison), vertex selection method m_n = *Next* for the greedy algorithm, vertex sorting method m_s = *Degree-first*, and cluster overlap resolving method m_r = *Avr-all* for the star algorithm, and edge combinations e_b = *Strong*, e_m = *Norm with WeakHigh*, and cluster similarity calculation method m_m = *Average* for the robust graph clustering algorithm. Results are shown for incorporating (T) or disregarding (NT) temporal constraints in clustering, for similarity threshold values δ_s ranging from 1.0 to 0.7 in steps of 0.05.

improvement was achieved at the cost of a minor 2.42% reduction in recall for the Kilm data set. As we discussed in Chapter 4, the average precision improvement is greater than 100% due to the linkage precision being very low when temporal constraints are disregarded.

In Table 10.1 we present the parameter configurations which produced the best results with regard to the F^*-measure for the Kilm data set. For greedy clustering, the vertex selection method m_n=*Next* produced the best linkage results, which is similar to the linkage results we presented for the IoS and UK birth data sets in Table 4.1 on page 80. However, these best clustering results were obtained for both with and without temporal constraints which further shows that greedy graph clustering using data characteristics fails to improve linkage quality compared to the baseline. With star clustering, the cluster overlap resolving method (m_r) *Avr-all* worked best for Kilm (similar to IoS and UK data sets as shown in Table 4.1), whereas the *Degree-first* method was best for vertex sorting (m_s). For robust graph clustering, the *Average* cluster similarity calculation method (m_m) produced the best results for the Kilm data set, similar to the IoS and UK data sets.

Next, in Figure 10.5, we show the precision-recall (PR) plots for the best algorithm specific parameter configurations presented in Table 10.1. We plot the precision and

Table 10.2: The minimum (Min), maximum (Max), average (Avr), and median (Med) run-times (in seconds) of each clustering algorithm for Kilm birth data set.

Algorithm	Run-time (NT)				Run-time (T)			
	Min	Max	Avr	Med	Min	Max	Avr	Med
Greedy	0.19	48,415.20	648.68	23.86	0.19	4,020.95	254.06	29.25
Star	34.80	1,783.67	114.74	79.73	34.95	2,827.55	221.29	80.34
Robust	690.30	591,269.97	49,294.48	1,015.26	18,063.94	46,633	28,032.57	28,236.32

recall values for similarity threshold values δ_s ranging from 1.0 to 0.7 in steps of 0.05. With greedy clustering, both precision and recall have rapidly declined for decreasing δ_s values. This is due to many non-matching record pairs being grouped together and true matches being missed with the vertex selection method (m_n) *Next* at lower similarity thresholds. Notice how greedy clustering with temporal constraints (T) has only produced slightly better results compared to the non-temporal (NT) approach, as we discussed earlier. Similarly, with the star clustering approach, both precision and recall decline for decreasing thresholds even though not as rapidly as with greedy clustering. For both star and robust graph clustering, relatively higher precision values have been maintained when temporal constraints were applied (T), compared to not applying temporal constraints (NT).

In Table 10.2, we present the minimum, maximum, average, and median values of algorithm run-times in seconds for the Kilm data set. The maximum and average run-times were considerably reduced for the greedy and robust clustering techniques when temporal constraints were applied (T) compared to not using temporal constraints (NT). However, the average run-time slightly increased when temporal constraints were considered for the star clustering approach, which was potentially caused by the extra computational effort required for checking temporal constraints. The run-time results obtained for the Kilm data set are similar to the efficiency analysis done for the real-world IoS data set in Table 4.2 on page 81. Based on the experimental evaluations we conducted in this chapter and in Chapter 4, we can state that our proposed star and robust graph-based clustering techniques using data characteristics improve the linkage quality, and often the efficiency of real-world RL tasks, compared to when using unsupervised techniques without data characteristics.

10.3 Record Linkage Using Transition Probabilities

In this section, we evaluate the three cluster goodness measures we proposed in Chapter 5 using the Kilm data set. As the baseline technique we used the star clustering approach with temporal (T) data characteristics using the best algorithm specific parameter configurations, and the pairwise similarity graph $\mathbf{G}_N$, as we specified in Table 10.1. We executed star clustering with the technique outlined in Algorithm 5 to comparatively evaluate our overlap resolving technique with the baseline approach. The experimental setup was the same as for the experiments conducted in Chapter 5, as described on page 91.

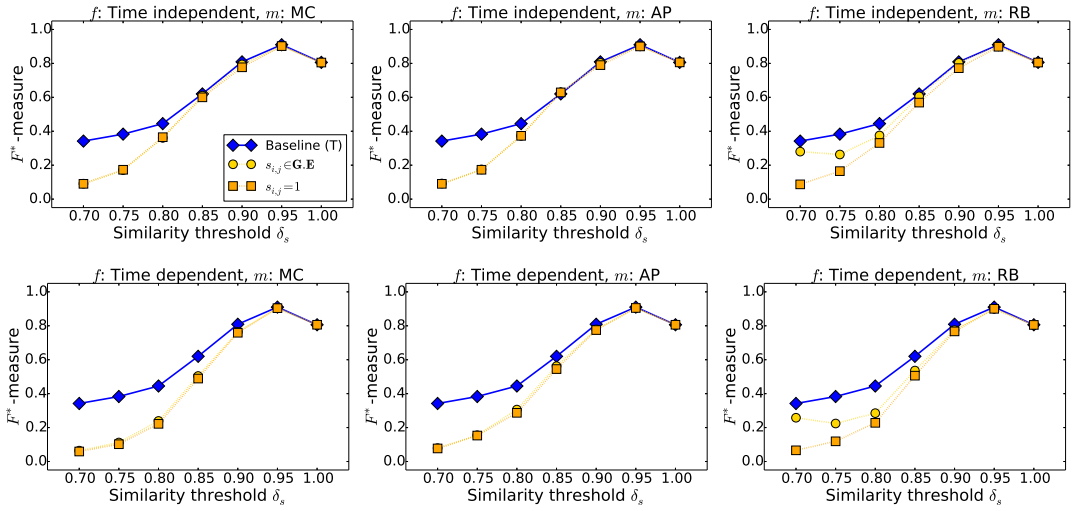

Figure 10.6: Clustering the Kilm data set with transition probabilities: F^* values obtained for different similarity threshold values δ_s (as we discussed in Chapter 4), shown for time independent and dependent population goodness calculations (f) for different cluster goodness measures (m). The pairwise similarity is either set to a constant value ($s_{i,j} = 1$) or taken from the edge weights in the pairwise similarity graph ($s_{i,j} \in \mathbf{G}.\mathbf{E}$) in the cluster goodness calculations, and clustering with temporal (T) constraints (proposed in Chapter 4) is used as the baseline.

Table 10.3: The minimum (Min), maximum (Max), average (Avr), and median (Med) run-times (in seconds) of star clustering using the goodness measures *MC*, *AP*, and *RB* for cluster overlap resolving, for the Kilm birth data set.

Baseline run-times	**Algorithm**	**Run-time**			
		Min	**Max**	**Avr**	**Med**
Min = 34.95, Max = 2,827.55	MC	114.50	15,858.74	2,770.52	183.79
Avr = 221.29,	AP	118.37	16,294.75	2,878.05	182.09
Med = 80.34	RB	120.06	16,957.62	2,969.14	237.78

Figure 10.6 shows the linkage quality achieved when cluster overlap resolving was conducted with our proposed cluster goodness measures *Markov Chain* (*MC*), *All pairs* (*AP*), and *Record-based* (*RB*), using different parameter configurations on the Kilm birth data set. The baseline temporal constraints-based (T) star clustering approach has performed better compared to the linkage quality achieved when the cluster goodness measures were incorporated. These results are similar to the observations we made based on the evaluation conducted in Chapter 5 for the real-world IoS data set. Similar to the IoS data set, the Kilm data set too has few infeasible patterns in the birth transition distributions due to data quality issues, which results in mistakes in the goodness calculations leading to reduced linkage quality.

Table 10.3 shows the summary of run-times obtained for star clustering incorporating overlap resolving with our proposed cluster goodness measures *MC*, *AP*, and *RB* for the Kilm birth data set. The time taken for executing star clustering

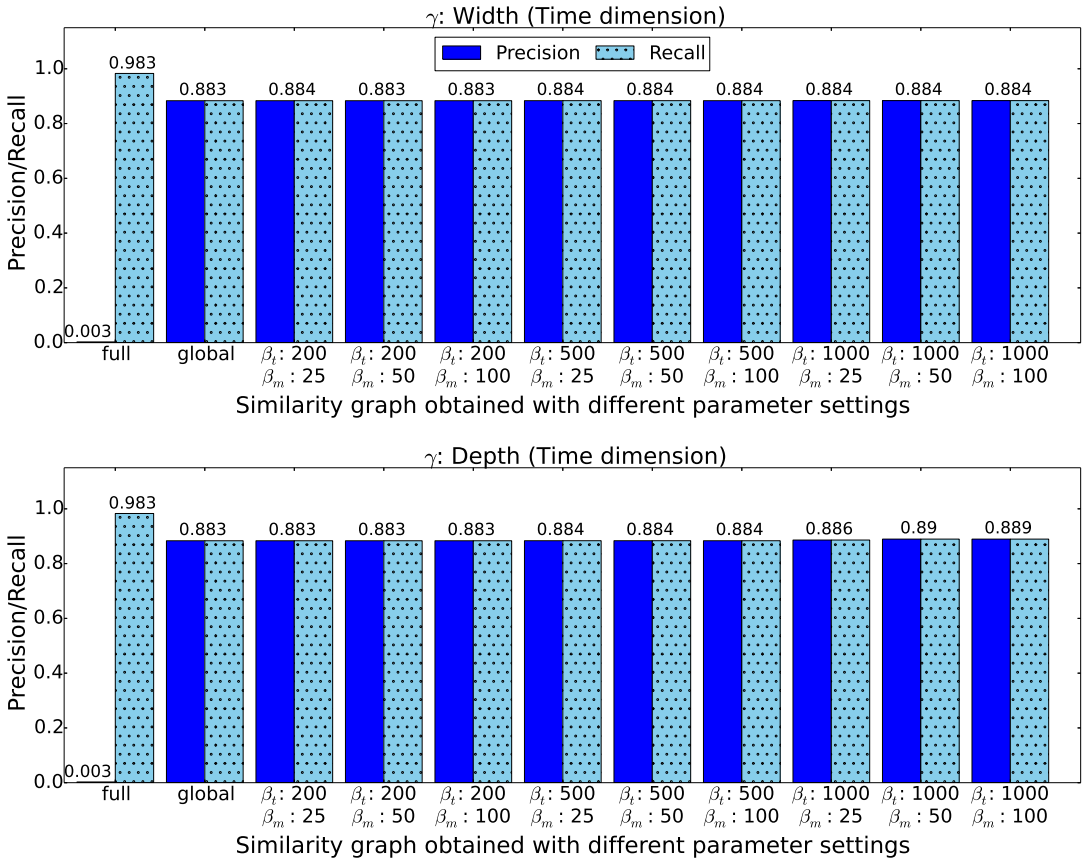

Figure 10.7: Graph filtering with binning on the time data dimension for the Kilm data set: Precision and recall results of the full similarity graph compared with the quality of the graph filtered with a global threshold (top o_m links), or binwise thresholds for different total budgets β_t and manual classifications per bin β_m.

with our proposed overlap resolving methods is considerably greater than the time taken to run the baseline star clustering approach with temporal constraints according to these results. The run-time results too for the Kilm data set are similar to the run-times we obtained for the IoS data set in Chapter 5. Based on the experimental evaluation conducted in this section and Chapter 5, we can conclude that incorporating data characteristics alone produces better RL results compared to incorporating transition probabilities, considering both the linkage quality and efficiency.

10.4 Active Learning-based Graph Filtering

In this section, we evaluate the active learning-based graph filtering technique we proposed in Chapter 6 using the Kilm data set. We used the pairwise similarity graph $\mathbf{G}_A$ (where *All* attribute values are compared) since it has more widely dis-

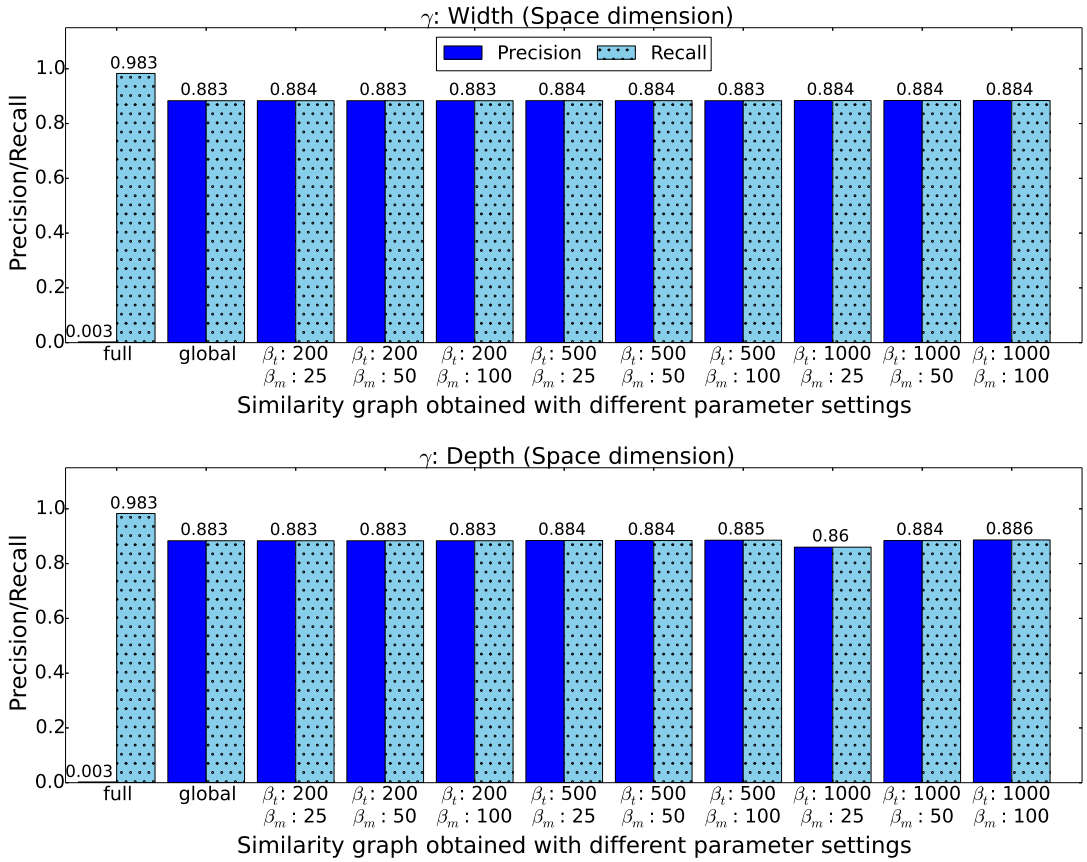

Figure 10.8: Graph filtering with binning on the space data dimension for the Kilm data set: Precision and recall results of the full similarity graph compared with the quality of the graph filtered with a global threshold (top o_m links), or binwise thresholds for different total budgets β_t and manual classifications per bin β_m.

tributed pairwise similarities, and set parameter configurations as specified in the experiments section of Chapter 6. The pairwise similarity graph $\mathbf{G}_A$ corresponding to the Kilm birth data set contains approximately 25 million record pairs as shown in Table 2.4 on page 30, which is significantly reduced to only o_m = 68,215 record pairs as shown in Table 2.1 on page 25 by applying our graph clustering method.

In Figures 10.7 and 10.8 we show the precision and recall results obtained for the original full similarity graph, as compared with the filtered graphs for the global threshold (the baseline technique where we select the top o_m record pairs), as well as results for our binning method when using different total budgets β_t and different manual classification thresholds per bin β_m. According to these figures, applying graph filtering has resulted in a considerable precision improvement at the cost of a minor decline in recall, compared to the quality of the full similarity graph. Figures 10.7 and 10.8 show the quality of the filtered graphs when binning was applied on the time and space dimensions, respectively, whereas a slight improvement in

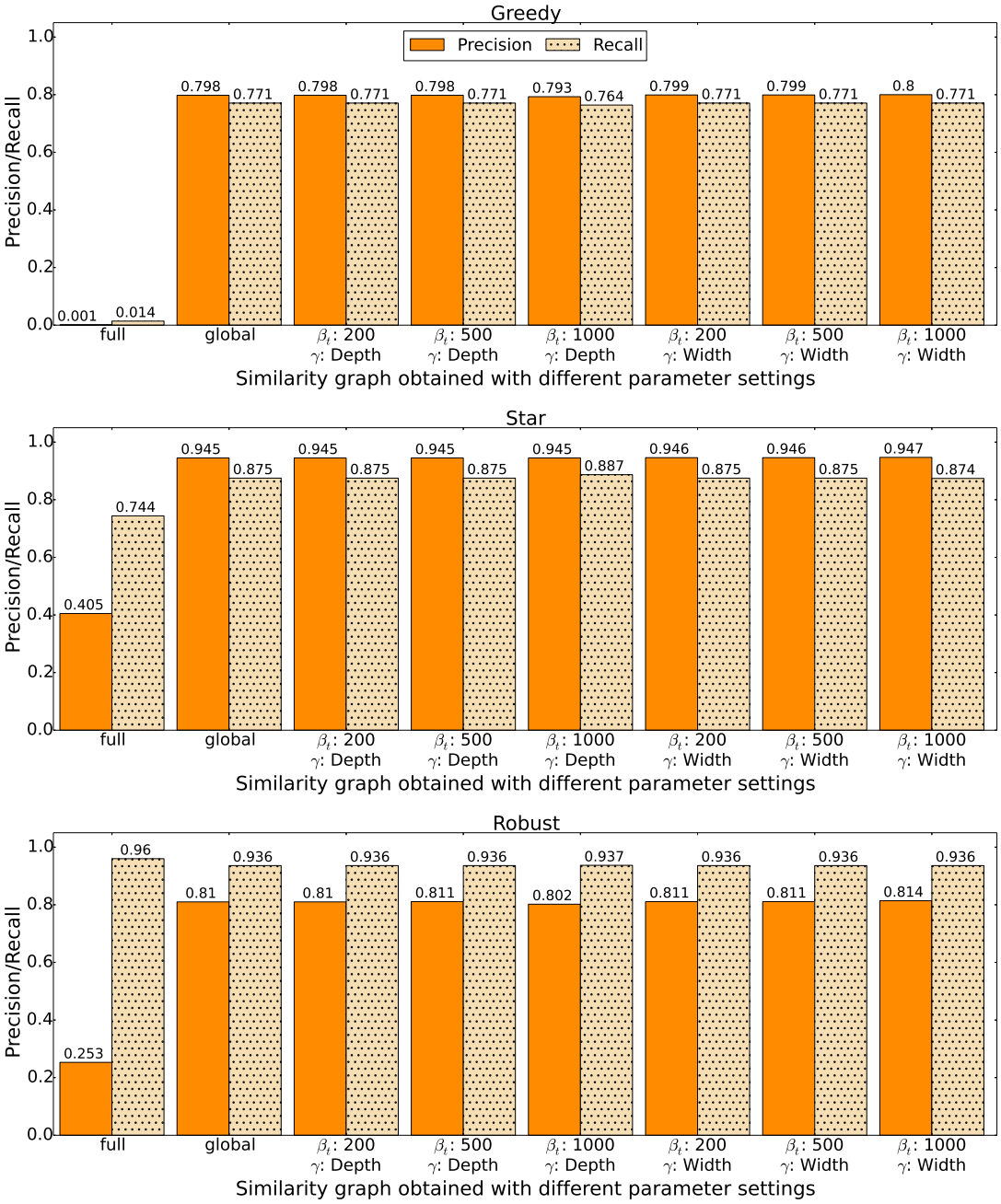

Figure 10.9: Precision and recall results obtained by applying greedy, star and robust graph clustering on the full similarity graph, graph filtered with a global threshold (top o_m links), or graphs filtered with equal depth and equal width binning using different total budgets β_t for the Kilm data set.

graph quality was observed for at least one parameter configuration when our filtering method was applied compared to filtering using the global threshold method.

Table 10.4: The run-times (in seconds) of graph clustering using the full similarity graph, graph filtered with a global threshold (top o_m links), or graphs filtered with equal depth (EDB) and equal width (EWB) binning using different total budgets β_t, for the Kilm data set.

	full	global	β_t: 200 γ: EDB	β_t: 500 γ: EDB	β_t: 1000 γ: EDB	β_t: 200 γ: EWB	β_t: 500 γ: EWB	β_t: 1000 γ: EWB
Greedy	1,060.28	1.45	1.43	1.43	1.41	1.49	1.45	1.47
Star	1,661.94	6.22	3.82	3.67	3.71	6.73	3.74	3.78
Rubust	101,067.32	19,362.61	19,138.77	19,094.53	19,241.75	20,118.98	20,126.56	20,111.43

Figure 10.9 shows the clustering quality achieved with the full and filtered pairwise similarity graphs for the Kilm data set. While both the clustering precision and recall have significantly improved for greedy and star techniques, the precision has improved considerably at the cost of a minor decline in recall for the robust method when clustering was conducted on the filtered graphs compared to clustering the full similarity graph. For all three clustering methods, β_t = 1000 produced the best results, where equal width binning worked best for the greedy and robust graph clustering techniques while equal depth binning worked best for star clustering. The corresponding baseline (global) clustering quality was slightly exceeded with these parameter settings.

In Table 10.4 we present the run-times obtained with applying graph clustering on the full and filtered pairwise similarity graphs for the Kilm data set. Notice how the RL classification time has been reduced considerably when clustering was applied on the filtered graphs compared to clustering the full similarity graphs for every clustering algorithm. Based on the experiments conducted in this section and in Chapter 6, we can further validate our proposed method of active learning-based graph filtering for improving both the quality and the efficiency of RL.

10.5 Active Learning-based Record Linkage With Filtering

In this section, we present the evaluation results obtained with applying our active learning with filtering approach for RL (RALF) which we proposed in Chapter 7 on the Kilm data set. Similar to the experiments conducted in Chapter 7, we used the pairwise similarity graph generated by comparing *All* attribute values $\mathbf{G}_A$ for the Kilm data set as well, since it contains the largest number of features for classifier training. According to the quality analysis of functional parameters in RALF we conducted in Chapter 7, the record pair sampling method m_s = *Random*, the inactivation method m_i = *Threshold*, and applying confidence scaling f = *True* produces better linkage results compared to using the other configurations for the corresponding parameters (see Table 7.2 on page 127). We therefore set the functional parameters to these values, and the remaining parameters were set as specified in Chapter 7.

Figure 10.10 shows how the linkage quality of our active learning technique changes with varying parameter values for the Kilm data set. The AdaBoost and

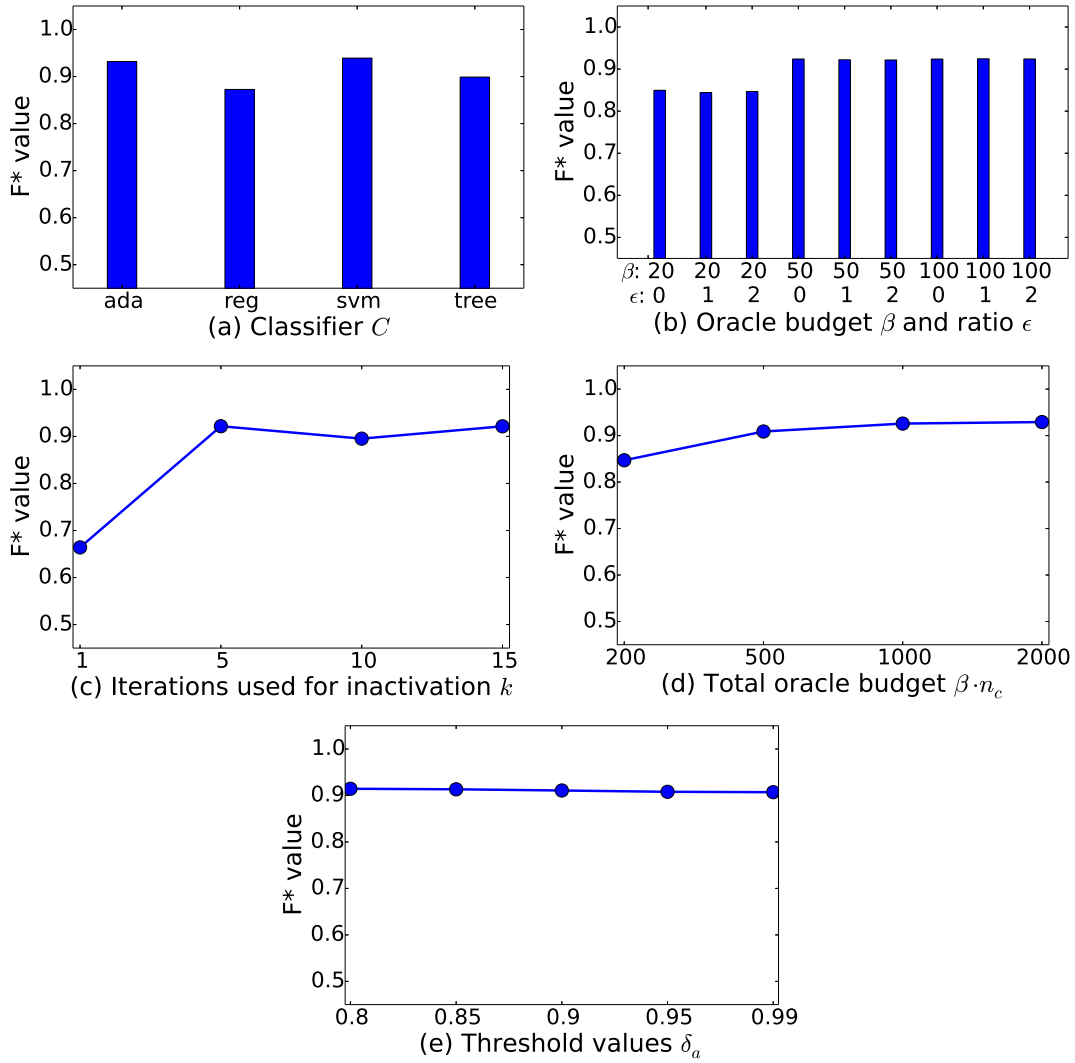

Figure 10.10: F^* values obtained with different parameter settings for executing active learning-based RL on the Kilm data set.

SVM algorithms have performed best on Kilm as shown in Figure 10.10 (a), which is similar to the behaviour we observed for the experiments with the larger data sets in Chapter 7. Figure 10.10 (b) shows an upward trend in F^* results when the oracle budget β is increased from 20 to 50, but not beyond that value, which is potentially caused by the linkage quality already being quite high for β = 50 which leaves little room for further quality improvement. As shown in Chapter 7 as well, changing the ratio ϵ has minimal impact on the linkage results.

Similar to the results shown in Chapter 7, the quality as measured with F^* generally improves for increasing values of k (the number of iterations to consider for inactivating record pairs) for the Kilm data set as shown in Figure 10.10 (c). This is due to considering the classification outcome of a larger number of classifiers to

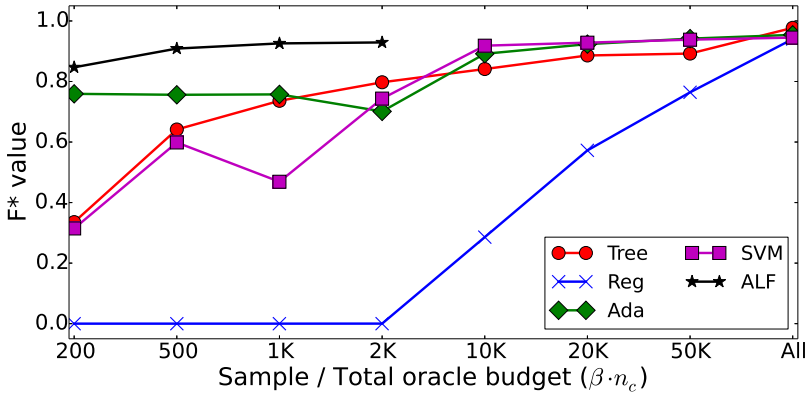

Figure 10.11: The F^* results obtained with different supervised classification algorithms with different training sample sizes, and our proposed RALF technique with different total oracle labelling budgets $\beta \cdot n_c$ for the Kilm data set.

Table 10.5: Time taken in seconds to run fully supervised (FullSup) classification and our proposed RALF technique, and the percentage run-time reduction for the Kilm data set.

	FullSup	RALF	Percentage run-time reduction
Ada	8,929	6,836	23.44%
SVM	1,209,019	4,643	99.62%

decide which record pairs to automatically label and inactivate, being more reliable than considering fewer classification outcomes. Our observation with increasing the oracle budget per iteration β in plot (b) is further attested by the trend in linkage quality for increasing total oracle budget values $\beta \cdot n_c$ as shown in Figure 10.10 (d). The last plot in Figure 10.10 (e) shows how the linkage quality does not change much for varying threshold values d_a, similar to the results obtained for experiments conducted in Chapter 7. Unlike in Chapter 7, we do not explore the percentage values δ_b since we set the inactivation method m_i = *Threshold* for experiments in this section.

In Figure 10.11 we compare the linkage quality of RALF and supervised classification techniques for the Kilm data set, similar to the experiments we conducted in Chapter 7. We increased the training sample size for supervised classifiers from 200 to 50,000 with random record pair sampling of an equal number of matches and non-matches, and also conducted classifier training on the full data set. We considered total budgets $\beta \cdot n_c$ of 200, 500, 1,000 and 2,000 for oracle labelling in our RALF method, and report the average quality obtained for each total budget.

As observed for the experiments conducted with the larger data sets in Chapter 7, our RALF method exceeds the linkage quality obtained with supervised classifiers with random sampling for corresponding budget values. Furthermore, the linkage quality achieved with only $\beta \cdot n_c$ = 2,000 labelled record pairs is similar to the linkage quality achieved with using the full data set for classifier training. This further jus-

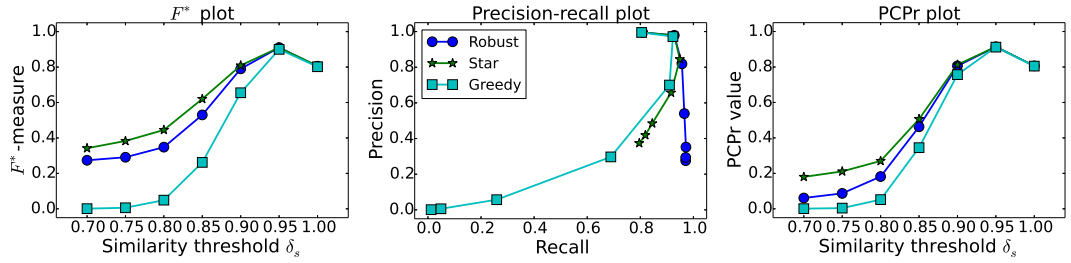

Figure 10.12: The F^*, precision-recall (PR), and penalised clustering precision (PCPr) plots corresponding to the linkage results obtained by conducting record linkage on the Kilm birth data set using the greedy, star, and robust graph clustering approaches.

tifies the application of our RALF technique for linking real-world data sets, rather than conducting supervised classification with randomly selected training data.

Table 10.5 shows a comparison of the average time taken to run RALF with the time taken to run the supervised classification techniques on the Kilm data set. We only compared our method with AdaBoost and SVM since they produced the best linkage results for Kilm as shown in Figure 10.10 (a). As shown in Table 10.5, the RALF method is significantly more efficient than running supervised classification on the full data sets, which is similar to the observations we made based on the experiments conducted in Chapter 7. Our RALF method has achieved an efficiency improvement (reduction in run-time) of 23% for AdaBoost, and above 99% for SVM, compared to running fully supervised classification with the same algorithms. The linkage quality and efficiency analysis conducted in this section as well as Chapter 7 show the effectiveness of our proposed RALF method for RL.

10.6 An Evaluation Technique for Group Record Linkage

In this section, we apply the novel evaluation measure we proposed in Chapter 8 for assessing group RL techniques on the Kilm data set. Similar to the experiments we conducted in Chapter 8, we assess the greedy, star and robust graph clustering approaches which we proposed in Chapter 4 using this novel evaluation measure. In this section, we only consider the clustering results obtained with the *Parent names only* ($\mathbf{G}_N$) pairwise similarity graph generated for the Kilm data set since it produced the best clustering results as shown in Table 10.1. Furthermore, the algorithm specific parameters were also set as shown in Table 10.1 for the Kilm data set since they produced the best linkage results.

Figure 10.12 shows the PR curves, the F^*-measures (described in Section 2.4), and the penalised clustering precision (PCPr) values (described in Section 3.5) obtained for the greedy, star and robust graph clustering approaches, for the Kilm birth data set. According to these evaluation measures, the star and robust graph clustering approaches have clearly performed better than the greedy algorithm as indicated by all three measures. In Table 10.6 we show the area under the PR curves (AUPRC)

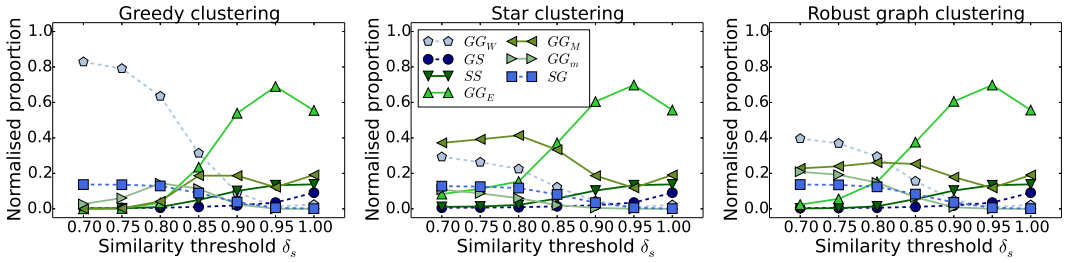

Figure 10.13: Plots for new evaluation measure corresponding to the linkage results obtained by conducting record linkage on the Kilm birth data set using the greedy, star, and robust graph clustering approaches.

Table 10.6: Area under the curve (AUC) values for linking the Kilm data set using the three clustering techniques with the best value highlighted in each column.

	Algorithm	**Area under the curve (AUC)**							**Average**
	AUPRC	GG_E	GG_M	GG_m	SS	GG_W	GS	SG	(AUC_{avr})
Greedy	0.82	0.30	0.11	0.06	0.06	0.38	0.02	0.08	-0.028
Star	0.86	**0.38**	**0.29**	0.04	**0.07**	**0.14**	0.02	**0.07**	**0.118**
Robust	**0.95**	0.36	0.21	**0.09**	0.06	0.18	0.02	0.08	0.087

which shows robust graph clustering as the best algorithm. However, as per the F^* and PCPr plots in Figure 10.12, the star clustering algorithm has performed slightly better than the robust graph clustering method. Similar to our observations based on the experiments conducted in Chapter 8 on the IoS and UK birth data sets, this further shows the possibility for ambiguities to occur among various existing evaluation measures.

Figure 10.13 shows the plots for our novel cluster evaluation method for the three clustering techniques. The normalised proportions of the seven categories from Table 8.1 on page 139 are shown against the similarity threshold δ_s used in each clustering algorithm to filter record pairs. As we described previously, for better clustering results the values of *correct singleton* (SS), *exact group match* (GG_E), *majority group match* (GG_M), and *minority group match* (GG_m) should be higher whereas the values of *wrongly grouped singleton* (SG), *missed group member* (GS), and *wrongly assigned member* (GG_W) should be lower.

According to Figure 10.13, the proportion of records in the GG_E and GG_M categories are much higher for the star and robust graph clustering approaches compared to greedy clustering. Furthermore, with the greedy clustering method, most records have been assigned to the incorrect cluster (group) as indicated by the large GG_W proportions. These results show that as per our evaluation method, star and robust graph clustering outperform the greedy method for Kilm, which agrees with the conclusion made based on existing cluster evaluation measures as shown in Figure 10.12.

We further compare the new evaluation results with existing evaluation results in Table 10.6, where we present the area under the curve (AUC) of PR (based on the PR curves in Figure 10.12), the AUC of our new cluster evaluation plots (based

on the plots in Figure 10.13) and a simple averaging of the AUC values across the seven categories AUC_{avr} as defined in Equation 8.1 on page 144. For the Kilm data set, the best clustering algorithm as per our evaluation measure AUC_{avr} is star, followed by robust graph clustering. The negative AUC_{avr} value obtained for the worst performing greedy algorithm indicates that it generates more incorrect clusters than correct clusters. This conclusion complements the F^* and PCPr values shown in Figure 10.12 but slightly varies from the assessment made based on the AUPRC values which shows robust graph clustering as the best performer. These results are similar to the results we obtained for the real-world IoS birth data set based on the experiments we conducted in Chapter 8. The experimental results presented in this chapter further show that our novel evaluation approach complements the F^*-measure, and clearly indicates which algorithms have generated the largest number of correct record groupings.

10.7 Graph Data Anonymisation

In this section we evaluate the graph anonymisation technique we proposed in Chapter 9 using the Kilm data set. We apply the same parameter configurations as used in the evaluation conducted in Chapter 9 with the IoS and UK birth data sets.

Table 10.7 shows how our anonymisation method maps sensitive attribute values from the Kilm data set to public attribute values from a lookup table, such that the pairwise similarities and attribute value lengths are maintained. Notice how the last names 'cree' and 'crorie' with shorter string lengths have been mapped to similarly short string values 'asby' and 'ashby', whereas longer last names 'mcclure' and 'mcilroy' are mapped to relatively longer public attribute values 'rehburg' and 'rhuberg', respectively, while maintaining pairwise similarities.

To illustrate the quality of the anonymised graphs generated by our anonymisation technique, Figure 10.14 shows the distribution of the pairwise similarity for record pairs from the Kilm data set. Similar to the experiments we conducted in Chapter 9, we have used a randomly sampled subset of 1 million record pairs including all true matching pairs from the Kilm data set for plotting. For each record pair, the similarity is calculated using the Jaro-Winkler similarity measure on names and the Dice coefficient similarity measure on addresses [128] (described in Section 2.3.3), followed by an averaging of these values. As can be seen from this figure, the similarity distribution of both the sensitive input values and the generated anonymised values are highly similar for the Kilm data set.

As shown in Figure 10.15, we also compared the linkage quality achieved with applying the graph clustering techniques we proposed in Chapter 4 on both the original sensitive input as well as the anonymised Kilm graph data sets. While we conducted experiments using the best algorithm specific parameters as listed in Table 10.1, both the sensitive and anonymised pairwise similarity graphs were generated by comparing the *Parent names and address* attributes ($\mathbf{G}_{NA}$) as described in Section 2.6, since our proposed anonymisation technique supports anonymising names and addresses.

Table 10.7: Sample of sensitive to public female (F) first name, male (M) first name, last name, and address attribute value mappings conducted by our graph data anonymisation technique for the Kilm data set.

	Sensitive attribute values	**Mapped public attribute values**
First name (F)	wilhelmina, william	melchorita, melzora
	mary, mr, mr0	yara, yura, yuri
	agnes, akns	peiyi, piya
First name (M)	francis, frnss	papanii, pavan
	andrew, antr	ollice, olsi
	gunnion	joseph
Last name	carter, cartwright, charters	maneval, mannepalli, manville
	mcclure, mcilroy, mcilwraith, mclarty, mcleary, mclure	rehburg, rhuberg, ripberger, roberge, roberson, ropers
	cree, crorie, crow, currie	asby, ashby, aspey, ausby
Address	back causeway, back causey, back causey?, back causway	palmer oakey, palmer oaky, palmers oakey, palmers oaky
	west woodstock st, woodstock st, woodstock st wesst, woodstock st west, woodstock st - west, woodstock street	lethbrdge park, lethbride park, lethbridge park, lethbrige park, letherbridge park, letheridge park
	onthank farm, top onthank farm, toponthank farm	glen warning, glenn warming, glenn warning

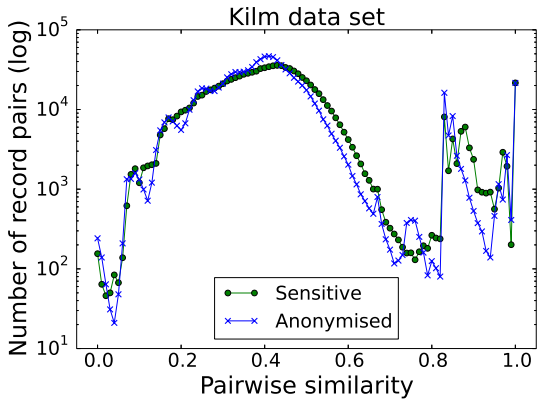

Figure 10.14: Comparison of pairwise record similarities of the sensitive input and the generated anonymised Kilm graph data sets.

Figure 10.15 shows the PR curves for each algorithm for decreasing similarity threshold values δ_s (see Chapter 4) from 1.0 to 0.7 in steps of 0.05. Notice how the PR curves for both the sensitive input and the anonymised Kilm graph data sets are similar for each clustering algorithm. The only noteworthy difference in these patterns is that a significant recall improvement is observed for different similarity threshold δ_s changes for the sensitive input (δ_s decreased from 0.9 to 0.85), and the anonymised graph data sets (δ_s decreased from 0.85 to 0.8). Such minor differences in linkage results can be caused by the slight variations in the similarity distributions of the sensitive input and the anonymised graph data sets, as we showed in Figure 10.14.

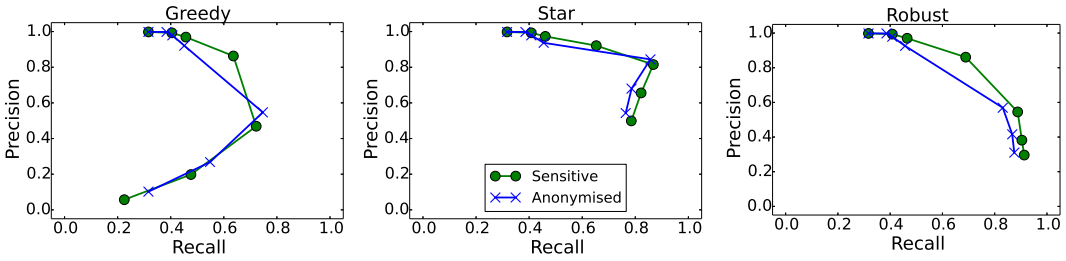

Figure 10.15: Comparison of the linkage quality achieved by applying the greedy, star, and robust graph clustering techniques (described in Chapter 4) on the sensitive input and the generated anonymised Kilm graph data sets. Results are shown for similarity threshold values δ_s ranging from 1.0 to 0.7 in steps of 0.05.

All the experiments conducted in this chapter and Chapter 9 show the capability of our anonymisation method to anonymise sensitive graph data, while maintaining its structure, as reflected by the pairwise similarity distributions and the clustering results obtained with the sensitive input and the anonymised graph data sets.

10.8 Summary

In this chapter, we discussed how the different techniques we proposed in this thesis can be integrated into an end-to-end RL framework, and we conducted an overall experimental evaluation using a real-world birth data set which we did not use for evaluation in the previous chapters. The experimental results presented in this chapter further verified the conclusions we made in the previous chapters. Note that while this chapter provides an overview of how our proposed techniques can be integrated into a RL framework, it does not necessarily demonstrate this with the new real-world data set that we have used for experiments. Rather, we further verify the generalisability of our proposed techniques for the application on real-world RL tasks, using a new population data set.

Based on the experiments conducted in this and the previous chapters, we showed that applying data characteristics (such as temporal and spacial information) on real-world unsupervised data linkage tasks improves the linkage quality compared to conducting unsupervised RL without considering data characteristics. However, extending our proposed methods to consider the probability of transitioning between states in a population data set (such as a person changing their occupation) did not improve results. We showed that our proposed active learning-based filtering technique improves the efficiency of the RL classification step while improving the linkage precision at the cost of a minor loss in recall. The active learning-based RL technique we proposed was shown to be on par with or exceeding the linkage quality obtained with existing active learning and supervised RL approaches.

We then discussed the issues with existing evaluation measures for group RL, and showed that our proposed novel evaluation measure provides more insightful information regarding the linkage results based on how records have been assigned into

clusters and the resulting cluster sizes. Finally, we evaluated a graph anonymisation technique which can effectively anonymise sensitive graph data (used for RL and in other domains) without compromising their human intepretability and similarity structure as shown in this and previous chapters. In the following final chapter, we summarise the work presented in our thesis and discuss future research directions.

Chapter 11

Conclusion and Future Work

In this thesis we have proposed efficient and effective algorithms for Record Linkage (RL), while ensuring that the privacy of the resulting linked data set is preserved, and that the linkage quality of group RL techniques can be assessed in a robust manner. Our proposed techniques, which we discussed in Chapters 4 to 9, aim to address limitations in the existing literature of the domain of RL. In this chapter, we conclude our thesis by highlighting the research problems in Section 11.1, and summarising our contributions to address these research problems in Section 11.2. Furthermore, we summarise the main research findings of our thesis in Section 11.3. Next, we discuss potential directions for future research in Section 11.4, and finally, in Section 11.5 we provide the final concluding remarks for our thesis.

11.1 Summary of the Research Problems

As we discussed in Section 1.2, there are several challenges in the Record Linkage (RL) domain which are still widely explored by the research community. A summary of these research problems is as follows.

- **Data Set Size**: The massive amount of data available in the modern world has resulted in the infeasibility of conducting RL with naïve record pair comparison. Even though methods such as blocking and indexing [12, 36] are applied to improve the efficiency of the RL process, due to the high class imbalance (true matches being considerably outweighed by the number of true non-matches), more advanced filtering techniques need to be developed to conduct RL efficiently.

- **Data Quality**: Real-world data sets, especially population data, often lack data quality due to errors introduced in data collection and transcription [11, 93, 143]. The task of RL is made more challenging when data quality issues are present, since reduced data quality makes it more difficult to distinguish between matches and non-matches when comparing attribute values.

- **Data Privacy**: Ensuring the privacy of linked data sets is a major concern, and is sometimes detrimental to research in the domain of RL [40]. Even though

Privacy-Preserving Record Linkage (PPRL) techniques and existing graph data anonymisation methods can be used to enhance the privacy of linked data, they often compromise the human interpretability and the relationships among the original data.

- **Lack of Ground-truth Data**: Due to the presence of a large number of record pairs across data sets to be linked, it is infeasible to manually label all record pairs in the RL context [35]. This commonly leads to a lack of ground-truth data in the form of known matches and non-matches. Supervised classification techniques are therefore often inapplicable in RL projects due to the limited availability of training (ground-truth) data.
- **Evaluating Linkage Results**: The existing evaluation measures used for assessing the quality of RL techniques were originally developed to assess machine learning or information retrieval approaches [35, 82, 85]. These methods are used to evaluate the linkage quality of group RL methods as well, by considering the record pairs in each group. However, conducting evaluation using individual record pairs does not provide insightful information about how accurately records have been grouped together (such as the distribution of cluster sizes). Therefore, it is necessary to conduct a proper assessment of these evaluation measures to determine their applicability for evaluating group RL methods.

11.2 Summary of Contributions

In this section, we provide a summary of our contributions where we proposed new techniques to address the challenges highlighted in the previous section.

- **Graph-based Clustering for Record Linkage Using Data Characteristics**: In Chapter 4 we proposed three novel unsupervised graph clustering techniques, named greedy, star, and robust, which incorporate data characteristics such as temporal and spatial information to improve linkage quality [122, 123]. With an empirical evaluation conducted on two real-world and one synthetic birth data set in Chapters 4 and 10, we showed that the linkage quality achieved with the unsupervised star and robust graph clustering techniques can be significantly improved by considering data characteristics, even when data quality issues are present in the data sets being linked.
- **Record Linkage Using Transition Probabilities on Data Characteristics**: In Chapter 5, we then further developed our concept of incorporating data characteristics in the RL process by considering transition probability distributions corresponding to data characteristics. That is, we proposed methods of modelling the *goodness* of groups of records (clusters) based on the probability for entities to transition between *states* (such as occupations, and roles as daughter or mother in population data) based on a data characteristic (such as within a

given time duration). The experiments conducted with two real-world and one synthetic birth data sets in Chapters 5 and 10 showed that this method is prone to errors when data quality is lacking, whereas considering data characteristics alone, as we proposed in Chapter 4, was shown to produce better linkage quality.

- **Active Learning-based Graph Filtering for Record Linkage**: Next, in Chapter 6, we proposed a novel active learning-based graph filtering technique, which, based on a domain expert's knowledge about the expected number of true matches and the data patterns along data dimensions (such as time and space), filters record pairs to improve the efficiency of the RL classification step [125]. With an experimental evaluation conducted using three real-world and one synthetic data set in Chapters 6 and 10 we showed that applying RL clustering methods on filtered graphs considerably improves the efficiency of the RL process while improving the precision at the cost of a minor decline in recall.

- **Active Learning-based Record Linkage With Filtering**: In Chapter 7, we then proposed a new iterative active learning-based RL approach which manually classifies a limited number of record pairs by a human oracle (domain expert), and employs an automatic classification method to obtain a larger training data set. Furthermore, this method filters record pairs based on the confidence of record pair classifications in previous iterations to improve the RL classification efficiency. We evaluated this method using two real-world bibliographic data sets, two real-world and one synthetic birth data sets, and a real-world voter data set in Chapters 7 and 10. With these evaluations we showed that our active learning-based RL technique is more efficient than complex supervised classification techniques, while being on par with or exceeding the linkage quality achieved with state-of-the-art active learning, deep learning, and machine learning RL approaches.

- **An Evaluation Technique for Group Record Linkage**: Subsequently, in Chapter 8 we discussed the ambiguous results which may be generated by existing evaluation techniques in the group RL context, and proposed a novel evaluation measure for group RL [126]. We assessed this group RL evaluation measure in Chapters 8 and 10, using the linkage result obtained by applying the three unsupervised clustering techniques (greedy, star, and robust) we proposed in Chapter 4 on two real-world and one synthetic data set. These assessments showed that our evaluation measure complements the results obtained with the F^*-measure while providing more insightful, unambiguous information on the distribution of predicted cluster sizes.

- **Graph Data Anonymisation for Record Linkage**: Finally, in Chapter 9 we introduced a new graph anonymisation technique which can be utilised in the RL domain to ensure the privacy of linked data, while preserving the structure of the original graph and ensuring the human interpretability of the graph

data [124]. With experiments conducted on two real-world and one synthetic data set in Chapters 9 and 10, we showed that our proposed graph anonymisation method successfully preserves the human interpretability and the structure of the original graph data. Furthermore, we applied the three unsupervised graph clustering techniques we proposed in Chapter 4 on both the original sensitive and anonymised graph data sets, and showed that the linkage results obtained with both data sets are similar.

11.3 Research Findings

In this section, we provide a summary of the main research findings of this thesis, based on the RL methods we have proposed and the experiments conducted in Chapters 4 to 10. In Chapter 4, we presented three unsupervised clustering techniques which utilise population data characteristics (such as temporal and spatial constraints), which was further developed in Chapter 5 to consider patterns (or transition probability distributions) corresponding to data characteristic. With empirical evaluations we showed that incorporating data characteristics can help achieve significant improvements in linkage quality with no decline in average efficiency. However, developing this approach to consider patterns in data characteristics was prone to errors in the presence of data quality issues, and therefore we do not recommend the method proposed in Chapter 5 to be used for real-world population RL applications.

We showed that the active learning-based graph filtering method we proposed in Chapter 6 significantly improves the efficiency and the overall quality of the RL process compared to conducting RL with no filtering. However, the advanced data dimension binning-based filtering method that we proposed in this chapter did not show noteworthy improvements over the simple global threshold based filtering approach. This indicates that even though active learning-based graph filtering (where the global threshold is set based on domain knowledge) is useful, allocating more resources for the binning approach may not be justifiable in real-world RL applications due to the potential efficiency improvements being limited.

While active learning has been previously applied for RL tasks, we explored this further in Chapter 7 due to the related literature being relatively novel and limited. We showed how our proposed active learning method achieves linkage quality on par with or superior to existing state-of-the-art supervised and semi-supervised RL approaches. The evaluation measure we proposed for assessing group RL techniques in Chapter 8 is more robust and unambiguous compared to existing evaluation measures such as precision and recall. The discussion in this chapter emphasises on the need for further research to be conducted on the topic of evaluation measures for RL.

In Chapter 9 we proposed an anonymisation technique which preserves the structure and human interpretability of data to be linked unlike in existing anonymisation methods. It is important to further develop anonymisation methods for RL applications to address their inadequacy in the literature. We further verified these findings with the experiments conducted on a new population data set in Chapter 10.

11.4 Future Work

Based on the exploration of the literature and the contributions we have made, we have identified the following as potential directions for future work in the RL domain.

- **Efficiency Enhancement of the RL Process**: While the majority of previous related RL research work has focused on improving the efficiency of the comparison step in RL, in this thesis we aimed to enhance the RL classification efficiency. Therefore, in Chapters 6 and 7 we proposed methods to reduce the number of record pairs considered in the classification step of RL. While these techniques were shown to be successful, further research on improving the efficiency of the overall RL process is valuable. Furthermore, since we mainly focused on improving the linkage quality of the unsupervised clustering methods we proposed in Chapter 4, we can further explore how the efficiency of techniques such as robust graph clustering can be enhanced. Research on the parallelisation of RL algorithms and techniques is also of significant value for improving the efficiency of the linkage process.

- **Learning Data Characteristics**: In the techniques we proposed in Chapters 4 and 5, we assumed that the constraints implied by data characteristics and their probability distributions can be modeled based on the knowledge of a domain expert. However, in certain RL projects this information may not be fully available or it might be expensive to entirely rely on human expertise for such constraint modelling. Therefore, it is useful to explore how we can employ semi-supervised methods such as active learning to model constraints related to data characteristics when it is expensive to consult domain experts.

- **Automatic Identification of Optimal Parameters**: The RL methods we proposed in Chapters 4 to 7 have several parameter settings where the optimal configurations need to be identified with experimental evaluation. Even though we were able to identify the best parameter configuration for some of our approaches irrespective of the data set used, for the remaining RL techniques we proposed, the optimal parameter settings are data set dependent. In a real-world RL application it is useful to have a mechanism of automatically identifying the best parameter configurations [135] and therefore supervised and semi-supervised techniques for automatic parameter tuning can be explored as future work.

- **Population Goodness-based Evaluation Measure**: In Chapter 5 we proposed a method of modelling the *goodness* of a population using the probability distributions related to data characteristics. We then used this population goodness measure to calculate the goodness of a set of overlapping clusters (cluster goodness) generated by a RL clustering technique, and used this cluster goodness to resolve overlaps. Even though this approach did not result in much improved

linkage quality, this population goodness measure can be adapted as an evaluation measure for population RL. Such an evaluation measure would reflect how closely a cluster prediction resembles the population distributions in the real-world from a related domain.

- **Improve the New Group Record Linkage Evaluation Measure**: In the novel group RL evaluation measure we introduced in Chapter 8, we used a greedy cluster mapping approach to map predicted clusters to ground-truth clusters. This approach can be further improved to dynamically map clusters, such that the globally optimal cluster mapping is guaranteed. However, the efficiency of the evaluation measure too needs to be improved when a dynamic approach is followed. Furthermore, it is valuable to develop unsupervised methods to evaluate group RL approaches where ground-truth data is not required [53].

- **Incremental Record Linkage**: The population data sets we have considered in this thesis are static historical data sets, where record attribute values, and the data set sizes stay constant throughout the linkage process. However, an important topic for future work would be incremental RL [79, 170], where the goal is to integrate data sources in an effective and efficient way as more records are added to them or the existing records are updated. Since real-world data sets, such as COVID-19 vaccination history data sets, are frequently updated, exploring incremental RL techniques is of paramount importance.

11.5 Conclusion

In this thesis we have presented our contributions to the domain of RL where we address limitations in existing solutions. We initially conducted a comprehensive literature review to study the state-of-the-art RL techniques and to identify the research problems, which we addressed with our proposed novel techniques for RL. We initially proposed three novel unsupervised graph clustering techniques utilising data characteristics and showed that incorporating data characteristics helps enhance the linkage quality of unsupervised RL clustering approaches with an empirical evaluation. Our attempt to further develop this concept to consider the transition probabilities related to data characteristics was shown to be ineffective in real-world applications where data can be error prone.

Another significant contribution we made in this thesis is the proposal of novel methods to improve the efficiency of the RL classification step. We proposed an active learning-based filtering technique to remove the likely non-matching record pairs, and another iterative active learning-based RL classification approach, where we classify record pairs while reducing the number of record pairs to be considered for classification in future iterations. We also proposed a novel evaluation measure for group RL techniques since existing RL evaluation measures can sometimes produce ambiguous results. Furthermore, we introduced a new graph anonymisation technique which can be used in the RL context to ensure the privacy of linked data.

With a comprehensive empirical evaluation, we have shown that all our proposed methods, except the method utilising transition probabilities on data characteristics, can be used to conduct real-world RL tasks in an efficient and effective manner.

References

1. Aggarwal, C. C., and Yu, P. S. *Privacy-preserving data mining: models and algorithms*, vol. 34 of *Advances in Database Systems*. Springer, 2008. (cited on page 57)

2. Akgün, Ö., Dearle, A., Kirby, G., and Christen, P. Using metric space indexing for complete and efficient record linkage. In *PAKDD* (Melbourne, 2018), pp. 89–101. (cited on pages 49 and 51)

3. Antonie, L., Grewal, G., Inwood, K., and Zarti, S. Automatic household identification for historical census data. In *Advances in Artificial Intelligence* (2017), M. Mouhoub and P. Langlais, Eds. (cited on pages 14, 33, 38, and 61)

4. Antonie, L., Inwood, K., Lizotte, D. J., and Ross, J. A. Tracking people over time in 19th century Canada for longitudinal analysis. *Machine Learning 95* (2014), 129–146. (cited on pages 14 and 25)

5. Arasu, A., Götz, M., and Kaushik, R. On active learning of record matching packages. In *ACM SIGMOD* (Indianapolis, 2010), pp. 783–794. (cited on pages 45, 98, 116, 118, and 119)

6. Aslam, J. A., Pelekhov, E., and Rus, D. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applications 8*, 1 (2004), 95–129. (cited on page 34)

7. Bailey, M., Cole, C., et al. How well do automated methods perform in historical samples? Evidence from new ground truth. Tech. rep., NBER, 2017. (cited on pages xxii and 14)

8. Bansal, N., Blum, A., and Chawla, S. Correlation clustering. *Machine Learning 56*, 1 (Jul 2004), 89–113. (cited on pages 34 and 35)

9. Bansal, N., Chiang, F., Koudas, N., and Tompa, F. Seeking stable clusters in the blogosphere. In *VLDB Endowment* (2007), p. 806–817. (cited on page 35)

10. Barlaug, N., and Gulla, J. A. Neural networks for entity matching: A survey. *ACM Trans. Knowl. Discov. Data 15*, 3 (Apr. 2021). (cited on page 117)

11. Batini, C., and Scannapieco, M. *Data quality: Concepts, methodologies and techniques*. Data-Centric Systems and Applications. Springer, 2006. (cited on page 183)

12. Baxter, R., Christen, P., and Churches, T. A comparison of fast blocking methods for record linkage. In *ACM SIGKDD Workshop on Data Cleaning, Record Linkage and Object Consolidation* (Washington DC, 2003), pp. 25–27. (cited on pages 17 and 183)

13. Bellahsene, Z., Bonifati, A., and Rahm, E. *Schema Matching and Mapping*. Data-Centric Systems and Applications. Springer, 2011. (cited on page 16)

14. Bellare, K., Iyengar, S., Parameswaran, A. G., and Rastogi, V. Active sampling for entity matching. In *ACM SIGKDD* (Beijing, 2012), pp. 1131–1139. (cited on page 45)

15. Bhattacharya, I., and Getoor, L. Collective entity resolution in relational data. *ACM TKDD 1*, 1 (2007). (cited on pages 3, 97, and 118)

16. Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006. (cited on page 40)

17. Bloothooft, G., Christen, P., Mandemakers, K., and Schraagen, M. *Population Reconstruction*. Springer, 2015. (cited on pages 2, 62, 135, and 152)

18. Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics 5* (2017), 135–146. (cited on pages 45 and 50)

19. Bomze, I. M., Budinich, M., Pardalos, P. M., and Pelillo, M. The maximum clique problem. In *Handbook of combinatorial optimization*. Springer, 1999, pp. 1–74. (cited on page 36)

20. Bonaccorso, G. *Machine Learning Algorithms: A Reference Guide to Popular Algorithms for Data Science and Machine Learning*. Packt Publishing, 2017. (cited on page 126)

21. Bondy, J. A., Murty, U. S. R., et al. *Graph theory with applications*, vol. 290. Macmillan London, 1976. (cited on page 71)

22. Bordes, A., Ertekin, S., Weston, J., and Bottou, L. Fast kernel classifiers with online and active learning. *JMLR 6*, 54 (2005), 1579–1619. (cited on page 132)

23. Boyd, K., Eng, K. H., and Page, C. D. Area under the precision-recall curve: Point estimates and confidence intervals. In *Proceedings of the 2013th European Conference on Machine Learning and Knowledge Discovery in Databases* (Berlin, Heidelberg, 2013), ECMLPKDD'13, Springer-Verlag, p. 451–466. (cited on page 22)

24. Buerli, M., and Obispo, C. The current state of graph databases. *Department of Computer Science, Cal Poly, San Luis Obispo 32*, 3 (2012), 67–83. (cited on page 151)

25. Caldeira, L., and Ferreira, A. Improvements in the blocking process for entity resolution based on the term relevance. In *Brazilian Symposium on Databases (SBBD)* (2018), pp. 61–72. (cited on page 52)

26. Caldeira, L. S., Bianco, G. D., and Ferreira, A. A. Experimental evaluation among reblocking techniques applied to the entity resolution. In *European Conference on Advances in Databases and Information Systems* (2021), pp. 229–243. (cited on pages 49, 51, and 52)

27. Campan, A., and Truta, T. A clustering approach for data and structural anonymity in social networks. In *ACM SIGKDD Workshop on Privacy, Security, and Trust in KDD (PinKDD)* (2008), vol. 5456 of *Lecture Notes in Computer Science*, Springer, pp. 33–54. (cited on page 56)

28. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., and Tzoumas, K. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 36*, 4 (2015). (cited on page 35)

29. Chen, H., Chung, W., Xu, J., Wang, G., Qin, Y., and Chau, M. Crime data mining: a general framework and some examples. *Computer 37*, 4 (2004), 50–56. (cited on page 2)

30. Chen, X., Xu, Y., Broneske, D., Durand, G. C., Zoun, R., and Saake, G. Heterogeneous committee-based active learning for entity resolution (healer). In *European Conference on Advances in Databases and Information Systems (ADBIS)* (2019), Springer, pp. 69–85. (cited on page 48)

31. Chierichetti, F., Dalvi, N., and Kumar, R. Correlation clustering in mapreduce. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2014). (cited on page 35)

32. Christen, P. A comparison of personal name matching: Techniques and practical issues. Tech. Rep. TR-CS-07-03, Department of Computer Science, The Australian National University, Canberra, Australia, 2006. (cited on page 17)

33. Christen, P. Automatic training example selection for scalable unsupervised record linkage. In *PAKDD* (Osaka, 2008), Springer, pp. 511–518. (cited on page 20)

34. Christen, P. Febrl: An open source data cleaning, deduplication and record linkage system with a graphical user interface. In *ACM SIGKDD* (Las Vegas, 2008), pp. 1065–1068. (cited on page 18)

35. Christen, P. *Data Matching – Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Springer, Heidelberg, 2012. (cited on pages 1, 3, 4, 5, 7, 10, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 37, 38, 39, 40, 76, 96, 97, 116, 118, 137, 138, 141, 155, 157, 158, and 184)

36. Christen, P. A survey of indexing techniques for scalable record linkage and deduplication. *Transactions on Knowledge and Data Engineering 24*, 9 (2012), 1537–1555. (cited on page 183)

37. Christen, P. Application of advanced record linkage techniques for complex population reconstruction. *ArXiv abs/1612.04286* (2016). (cited on page 3)

38. Christen, P. Data linkage: The big picture. *Harvard Data Science Review 1*, 2 (2019). (cited on page 16)

39. Christen, P., Hegland, M., Roberts, S., Nielsen, O., Churches, T., Lim, K., and Branch, S. Parallel computing techniques for high-performance probabilistic record linkage. *Symposium on Health Data Linkage* (04 2002). (cited on page 8)

40. Christen, P., Ranbaduge, T., and Schnell, R. *Linking Sensitive Data*. Springer, Heidelberg, 2020. (cited on pages 3, 4, 6, 8, 55, and 183)

41. Christen, P., Vatsalan, D., and Wang, Q. Efficient entity resolution with adaptive and interactive training data selection. In *IEEE ICDM* (Atlantic City, 2015), pp. 727–732. (cited on pages 116, 118, 123, and 130)

42. Christen, V., Christen, P., and Rahm, E. Informativeness-based active learning for entity resolution. In *PKDD/ECML DINA* (Würzburg, 2019). (cited on pages 44, 45, and 116)

43. Christen, V., Gross, A., Fisher, J., Wang, Q., Christen, P., and Rahm, E. Temporal group linkage and evolution analysis for census data. In *EDBT* (Venice, Italy, 2017), pp. 620–631. (cited on pages 2, 3, 14, 21, 27, and 151)

44. Ciaccia, P., Patella, M., Rabitti, F., and Zezula, P. Indexing metric spaces with m-tree. In *Italian Symposium on Advanced Database Systems (SEBD)* (1997), vol. 97, p. 67–86. (cited on page 51)

45. Cormode, G., Srivastava, D., Yu, T., and Zhang, Q. Anonymizing bipartite graph data using safe groupings. *VLDB Endowment 1* (2008), 833–844. (cited on page 56)

46. Das, S., Egecioglu, O., and Abbadi, A. Anónimos: An LP-based approach for anonymizing weighted social network graphs. *IEEE TKDE 24*, 4 (2012), 590–604. (cited on page 152)

47. Das, S., G.C., P. S., Doan, A., Naughton, J. F., Krishnan, G., Deep, R., Arcaute, E., Raghavendra, V., and Park, Y. Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD)* (New York, NY, USA, 2017), SIGMOD '17, Association for Computing Machinery, p. 1431–1446. (cited on page 41)

48. Davis, C. The norm of the schur product operation. *Numerische Mathematik 4*, 1 (Dec 1962), 343–344. (cited on page 41)

49. Davis, J., and Goadrich, M. The relationship between Precision-Recall and ROC curves. In *ACM ICML* (Pittsburgh, 2006), pp. 233–240. (cited on page 10)

50. Dean, J., and Ghemawat, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM 51*, 1 (2008), 107–113. (cited on page 49)

51. Delanaux, R., Bonifati, A., Rousset, M., and Thion, R. Rdf graph anonymization robust to data linkage. In *Web Information Systems Engineering (WISE)* (2019), vol. 11881 of *Lecture Notes in Computer Science*, Springer, pp. 491–506. (cited on pages 7, 55, 58, and 59)

52. Dillon, L. Y. Integrating nineteenth-century Canadian and American census data sets. *Computers and the Humanities 30*, 5 (1996), 381–392. (cited on page 14)

53. Doidge, J. C., and Harron, K. L. Reflections on modern methods: linkage error bias. *International Journal of Epidemiology 48*, 6 (2019), 2050–2060. (cited on pages 21 and 188)

54. Dong, X. L. Challenges and innovations in building a product knowledge graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2018), KDD '18, Association for Computing Machinery, p. 2869. (cited on page 2)

55. Dong, X. L., and Rekatsinas, T. Data integration and machine learning: A natural synergy. In *Proceedings of the 2018 International Conference on Management of Data* (New York, NY, USA, 2018), SIGMOD '18, Association for Computing Machinery, p. 1645–1650. (cited on pages 20 and 39)

56. Dong, X. L., and Srivastava, D. *Big Data Integration*. Synthesis Lectures on Data Management. Morgan and Claypool Publishers, 2015. (cited on pages 3 and 115)

57. Dongen, S. v. *Graph Clustering By Flow Simulation*. PhD thesis, PhD thesis, University of Utrecht, 2000. (cited on page 35)

58. Draisbach, U., Christen, P., and Naumann, F. Transforming pairwise duplicates to entity clusters for high-quality duplicate detection. *ACM JDIQ 12*, 1 (2019), 1–30. (cited on pages 5, 20, 33, 36, 96, 97, 116, and 118)

59. Dunn, H. Record linkage. *American Journal of Public Health 36*, 12 (1946), 1412. (cited on pages 1 and 13)

60. Ebraheem, M., Thirumuruganathan, S., Joty, S. R., Ouzzani, M., and Tang, N. Distributed representations of tuples for entity resolution. *VLDB Endowment 11* (2018), 1454–1467. (cited on pages 39, 40, 41, 51, and 61)

61. Efthymiou, V., Papadakis, G., Papastefanatos, G., Stefanidis, K., and Palpanas, T. Parallel meta-blocking for scaling entity resolution over big heterogeneous data. *Information Systems 65* (2017), 137–157. (cited on pages 4, 49, and 95)

62. El-Shalakani, M., and Pandey, A. Distribution of births in an abrupt sequence: a stochastic model. *Mathematical biosciences 95*, 1 (1989), 1–11. (cited on page 84)

63. Feder, T., Nabar, S., and Terzi, E. Anonymizing graphs. *arXiv Preprint* (2008). (cited on page 152)

64. Fellegi, I. P., and Sunter, A. B. A theory for record linkage. *Journal of the American Statistical Association 64*, 328 (1969), 1183–1210. (cited on pages 13 and 14)

65. Flake, G., Tarjan, R., and Tsioutsiouliklis, K. Graph clustering and minimum cut trees. *Internet Mathematics 1* (01 2003). (cited on page 35)

66. Ford, L. R., and Fulkerson, D. R. Maximal flow through a network. *Canadian Journal of Mathematics 8* (1956), 399–404. (cited on page 35)

67. Fu, Z. *Linking Historical Census Data Across Time*. PhD thesis, The Australian National University, 2014. (cited on page 27)

68. Fu, Z., Boot, M., Christen, P., and Zhou, J. Automatic record linkage of individuals and households in historical census data. *International Journal of Humanities and Arts Computing* (2014). (cited on pages 2 and 3)

69. Fu, Z., Christen, P., and Boot, M. A supervised learning and group linking method for historical census household linkage. In *AusDM, CRPIT* (Ballarat, Australia, 2011), vol. 125. (cited on page 20)

70. Fu, Z., Christen, P., and Zhou, J. A graph matching method for historical census household linkage. In *PAKDD* (Tainan, Taiwan, 2014). (cited on pages 3, 14, and 25)

71. Fu, Z., Zhou, J., Christen, P., and Boot, M. Multiple instance learning for group record linkage. In *PAKDD* (Kuala Lumpur, 2012). (cited on page 21)

72. Gagniuc, P. A. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017. (cited on page 85)

73. García, S., Luengo, J., and Herrerar, F. *Data Preprocessing in Data Mining*. Intelligent Systems Reference Library. Springer, 2015. (cited on page 15)

74. Gill, L. *Methods for Automatic Record Matching and Linkage and Their Use in National Statistics*. No. 25 in National statistics methodology series. Office for National Statistics, London, 2001. (cited on page 1)

75. Gokhale, C., Das, S., Doan, A., Naughton, J. F., Rampalli, N., Shavlik, J., and Zhu, X. Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM International Conference on Management of Data (SIGMOD)* (New York, NY, USA, 2014), SIGMOD '14, Association for Computing Machinery, p. 601–612. (cited on page 41)

76. Gottapu, R. D., Dagli, C., and Ali, B. Entity resolution using convolutional neural network. In *Conference Organized by Missouri University of Science and Technology* (Los Angeles, CA, 2016), pp. 153–158. (cited on pages 39 and 61)

77. Gravano, L., Ipeirotis, P. G., Jagadish, H. V., Koudas, N., Muthukrishnan, S., and Srivastava, D. Approximate string joins in a database (almost) for free. In *VLDB* (Roma, 2001), pp. 491–500. (cited on page 49)

78. Gravano, L., Ipeirotis, P. G., Koudas, N., and Srivastava, D. Text joins in an rdbms for web data integration. In *WWW* (2003). (cited on page 37)

79. Gruenheid, A., Dong, X. L., and Srivastava, D. Incremental record linkage. *Proc. VLDB Endow. 7*, 9 (2014), 697–708. (cited on page 188)

80. Gu, L., and Baxter, R. Adaptive filtering for efficient record linkage. In *SIAM international conference on data mining* (Orlando, Florida, 2004). (cited on pages 49 and 52)

81. Hamm, N. C., Hamad, A. F., Wall-Wieler, E., Roos, L. L., Plana-Ripoll, O., and Lix, L. M. Multigenerational health research using population-based linked databases: An international review. *International Journal of Population Data Science 6*, 1 (2021). (cited on page 15)

82. Han, J., Kamber, M., and Pei, J. *Data mining: concepts and techniques*, 3 ed. Morgan Kaufmann, 2012. (cited on pages 13, 20, 100, and 184)

83. Hand, D., Christen, P., and Kirielle, N. F*: an interpretable transformation of the f-measure. *Machine Learning 110* (03 2021), 451–456. (cited on pages xxii, 23, and 55)

84. Hand, D. J. Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine learning 77*, 1 (2009), 103–123. (cited on page 144)

85. Hand, D. J., and Christen, P. A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing 28*, 3 (2018). (cited on pages xxii, 5, 10, 22, 53, 54, 137, 138, and 184)

86. Hassanzadeh, O., Chiang, F., Lee, H., and Miller, R. Framework for evaluating clustering algorithms in duplicate detection. *VLDB 2*, 1 (2009). (cited on pages 20, 33, 34, 35, 37, 44, 53, 54, 61, 62, 64, 97, 118, 135, 137, 138, 143, and 158)

87. Hassanzadeh, O., and Miller, R. Probabilistic management of duplicated data. Tech. Rep. Technical Report CSRG-568, University of Toronto, Toronto, 2007. (cited on page 34)

88. Haveliwala, T., Gionis, A., and Indyk, P. Scalable techniques for clustering the web (extended abstract). In *Proceedings of the International Workshop on the Web and Databases (WebDB)* (2000), p. 129–134. (cited on page 34)

89. Hay, M., Miklau, G., Jensen, D., Towsley, D., and Weis, P. Resisting structural identification in anonymised social networks. *VLDB Endowment 1* (2008), 102–114. (cited on page 56)

90. Hay, M., Miklau, G., Jensen, D., Weis, P., and Srivastava, S. Anonymizing social networks. Tech. Rep. Technical Report 07-19, University of Massachusetts, Amherst, 2007. (cited on page 57)

91. Heineman, G., Pollice, G., and Selkow, S. *Algorithms in a Nutshell*. O'Reilly Media, Inc., 2008. (cited on page 125)

92. Hernández, M., Koutrika, G., Krishnamurthy, R., Popa, L., and Wisnesky, R. Hil: A high-level scripting language for entity integration. In *EDBT* (New York, NY, USA, 2013), Association for Computing Machinery, p. 549–560. (cited on page 3)

93. Herzog, T., Scheuren, F., and Winkler, W. *Data quality and record linkage techniques*. Springer Verlag, 2007. (cited on pages 2, 16, 20, and 183)

94. Hu, Y., Wang, Q., Vatsalan, D., and Christen, P. Improving temporal record linkage using regression classification. In *PAKDD* (2017), Springer, pp. 561–573. (cited on pages 96 and 97)

95. Jaro, M. A. Advances in record-linkage methodology a applied to matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association 84* (1989), 414–420. (cited on pages 18 and 19)

96. Ji, S., Mittal, P., and Beyah, R. Graph data anonymization, de-anonymization attacks, and de-anonymizability quantification: A survey. *IEEE Communications Surveys & Tutorials 19*, 2 (2016), 1305–1326. (cited on page 151)

97. Jiang, H., Gurajada, S., Lu, Q., Neelam, S., Popa, L., Sen, P., Li, Y., and Gray, A. LNN-EL: A neuro-symbolic approach to short-text entity linking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (Online, 2021), Association for Computational Linguistics. (cited on page 19)

98. Jones, K. H., and Ford, D. V. Population data science: advancing the safe use of population data for public benefit. *Epidemiology and health 40* (2018). (cited on page 8)

99. Karapiperis, D., Gkoulalas-Divanis, A., and Verykios, V. S. Lshdb: A parallel and distributed engine for record linkage and similarity search. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)* (2016), pp. 1–4. (cited on page 8)

100. Kasai, J., Qian, K., Gurajada, S., Li, Y., and Popa, L. Low-resource deep entity resolution with transfer and active learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 5851–5861. (cited on pages 44 and 45)

101. Kelman, C. W., Bass, J., and Holman, D. Research use of linked health data – A best practice protocol. *Aust NZ Journal of Public Health 26* (2002), 251–255. (cited on page 15)

102. Kirielle, N., Christen, P., and Ranbaduge, T. Outlier detection based accurate geocoding of historical addresses. In *Australasian Conference on Data Mining* (Adelaide, 2019), Springer, pp. 41–53. (cited on pages 62, 97, and 103)

103. Konda, P., Das, S., Suganthan G. C., P., Doan, A., Ardalan, A., Ballard, J. R., Li, H., Panahi, F., Zhang, H., Naughton, J., Prasad, S., Krishnan, G., Deep, R., and Raghavendra, V. Magellan: Toward building entity matching management systems. *Proc. VLDB Endow. 9*, 12 (Aug. 2016), 1197–1208. (cited on pages 20, 39, 40, 41, 42, 45, 61, and 130)

104. Köpcke, H., Thor, A., and Rahm, E. Evaluation of entity resolution approaches on real-world match problems. *VLDB Endowment 3*, 1-2 (2010), 484–493. (cited on pages 29 and 41)

105. Kouki, P., Pujara, J., Marcum, C., Koehly, L., and Getoor, L. Collective entity resolution in familial networks. In *2017 IEEE International Conference on Data Mining (ICDM)* (2017), pp. 227–236. (cited on pages 33, 38, 39, 61, and 62)

106. Kum, H.-C., Krishnamurthy, A., Machanavajjhala, A., and Ahalt, S. C. Social genome: Putting big data to work for population informatics. *Computer 47*, 1 (2014). (cited on page 2)

107. Leskovec, J., Rajaraman, A., and Ullman, J. D. *Mining of massive datasets*. Cambridge University Press, 2014. (cited on pages 17, 41, 52, and 96)

108. Li, P., Dong, X., Maurino, A., and Srivastava, D. Linking temporal records. *VLDB Endowment 4*, 11 (2011). (cited on pages 44 and 61)

109. Li, Y., Li, J., Suhara, Y., Doan, A., and Tan, W.-C. Deep entity matching with pre-trained language models. *Proc. VLDB Endow. 14*, 1 (2020), 50–60. (cited on pages 39 and 42)

110. Liu, K., and Terzi, E. Towards identity anonymization on graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'08)* (2008), pp. 93–106. (cited on page 57)

111. Liu, L., Wang, J., Liu, J., and Zhang, J. Privacy preserving in social networks against sensitive edge disclosure. Tech. Rep. Technical Report CMIDA-HiPSCCS 006-08, University of Kentucky, Department of Computer Science, Kentucky, 2008. (cited on page 57)

112. Lo, W.-C., Wang, F.-C., Lin, L.-Y., Jyan, H.-W., Wu, H.-C., Huang, Y.-L., Parng, I.-M., and Chiou, H.-Y. Enhancing data linkage to break the chain of covid-19 spread: The taiwan experience. *Journal of Medical Internet Research 23*, 5 (May 2021). (cited on page 15)

113. Lu, C., Huang, G., and Xiang, Y. Community enhanced record linkage method for vehicle insurance system. In *Advanced Data Mining and Applications* (2019), pp. 761–776. (cited on page 15)

114. McGrail, K., and Jones, K. Population data science: The science of data about people. *International Journal of Population Data Science 3*, 4 (2018). (cited on page 8)

115. McGrail, K., Jones, K., Akbari, A., Bennett, T. D., Boyd, A., et al. A position statement on population data science: The science of data about people. *International Journal of Population Data Science 3*, 1 (2018). (cited on page 8)

116. Meduri, V. V., Popa, L., Sen, P., and Sarwat, M. A comprehensive benchmark framework for active learning methods in entity matching. In *ACM SIGMOD* (New York, 2020), pp. 1133–1147. (cited on pages 44, 47, 130, 131, and 133)

117. Menestrina, D., Whang, S., and Garcia-Molina, H. Evaluating entity resolution results. *VLDB Endowment 3*, 1–2 (2010), 208–219. (cited on page 53)

118. Morgan, O., Aguilera, X., Ammon, A., Amuasi, J., Fall, I., Frieden, T., Heymann, D., Ihekweazu, C., kyeong Jeong, E., Leung, G., Mahon, B., Nkengasong, J., Qamar, F., Schuchat, A., Wieler, L., and Dowell, S. Disease surveillance for the covid-19 era: time for bold changes. *Lancet (London, England)* (2021). (cited on page 1)

119. Mozafari, B., Sarkar, P., Franklin, M., Jordan, M., and Madden, S. Scaling up crowd-sourcing to very large datasets: a case for active learning. *Proceedings of the VLDB Endowment 8*, 2 (2014), 125–136. (cited on page 46)

120. Mudgal, S., Li, H., Rekatsinas, T., Doan, A., et al. Deep learning for entity matching: A design space exploration. In *ACM SIGMOD* (Houston, 2018), pp. 19–34. (cited on pages 20, 29, 30, 39, 40, 41, 47, 61, and 130)

121. Munkova, D., Munk, M., and Vozar, M. Influence of stop-words removal on sequence patterns identification within comparable corpora. *Advances in Intelligent Systems and Computing 231* (01 2014), 67–76. (cited on pages 16 and 40)

122. Nanayakkara, C., Christen, P., and Ranbaduge, T. Temporal graph-based clustering for historical record linkage. In *MLG, held at ACM SIGKDD* (London, 2018). (cited on pages 143 and 184)

123. Nanayakkara, C., Christen, P., and Ranbaduge, T. Robust temporal graph clustering for group record linkage. In *PAKDD* (Macau, 2019). (cited on pages 143, 156, and 184)

124. Nanayakkara, C., Christen, P., and Ranbaduge, T. An anonymiser tool for sensitive graph data. In *International Workshop on Entity Retrieval and Learning (EYRE) co-located with CIKM* (2020). (cited on pages xxii and 186)

125. Nanayakkara, C., Christen, P., and Ranbaduge, T. Active learning based similarity filtering for efficient and effective record linkage. In *PAKDD* (Delhi, 2021). (cited on pages 116, 119, and 185)

126. Nanayakkara, C., Christen, P., Ranbaduge, T., and Garrett, E. Evaluation measure for group-based record linkage. *International Journal of Population Data Science 4*, 1 (2019). (cited on page 185)

127. Naumann, F., and Herschel, M. *An introduction to duplicate detection*. Synthesis Lectures on Data Management. Morgan and Claypool Publishers, 2010. (cited on page 22)

128. Navarro, G. A guided tour to approximate string matching. *ACM Computing Surveys 33*, 1 (2001), 31–88. (cited on pages 153, 157, 159, and 178)

129. Newcombe, H., and Kennedy, J. Record linkage: making maximum use of the discriminating power of identifying information. *Communications of the ACM 5*, 11 (1962), 563–566. (cited on page 13)

130. Newcombe, H., Kennedy, J., Axford, S., and James, A. Automatic linkage of vital records. *Science 130*, 3381 (1959), 954–959. (cited on page 13)

131. Newcombe, H. B. *Handbook of record linkage: methods for health and statistical studies, administration, and business*. Oxford University Press, Inc., New York, NY, USA, 1988. (cited on page 1)

132. Nguyen, L., Stoové, M., Boyle, D., Callander, D., McManus, H., Asselin, J., Guy, R., Donovan, B., Hellard, M., and El-Hayek, C. Privacy-preserving record linkage of deidentified records within a public health surveillance system: Evaluation study. *Journal of Medical Internet Research 22*, 6 (Jun 2020). (cited on page 15)

133. Odell, M., and Russell, R. The Soundex coding system. *US Patents 1261167* (1918). (cited on page 17)

134. On, B.-W., Koudas, N., Lee, D., and Srivastava, D. Group linkage. In *IEEE ICDE* (Istanbul, 2007), pp. 496–505. (cited on pages 2, 3, 21, 33, 37, 38, 62, and 135)

135. Paganelli, M., Del Buono, F., Marco, P., Guerra, F., and Vincini, M. Automated machine learning for entity matching tasks. In *EDBT* (2021). (cited on page 187)

136. Papadakis, G., Ioannou, E., Thanos, E., and Palpanas, T. The four generations of entity resolution. *Synthesis Lectures on Data Management 16*, 2 (2021), 1–170. (cited on page 115)

137. Papadakis, G., Papastefanatos, G., Palpanas, T., and Koubarakis, M. Scaling entity resolution to large, heterogeneous data with enhanced meta-blocking. In *EDBT* (2016), pp. 221–232. (cited on page 52)

138. Papadakis, G., Skoutas, D., Thanos, E., and Palpanas, T. Blocking and filtering techniques for entity resolution: A survey. *ACM CSUR 53*, 2 (2020), 1–42. (cited on pages 3, 4, 17, 49, 95, 96, 115, and 116)

139. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research 12*, Oct (2011), 2825–2830. (cited on page 126)

140. Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014). (cited on page 41)

141. Primpeli, A., and Bizer, C. Graph-boosted active learning for multi-source entity resolution (preprint). In *The 20th International Semantic Web Conference (ISWC)* (2021). (cited on pages 44 and 48)

142. Primpeli, A., Bizer, C., and Keuper, M. Unsupervised bootstrapping of active learning for entity resolution. In *ESWC* (Crete, 2020), pp. 215–231. (cited on pages 20, 44, 46, 48, 118, and 121)

143. Pyle, D. *Data preparation for data mining*. Morgan Kaufmann, 1999. (cited on page 183)

144. Qian, K., Chozhiyath Raman, P., Li, Y., and Popa, L. Learning structured representations of entity names using active learning and weak supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Online, 2020), Association for Computational Linguistics, pp. 6376–6383. (cited on pages 116 and 117)

145. Qian, K., Popa, L., and Sen, P. Active learning for large-scale entity resolution. In *ACM CIKM* (Singapore, 2017), p. 1379–1388. (cited on pages 20, 116, and 118)

146. Rahm, E. Discovering product counterfeits in online shops: A big data integration challenge. *ACM Journal on Data and Information Quality 5*, 1–2 (2014). (cited on page 2)

147. Rastogi, V., Dalvi, N., and Garofalakis, M. Large-scale collective entity matching. *VLDB Endowment 4* (2011), 208–218. (cited on page 3)

148. Reas, R., Ash, S., Barton, R., and Borthwick, A. Superpart: Supervised graph partitioning for record linkage. In *2018 IEEE International Conference on Data Mining (ICDM)* (2018), pp. 387–396. (cited on pages 19, 20, and 152)

149. Reid, A., Davies, R., and Garrett, E. 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. (cited on pages 2, 24, 25, 26, 63, 84, 85, 96, and 118)

150. Richardson, M., and Domingos, P. Markov logic networks. *Machine Learning 62*, 1-2 (2006), 107–136. (cited on page 84)

151. Roddick, J. F., Hornsby, K., and de Vries, D. A unifying semantic distance model for determining the similarity of attribute values. In *Proceedings of the 26th Australasian computer science conference* (2003), vol. 16, pp. 111–118. (cited on page 62)

152. Ruggles, S., Fitch, C. A., and Roberts, E. Historical census record linkage. *Annual Review of Sociology 44*, 1 (2018), 19–37. (cited on pages xxii and 14)

153. Saeedi, A., Peukert, E., and Rahm, E. Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In *ADBIS* (Nicosia, 2017), pp. 278–293. (cited on pages 20, 33, 35, 36, 61, 62, 64, 68, 135, and 143)

154. Saeedi, A., Peukert, E., and Rahm, E. Using link features for entity clustering in knowledge graphs. In *ESWC* (Greece, 2018), pp. 576–592. (cited on pages 19, 20, 33, 36, 61, 62, 64, 70, 71, 97, 118, and 152)

155. Sarawagi, S., and Bhamidipaty, A. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2002), KDD '02, Association for Computing Machinery, p. 269–278. (cited on pages 43 and 116)

156. Schnell, R., Bachteler, T., and Reiher, J. Privacy-preserving record linkage using Bloom filters. *BMC Medical Informatics and Decision Making 9*, 1 (2009). (cited on page 6)

157. Schnell, R., Klingwort, J., and Farrow, J. M. Locational privacy-preserving distance computations with intersecting sets of randomly labeled grid points. *International Journal of Health Geographics 20*, 1 (2021), 1–16. (cited on page 161)

158. Settles, B. Active learning literature survey. Tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2009. (cited on page 46)

159. Silverman, B. W. *Density Estimation for Statistics and Data Analysis*. CRC Press, 1986. (cited on page 85)

160. Simonini, G., Bergamaschi, S., and Jagadish, H. Blast: a loosely schema-aware meta-blocking approach for entity resolution. *Proceedings of the VLDB Endowment 9*, 12 (2016), 1173–1184. (cited on page 52)

161. Srivastava, R. K., Greff, K., and Schmidhuber, J. Training very deep networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2* (Cambridge, MA, USA, 2015), NIPS'15, MIT Press, p. 2377–2385. (cited on page 45)

162. Sweeney, L. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge Based Systems 10*, 5 (2002), 557–570. (cited on pages 56 and 57)

163. Tamariz, L., Medina, H., Suarez, M., Seo, D., and Palacio, A. Linking census data with electronic medical records for clinical research: A systematic review. *Journal of Economic and Social Measurement 43* (2018), 1–14. (cited on page 1)

164. Tao, Y. Entity matching with active monotone classification. In *ACM PODS* (Houston, 2018), pp. 49–62. (cited on pages 98 and 119)

165. Tejada, S., Knoblock, C. A., and Minton, S. Learning object identification rules for information integration. *Inf. Syst. 26*, 8 (Dec. 2001), 607–633. (cited on page 43)

166. Teong, K.-S., Soon, L.-K., and Su, T. T. Schema-agnostic entity matching using pre-trained language models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM)* (New York, NY, USA, 2020), Association for Computing Machinery, p. 2241–2244. (cited on page 40)

167. Torrey, L., and Shavlik, J. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264. (cited on page 43)

168. Tsai, M.-H., Ho, C.-H., and Lin, C.-J. Active learning strategies using svms. In *The 2010 International Joint Conference on Neural Networks (IJCNN)* (2010), IEEE, pp. 1–8. (cited on page 46)

169. Van Hout-Wolters, B., and Simons, Robert-Jan, V. S. Active learning: Self-directed learning and independent work. In *New Learning*, R.-J. Simons, J. Van der Linden, and T. Duffy, Eds. Springer, 2000, ch. 2, pp. 21–36. (cited on pages 7 and 43)

170. Vatsalan, D., Christen, P., and Rahm, E. Incremental clustering techniques for multi-party privacy-preserving record linkage. *Data and Knowledge Engineering* (2020). (cited on page 188)

171. Wang, J., Li, G., and Feng, J. Can we beat the prefix filtering? an adaptive framework for similarity join and search. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data* (2012), pp. 85–96. (cited on page 52)

172. Wang, K., Zhang, D., Li, Y., Zhang, R., and Lin, L. Cost-effective active learning for deep image classification. *IEEE TCSVT 27* (2017), 2591–2600. (cited on page 118)

173. Wang, L.-E., and Li, X. A graph-based multifold model for anonymizing data with attributes of multiple types. *Computers and Security 72*, C (2018), 122–135. (cited on pages 7, 55, 57, and 58)

174. Wang, L.-E., and Li, X. A graph-based multifold model for anonymizing data with attributes of multiple types. *CaS 72* (2018), 122–135. (cited on page 152)

175. Wang, Q., Vatsalan, D., and Christen, P. Efficient interactive training selection for large-scale entity resolution. In *PAKDD* (Ho Chi Minh City, 2015), pp. 562–573. (cited on pages 45 and 46)

176. Wijaya, D. T., and Bressan, S. Ricochet: A family of unconstrained algorithms for graph clustering. In *Database Systems for Advanced Applications (DASFAA)* (2009), vol. 5463 of *Lecture Notes in Computer Science*, Springer, pp. 153–167. (cited on page 34)

177. Winkler, W. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods* (1990), American Statistical Association, pp. 354–359. (cited on pages 13 and 19)

178. Witten, I. H., Moffat, A., and Bell, T. C. *Managing Gigabytes*, 2 ed. Morgan Kaufmann, 1999. (cited on page 22)

179. Wrigley, E., and Schofield, R. Nominal record linkage by computer and the logic of family reconstitution. *Identifying people in the past* (1973), 64–101. (cited on pages 14 and 25)

180. Xiang, R., Neville, J., and Rogati, M. Modeling relationship strength in online social networks. In *WWW* (Raleigh, NC, 2010), ACM, pp. 981–990. (cited on page 154)

181. Xiao, C., Wang, W., Lin, X., Yu, J. X., and Wang, G. Efficient similarity joins for near-duplicate detection. *ACM Transactions on Database Systems 36*, 3 (2011), 1–41. (cited on page 52)

182. Yancey, W. E. Evaluating string comparator performance for record linkage. Tech. Rep. RR2005/05, US Bureau of the Census, 2005. (cited on page 19)

183. Ying, X., and Wu, X. Randomizing social networks: a spectrum preserving approach. In *In Proceedings of the SIAM International Conference on Data Mining (SDM'08)* (2008), p. 739–750. (cited on page 57)

184. Zhang, W., Wei, H., Sisman, B., Dong, X. L., Faloutsos, C., and Page, D. Autoblock: A hands-off blocking framework for entity matching. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (New York, NY, USA, 2020), WSDM '20, Association for Computing Machinery, p. 744–752. (cited on pages 18, 49, and 50)

185. Zheleva, E., and Getoor, L. Preserving the privacy of sensitive relationships in graph data. In *ACM SIGKDD Workshop on Privacy, Security, and Trust in KDD (PinKDD)* (2007), vol. 4890 of *Lecture Notes in Computer Science*, Springer, pp. 153–171. (cited on page 56)

186. Zhou, B., and Pei, J. Preserving privacy in social networks against neighborhood attacks. In *In Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE'08)* (2008), pp. 506–515. (cited on page 57)

187. Zhou, B., and Pei, J. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowledge and Information Systems 28* (2010), 47–77. (cited on page 57)

188. Zhou, B., Pei, J., and Luk, W. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explor. Newsl. 10*, 2 (2008), 12–22. (cited on pages 7, 55, 57, and 152)