Advanced Entity Resolution Techniques

Jeffrey Stewart Fisher

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

June 2020

Submitted – 18 June 2020

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

Jeffrey Stewart Fisher
18 June 2020

Submitted – 18 June 2020

For Ruth, Matthew and George

Submitted – 18 June 2020

Submitted – 18 June 2020

Acknowledgments

While there may only be my name on the first page of this thesis, there are many other people without whom it would never have happened. Throughout the time it has taken to finish, there have been numerous people who have provided advice, support, encouragement and criticism, all of which have been very important in it reaching its conclusion.

To my colleagues, Thilina, Dinusha, Minkyoung and Banda, I really appreciated your advice and encouragement and for passing on some of the lessons you learned. It was great to have people in a similar situation to ask when I was stuck or wanted advice on how to proceed.

To my friends Sam, Kris, Nige, Steph, Guy, Cat, and Stef, who I have constantly had to postpone, bail on, leave early, and otherwise neglect, I'm grateful for the encouragement and understanding and for the fact that you are all forgiving enough to still be my friends now this is over. Hopefully I can see a bit more of you all in the future.

To the school administrators and support staff who helped book travel, fix computers, organise offices and a whole host of other seemingly simple yet vital tasks, I'm really grateful. Special thanks to Janette, Cathy and Alistair. Your advice, guidance and support was invaluable when I was struggling to balance the conflicting demands of a PhD, course teaching, poor health and new parenthood.

To Swapnil and Anish, we started this Computer Science thing at ANU at the same time, and now we are all finishing at the same time. Having others going through the PhD journey along side me, and knowing there was someone there to commiserate with has made the whole thing more bearable. I'll miss our movie nights, catch-ups, and the *Shut Up and Write* sessions that brought this to a conclusion.

To my supervisors, Peter, Qing and Paul, no PhD thesis is possible without a strong panel providing advice. I am very grateful for your guidance, support and encouragement throughout. You each brought a very different perspective, and this lead to a much better thesis overall. Qing, I'm particularly grateful for your help in dragging this thing over the line at the end.

Thank you to my family who have supported me through the whole process, even if a few of you thought I was crazy (perhaps I was). Mick, Sim and Tony, you were all very good at helping

me keep a healthy perspective on things. Dad, I've appreciated your encouragement and mum, well, you have given me the same unconditional support and encouragement that you always do, not to mention the meals, cups of tea and a place to stay when I needed some time out. You also extended that to Ruth, Matthew and George and it made the long days and weekends easier to bear knowing that you were there looking after them.

Lastly, to my long suffering wife Ruth, and my sons Matthew and George. Knowing the burden this thesis placed on you all has been the hardest part of the journey. Despite this, you supported me throughout, even when it got really tough at the end. I am enormously grateful, and always will be. Thank you.

Abstract

Entity resolution is the task of determining which records in one or more data sets correspond to the same real-world entities. Entity resolution is an important problem with a range of applications for government agencies, commercial organisations, and research institutions. Due to the important practical applications and many open challenges, entity resolution is an active area of research and a variety of techniques have been developed for each part of the entity resolution process.

This thesis is about trying to improve the viability of sophisticated entity resolution techniques for real-world entity resolution problems. Collective entity resolution techniques are a subclass of entity resolution approaches that incorporate relationships into the entity resolution process and introduce dependencies between matching decisions. Group linkage techniques match multiple related records at the same time. Temporal entity resolution techniques incorporate changing attribute values and relationships into the entity resolution process. Population reconstruction techniques match records with different entity roles and very limited information in the presence of domain constraints. Sophisticated entity resolution techniques such as these produce good results when applied to small data sets in an academic environment. However, they suffer from a number of limitations which make them harder to apply to real-world problems. In this thesis, we aim to address several of these limitations with the goal that this will enable such advanced entity resolution techniques to see more use in practical applications.

One of the main limitations of existing advanced entity resolution techniques is poor scalability. We propose a novel size-constrained blocking framework, that allows the user to set minimum and maximum block-size thresholds, and then generates blocks where the number of records in each block is within the size range. This allows efficiency requirements to be met, improves parallelisation, and allows expensive techniques with poor scalability such as Markov logic networks to be used.

Another significant limitation of advanced entity resolution techniques in practice is a lack of training data. Collective entity resolution techniques make use of relationship information so a bootstrapping process is required in order to generate initial relationships. Many techniques for temporal entity resolution, group linkage and population reconstruction also require training

data. In this thesis we propose a novel approach for automatically generating high quality training data using a combination of domain constraints and ambiguity. We also show how we can incorporate these constraints and ambiguity measures into active learning to further improve the training data set.

We also address the problem of parameter tuning and evaluation. Advanced entity resolution approaches typically have a large number of parameters that need to be tuned for good performance. We propose a novel approach using transitive closure that eliminates unsound parameter choices in the blocking and similarity calculation steps and reduces the number of iterations of the entity resolution process and the corresponding evaluation.

Finally, we present a case study where we extend our training data generation approach for situations where relationships exist between records. We make use of the relationship information to validate the matches generated by our technique, and we also extend the concept of ambiguity to cover groups, allowing us to increase the size of the generated set of matches. We apply this approach to a very complex and challenging data set of population registry data and demonstrate that we can still create high quality training data when other approaches are inadequate.

Publications

Primary Publications

1. *A Clustering-Based Framework to Control Block Sizes for Entity Resolution*, **Fisher, J.**, Christen, P., Wang, Q. and Rahm, E., Proceedings of the 21st International Conference on Knowledge Discovery and Data Mining (KDD), 279–288. ACM. 2015. (Full paper)

2. *Unsupervised Measuring of Entity Resolution Consistency*, **Fisher, J.**, and Wang, Q., Proceedings of the 15th International Conference on Data Mining Workshop (ICDMW), 218–221. IEEE. 2015.

3. *Active Learning Based Entity Resolution Using Markov Logic*, **Fisher, J.**, Christen, P., and Wang, Q., Proceedings of the 20th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 338–349. Springer. 2016. (Full paper)

Secondary Publications

4. *Data Cleaning and Matching of Institutions in Bibliographic Databases*, **Fisher, J.**, Wang, Q., Wong, P., and Christen, P., Proceedings of the 11th Australasian Data Mining Conference (AusDM), 139–148. Australian Computer Society, 2013.

5. *Automatic Discovery of Abnormal Values in Large Textual Databases*, Christen, P., Gayler, R., Tran, K., **Fisher, J.**, and Vatsalan, D., Journal of Data and Information Quality (JDIQ), 7, 1-2(2016), 7. ACM, 2016.

6. *Temporal group linkage and evolution analysis for census data.*, Christen, V., Groß, A., **Fisher, J.**, Wang, Q., Christen, P., and Rahm, E., Proceedings of the 20th International Conference on Extending Database Technology (EDBT), 620 –631. 2017.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Entity resolution, also called data matching, record linkage, deduplication and several other names [62], is a common practical problem. It involves determining which records in one or more data sets correspond to the same real-world entities. In many cases entity resolution is a necessary pre-processing step to further data analysis tasks. Detecting such matching records in two or more different data sets, allows them to be linked and analysed jointly, which may provide insights that are not apparent when each data set is analysed in isolation [16]. Alternatively, detecting duplicate records within a single data set improves the accuracy and consistency of the data set and the validity of conclusions that are drawn from any subsequent analysis [15]. Due to its importance, entity resolution sees widespread use in application domains such as government agencies, commercial organisations and research institutions, for tasks including business intelligence [85], fraud and compliance [7], and demographic studies [10].

The entity resolution process (described in detail in Section 2.3) was given a mathematical formalisation by Felligi and Sunter [47] in the late 1960's. This seminal work formalised the notion that two records having similar attribute values was evidence that they might refer to the same real-world entity. It also presented the idea that different attributes had differing levels of importance in this decision making process. Subsequently, more sophisticated entity resolution techniques have been developed such as collective entity resolution techniques [12, 39, 86, 165], group linkage [34, 53, 54, 126], temporal entity resolution [21, 28, 106] and more. However, despite achieving good results in academic environments [12, 146], advanced techniques such as these suffer from a number of limitations that make them harder to apply in real-world situations. In this thesis we propose approaches to overcome several practical limitations of advanced entity resolution techniques in order to make them easier to use on real-world problems.

The remainder of this chapter is structured as follows: in Section 1.1 we present our thesis statement and state the problem we are trying to address. In Section 1.2 we motivate our

research and explain why our problem is important. In Section 1.3 we expand upon the thesis statement and provide our detailed research questions. In Section 1.4 we describe our approach to answering our research questions and the overall methodological framework used in this thesis. In Section 1.5 we describe the outcomes of this thesis and the techniques and approaches we have developed. Finally, in Section 1.6 we detail the structure of the thesis and briefly summarise the topics and content of each of the chapters.

1.1 Thesis Statement

Entity resolution is the problem of determining which records in one or more data sets refer to the same real-world entities. We define an *advanced entity resolution technique* as a technique which extends the traditional entity resolution approach by looking at the context of a record, including relationship information, classifying groups of records simultaneously, or incorporating temporal information. Such advanced entity resolution techniques have been shown to perform very well in academic experiments [12, 21, 53, 165]. However, there are many limitations that prevent advanced entity resolution techniques from being widely applied in practice. In this thesis we present techniques to address three particular limitations of advanced entity resolution techniques:

- The poor scalability of many advanced entity resolution techniques,
- The lack of training data that is required by many advanced entity resolution techniques, and
- The need to tune the many parameters of advanced entity resolution techniques without evaluation data.

1.2 Research Motivation

In this section we discuss the limitations of traditional entity resolution, and the approaches taken by more sophisticated entity resolution techniques to overcome these weaknesses. We also identify three limitations that prevent such advanced entity resolution techniques from seeing widespread practical application as a way of justifying the work done in this thesis.

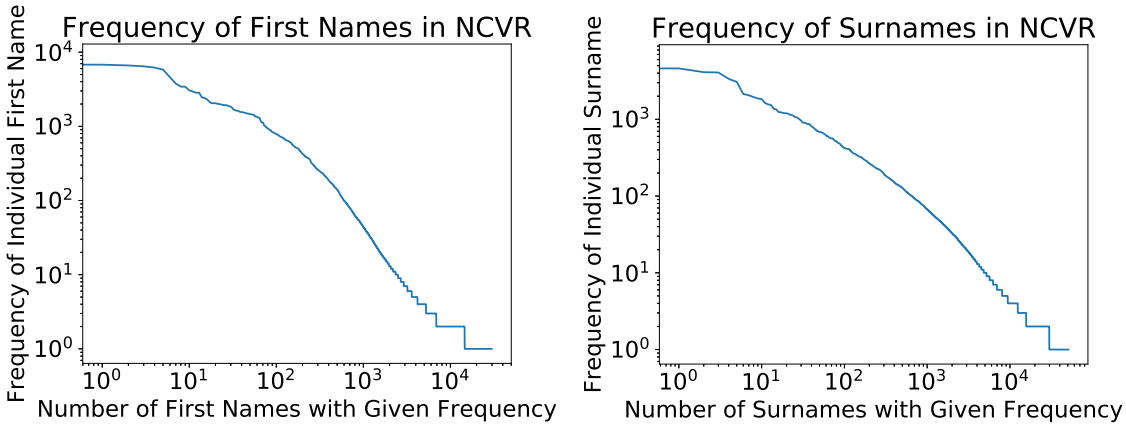

Figure 1.1: Frequencies of first names and surnames in NCVR-450. Both attributes are very skewed with a small number of very common values, and a large number of individually rare values.

1.2.1 Limitations of Traditional Entity Resolution

The basic ideas of traditional entity resolution were formalised by Felligi and Sunter [47] and still form an important part of most entity resolution approaches [23]. However, despite subsequent improvements in areas such as blocking and indexing, similarity calculations, classification and evaluation, traditional entity resolution suffers from a number of limitations that make it inadequate for certain types of data or in certain problem domains.

One limitation of traditional entity resolution is that it only considers attribute values in the entity resolution process. In some situations this may be sufficient, however for other problems skewness in the frequency of attribute values make it hard to distinguish between records with common attribute values. For example, even in clean data sets, distinguishing between different records with name 'John Smith' may be difficult. In some data sets or domains this can be a significant problem. In the UKCD data set used in this thesis (see Section 2.5) more than 1 in 25 people have the surname 'Ashworth' which makes it difficult to unambiguously match people with this surname.

This problem gets worse as data sets get larger, since more records are likely to have similar attribute values purely by chance. The NCVR-450 dataset (also described in Section 2.5) contains approximately 450 thousand records and the frequency plots of *first name* and *surname* are shown in Figure 1.1. Both distributions are highly skewed with the most common names occurring approximately 5,000 times.

One solution is to use multiple attributes for the entity resolution process, however combinations of attribute values can be repeated if the data sets are large enough. For example 'Jeffrey Fisher' is not a common name, however there are 63 records in the full NCVR data set with some variation of 'Jeffrey Fisher' as the name value ('Jeff', 'Geoff', 'Fischer', etc). These records could potentially be distinguished by introducing a third attribute, such as *address* or *date-of-birth*, but in practice, there are usually limits to the number of clean and reliable attributes available. Even when there are additional attributes, they may still be insufficient. In the full NCVR data set there are 6 different people named 'John Smith', aged '49' living in 'Charlotte'. In situations like this, a more sophisticated entity resolution technique may be necessary.

Another limitation of traditional entity resolution techniques is that match decisions are made independently. In traditional pairwise entity resolution, whether record pair $\langle r_i, r_j \rangle$ was classified as a match has no influence on whether record pair $\langle r_j, r_k \rangle$ will be classified as a match. However, if transitive closure holds (see Section 2.2), and it generally does, then classifying record pair $\langle r_i, r_j \rangle$ as a match and record pair $\langle r_j, r_k \rangle$ as a match, implicitly classifies record pair $\langle r_i, r_k \rangle$ as a match as well, and this fact should be incorporated into the classification process.

This situation extends to relationships. If the person represented by record r_i is married to the person represented by record r_j, then classifying the record pair $\langle r_i, r_k \rangle$ as a match, implies that the people represented by records r_k and r_j must also be married, and evidence supporting or contradicting this fact should be considered when classifying record pair $\langle r_i, r_k \rangle$.

As a result of the independence in matching decisions, traditional entity resolution can also lead to situations where the output is impossible, either because transitive closure does not hold in the output but must hold in the real-world, or because relationship information is contradictory. This can potentially be corrected in a post processing step [23], however sophisticated entity resolution techniques such as collective entity resolution can enforce transitive closure as part of the matching process as well as take relationship information into account [12, 165].

Finally, traditional entity resolution assumes entities are static, i.e. their qualities, represented by attribute values in the data set(s), do not change. However, for many types of entities this assumption does not reflect reality. For example, people can change almost every characteristic, including their name, address, age, occupation and gender. Even fixed characteristics such as date of birth can change as a result of errors or uncertainty. Where information about the likelihood of such changes is available, along with timestamp information for the records (i.e. when each record was created), it is possible to include this in the entity resolution process in a much more effective manner than that of traditional entity resolution [21, 106].

Temporal aspects can also be a source of logical constraints. For example, a person cannot die before they are born, so an individual's death certificate should have an equal or later date than their birth certificate. Traditional entity resolution is unable to capture such constraints in the matching process (or constraints from any other source) and so again, in order to prevent inconsistency in the results, more sophisticated entity resolution techniques have been developed [156, 165].

1.2.2 Characteristics of Advanced Entity Resolution Techniques

In order to overcome the limitations discussed in the previous section, more sophisticated entity resolution techniques have been developed. We briefly summarise the main characteristics of important advanced entity resolution techniques here and give a detailed description in Chapter 3.

Clustering techniques [71, 78, 151, 182] can incorporate transitive closure, thus introducing dependencies between matching decisions. The naïve relational approach of Bhattacharya and Getoor [12] incorporates relationships while still making independent matching decisions. However, collective entity resolution techniques [12, 39, 86, 165] incorporate both relationships and dependencies in matching decisions and it is this combination which allows them to produce good results in situations where traditional entity resolution may not work well.

Group linkage techniques [34, 53, 54, 126] take advantage of (possibly implicit) relationships between records and attempt to match records as group or sets. While the techniques are not applicable to every data set, they can be very effective for entity resolution problems where such groups structures do exist such as matching households between censuses (a household is a group of people) or product categories between internet sites (a category is a group of products).

Temporal entity resolution [21, 28, 106] takes advantage of time stamp information (i.e. the date the record was created) during the entity resolution process. This allows logical constraints to be enforced (such as a person not being born after their death), and also refinement of attribute weights and similarity scores based on the likelihood of each particular attribute changing.

Population reconstruction techniques [29, 42, 157] incorporate many of the concepts used in both group linkage and temporal entity resolution. Birth certificates, death certificates and marriage certificates all contain information about multiple people, thus allowing group linkage techniques to be applied. In addition, temporal constraints exist in the domain, sometimes as hard constraints (such as a person cannot be married before they are born), and sometimes as

soft constraints (there is usually at least a year between the births of two different children to the same mother). In addition, an individual record on one certificate can match to multiple different individuals on another certificate. For example, a baby boy on a birth certificate can match to the groom, father of the groom, or father of the bride, on a marriage certificate. In order to incorporate all these aspects, population reconstruction techniques often work in stages or iteratively, with initial results being refined through the addition of constraints or based on the outcome of other parts of the process.

1.2.3 Limitations of Advanced Entity Resolution Techniques

While advanced entity resolution techniques help overcome the limitations of traditional entity resolution, they present their own problems, particularly when applied to real-world problems.

- **Scalability**: Poor scalability is a problem for entity resolution in general. The traditional entity resolution problem is quadratic, i.e. $O(m * n)$ where m and n are the numbers of records in the two data sets being compared. As a result, even early entity resolution techniques employed blocking to reduce the necessary search space [47]. However, advanced entity resolution techniques often have much poorer scalability than traditional entity resolution. The scalability of collective entity resolution techniques range from $O(nlog_2 n)$ for the relational clustering technique of Bhattacharya and Getoor [12], through to worse than NP-Hard for Markov logic networks [150]. Some temporal entity resolution techniques are only marginally slower than traditional entity resolution [28, 106], however those that enforce constraints or consider entity evolution [21] are typically much slower. As a result, techniques to improve scalability (blocking, indexing, hashing, etc.) that work for traditional entity resolution approaches may be inadequate when applying advanced entity resolution techniques to large data sets.

- **Lack of Training Data**: Many real-world entity resolution problems do not have training data, i.e. a statistically representative sample of pairs (or groups) of records where the true match status is known. In the absence of such training data, collective entity resolution techniques use some kind of bootstrapping process in order to generate the starting relationships. The approach taken is usually to match the 'easy' cases using heuristic measures or attribute similarity and then try and infer the more difficult cases. However, this does not work well in practice for many domains and can lead to the problem where the entity resolution process either has too little information to make inference or has too many

incorrect starting matches to achieve good results.

This is also a problem for group linkage. Many group linkage techniques are supervised [52, 53], meaning that they require an initial set of training data in order to work correctly. Similarly, temporal entity resolution techniques require training data to calculate parameters such as agreement decay and disagreement decay [28, 106]. Again, a lack of training data in these instances is a significant practical problem.

Since the entity resolution problem is a binary classification problem, another option is to use active learning [159] to generate training data. However, the unbalanced nature of the entity resolution problem (i.e. there are likely far more true non-matches than true matches) means that care is necessary to get the right training data. While there are various active learning techniques (density based [193], uncertainty sampling [192], etc.), they all need to address the problem of class imbalance in order to be effectively used for entity resolution.

- **Parameter Tuning and Evaluation**: Traditional entity resolution has a number of parameters to tune and decisions to make, including choice of attributes, similarity functions, weightings, similarity thresholds, blocking approach, blocking attributes, etc. Advanced entity resolution techniques typically have the same parameters, and often many additional ones. For example, the collective entity resolution approach of Bhattacharya and Getoor [12] requires a relational similarity measure, maximum path length to determine neighbours and a weighting parameter for the split between attribute and relational similarity. Temporal entity resolution techniques such as that of Li et al. [106] require agreement and disagreement decay parameters to be set for each attribute in addition to the similarity functions and weights of traditional entity resolution. Population reconstruction techniques [41] may require the similarity functions and weights to be set many times for an individual problem to account for the different information in the different data sources. This means it is usually necessary to run the entity resolution process many times and evaluate the results in order to fine-tune the approach for good performance. Given the poor scalability of most advanced entity resolution techniques, and the lack of training or evaluation data (see above), this can be an extremely time-consuming process, particularly if a manual evaluation is required for each iteration.

Each of these limitations is much more likely to be present for real-world entity resolution problems than in an academic environment. In an academic environment, problems with eval-

uation, model tuning, obtaining training data and scalability can potentially be overcome (or at least minimised) by choosing an appropriate data set or data sets, something that is unlikely to be possible in real-world applications.

1.3 Research Questions

Based on the three limitations we identified, we propose the following three research questions as the basis for the work in this thesis.

- How can we find ways to allow advanced entity resolution techniques to be applied to large data sets? Rather than try and optimise the performance of existing advanced entity resolution techniques, we instead look at ways of blocking the data sets to reduce the impact of the high computational complexity of advanced entity resolution techniques.
- How can we improve the training data generation and bootstrapping process for advanced entity resolution techniques, particularly in situations where current techniques do not perform well? Collective entity resolution techniques require bootstrapping or training data to create the initial relationships, and while the current approaches to dealing with this problem work well in some situations, they are less successful in others. In addition, other types of advanced entity resolution techniques such as group linkage or temporal entity resolution require training data in order to learn models or accurately set parameters.
- How can we improve the parameter tuning and evaluation process for entity resolution? Entity resolution techniques have a number of parameters to tune including similarity functions and thresholds, blocking techniques and attributes, classifier parameters and so on. Advanced entity resolution techniques have all these parameters and many additional ones. It is often neccessary to iterate extensively in order to tune these parameters, and given the poor scalability of many advanced entity resolution techniques and the lack of evaluation data, this can be very time consuming and costly.

1.4 Research Methodology

The research methodology used in this thesis largely mirrors the traditional research approach [101], with minor modifications to reflect the current practices in the computer science domain. The main steps in the methodology are shown in Figure 1.2 and briefly described here, noting

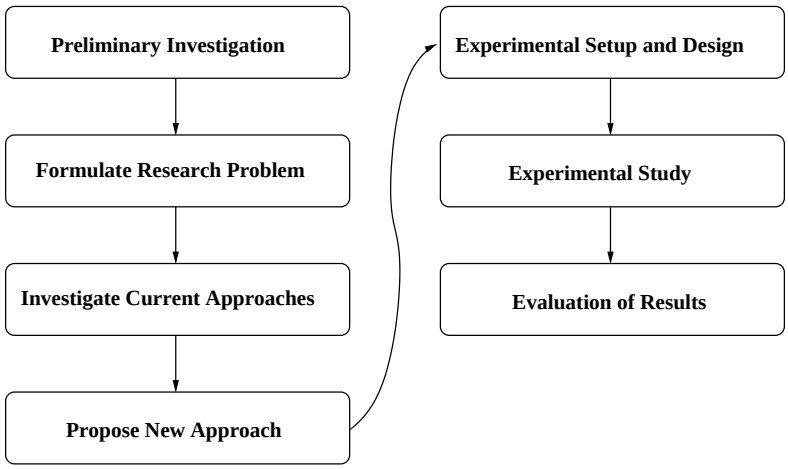

Figure 1.2: Thesis research methodology

that although they are presented sequentially, there is often iteration and repetition necessary as possible approaches are tried, tested, discarded, refined, and so on:

1. **Preliminary investigation**. In our preliminary investigation, we developed a general overview of entity resolution including current challenges, open questions and areas of investigation, along with the conventions, standards, general approach and methodology used, and so forth.

2. **Formulate research problem**. In the next stage, we focused on particular problems and short-comings in current practice. Based on what we discovered in the preliminary investigation, we identified a gap between approaches used in current academic practice and approaches that were in use in the real-world, particularly with respect to more advanced entity resolution techniques. We also observed several factors that contributed to this difference in practice. Based on these factors, we formulated the research problems detailed in Section 1.3.

3. **Investigate current approaches (literature review)**. After deciding to investigate practical limitations of advanced entity resolution techniques, we researched current approaches to dealing with these limitations, how successful they were and any other approaches that have been proposed but shown to be ineffective. This step is the focus of Chapter 3.

4. **Propose new approach**. For each of the specific limitations we observed, we developed a new approach to solve the problem. To verify the soundness of our approach, we devel-

oped a prototype for proof of concept and small scale testing prior to the full experimental evaluation. We also conducted a theoretical analyisis examining such aspects as computational complexity, mathematical formulation and guarantees, optimality and so forth. This was done separately for each approach and the details are presented in Chapters 4, 5 and 6.

5. **Experimental setup and design**. After verifying the soundness of our proposed approach, we designed our experimental setup. This involved developing software to implement our approach and algorithm, as well as any base-line comparisons that were appropriate. We also selected evaluation metrics based on conventions in the entity resolution discipline and appropriateness for our particular problems. As with the previous step, this was done separately for each of Chapters 4, 5 and 6.

6. **Experimental study**. After designing our experimental setup, we ran a full experimental evaluation. Each contribution in this thesis has been tested on multiple data sets from different problem domains and with different parameter settingss. This is to ensure that the results are not only applicable to a single data set or problem domain. Where relevant, we also compared our results against appropriate base-line techniques in order to demonstrate improvements over the current approaches used. Again, these are discussed separately in the corresponding chapters.

7. **Evaluation and discussion of results**. Finally, for each of our approaches, we analysed and evaluated our results to determine whether we had addressed our original research questions. Where we were only partially successful, we analysed the circumstances under which our approaches were effective, whether these circumstances could be identified in advance, and whether there are additional improvements that could be made to our approaches in order to address the shortcomings.

1.5 Research Contributions

A diagram of the traditional entity resolution process is shown in Figure 1.3 and is described in more detail in Section 2.3. This thesis has three main contributions which span the entire entity resolution process.

Our first contribution is applied in the blocking and indexing step and is described in Chapter 4. We propose a blocking technique that produces blocks within a size range. This limits the

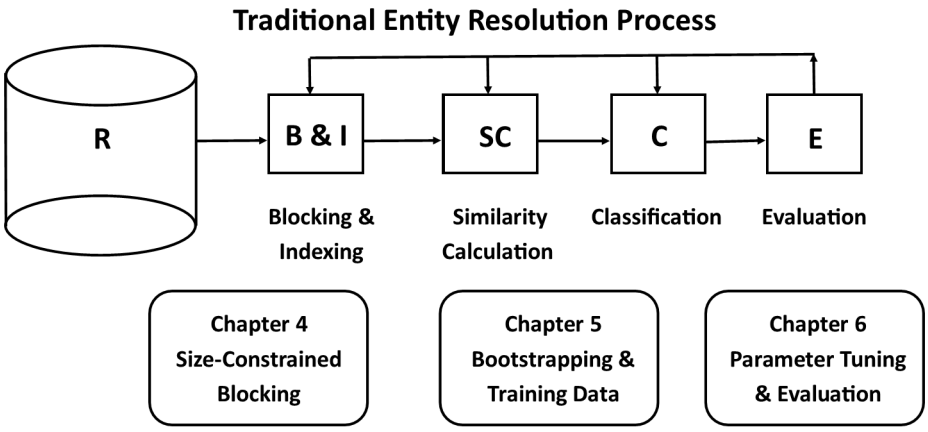

Figure 1.3: The entity resolution process and thesis contributions

effects of the poor scalibility of many advanced entity resolution techniques, since the processing time for blocks of a fixed size is constant. By also incorporating a minimum block size, we attempt to maximise the recall of our blocking approach by selecting the most likely candidates for matches, subject to the efficiency requirements. This work was published in Fisher et al. [49].

Our second contribution is applied between the similarity calculation and classification steps, and is detailed in Chapter 5. We propose a technique for generating training and bootstrapping data for advanced entity resolution techniques, most of which are supervised in some fashion. We show how we can exploit matching constraints that are present in many entity resolution problems in order to improve the size of the training data set without sacrificing precision. An early version of the work in this chapter was published in Fisher et al. [48].

Our third main contribution is an unsupervised approach for evaluating entity resolution that makes use of transitive closure to detect inconsistencies in an entity resolution result. The approach has two applications in the entity resolution process and is described in Chapter 6. The first application is to perform an unsupervised evaluation using transitive closure to detect parameter settings that produce inconsistent results in between the similarity calculation and classification steps. This allows poor parameter choices to be discarded without proceeding to the (often expensive) classification and evaluation steps of the entity resolution process. Our approach can also be applied during the evaluation step to detect groups of records that require further investigation or assessment, or which may be indicative of wider problems in the entity resolution process. The work in this chapter was published in Fisher and Wang [50].

In addition to our three main contributions, we also present an extension of our training data generation approach as a case study in Chapter 7. We show how the concept of ambiguity can be extended to situations where relationships exist between records. This allows us to both validate the matches in the training data, and also increase the size of the training set by considering groups of records which are collectively unambiguous. We demonstrate the effectiveness of this approach on historical registry data from the Isle of Skye in Scotland, and show that despite the difficulty of the problem domain and the inadequacy of other techniques, we are still able to produce a viable training data set for the problem.

1.6 Thesis Outline

The remainder of this thesis is structured as follows:

- In Chapter 2 we present background information on entity resolution. We discuss the entity resolution process as well as common application domains. We also address some practical matters including common notation, data sets and the experimental setup used in this thesis.
- In Chapter 3 we provide a summary of related research. We include a brief summary of historical developments in entity resolution. This is followed by a discussion of advanced entity resolution techniques including collective entity resolution techniques, group entity resolution techniques, temporal entity resolution techniques and population reconstruction techniques. The remainder of the chapter is split into three sections, each corresponding to one of our major contributions. The first section deals with blocking techniques, the second section deals with training data generation and active learning, and the third section deals with evaluating entity resolution results.
- In Chapter 4 we consider the scalability problem for advanced entity resolution techniques. Rather than trying to improve scalability by optimising the performance of particular entity resolution techniques, we note that generally their scalability is poor but that in practice they run in a reasonable timeframe if the data sets are small enough. As a result, we present a blocking framework which allows us to create disjoint blocks with fixed size constraints (subject to some reasonable conditions). This allows entity resolution techniques to be run sequentially (or in parallel) on small data sets, thereby limiting the performance impact of the poor scalability.

- In Chapter 5 we address the limitation of not having training data to build and evaluate models. Collective entity resolution techniques rely on bootstrapping or training data in order to work effectively. Other entity resolution techniques are supervised and rely on labeled training examples in order to learn the appropriate classification model. While current approaches work in some situations, they have significant limitations in certain problem domains and on certain types of data. We present a technique that makes use of matching restrictions and ambiguity to generate training and bootstrapping data in situations where current approaches are less successful. We also present an alternative approach to active learning [159] which can be used if resources for limited manual classifications are available.

- In Chapter 6 we examine the dual problems of parameter tuning and evaluation of entity resolution techniques. We present a technique for quickly eliminating unsuitable parameter settings in an unsupervised fashion by examining the consistency of the output from the blocking and similarity calculation steps (as described in Section 2.3). This addresses the problem of parameter tuning by detecting parameter settings that will result in inconsistent results prior to running a time-consuming entity resolution technique. By doing this in an unsupervised fashion, it also reduces the need for evaluation in situations where training and testing data does not exist.

- In Chapter 7 we present an extension of the training data generation technique from Chapter 5 for situations where relationships exist between entities. We expand the concept of ambiguity to groups of related records, and use this to both validate our training data and improve the size of the training set. We evaluate the approach as a case study using population reconstruction data and show that despite the difficulty of the problem domain, we can generate a viable set of training data.

- Finally, in Chapter 8 we present our conclusions, summarise our research contributions and present some possible directions for extending our work in the future.

Chapter 2

Background and Definitions

Entity resolution is the task of determining which records in one or more data sets refer to the same real-world entities. Entity resolution has been an active area of research since at least the 1960's [47] and many different techniques have been developed for each stage of the entity resolution process. In this chapter we define the entity resolution problem, first informally through an example in Section 2.1, then formally in Section 2.2, along with the terms and notation we use throughout this thesis. In Section 2.3 we provide an overview of the entity resolution process. In Section 2.4 we describe the practical uses of entity resolution and various problem domains where it is applied. We conclude the chapter by providing the details of our data sets and experimental setup in Section 2.5, before presenting an overall summary in Section 2.6.

2.1 Entity Resolution - An Example

In an ideal world all data would be standardised and correct and require no preparation before being analysed or used. In practice this is normally not the case. The phrase 'real-world data is dirty' [73] is very common in data mining. One example of 'dirty' data is having multiple records for the same entity (which could be a person, product, research paper, etc.) in the same data set. Entity resolution (also known as *data matching*, *record linkage*, *merge/purge*, *duplicate detection*, etc., [23, 47, 72, 119]) is the task of detecting such duplicate records. It also encompasses a similar problem where two (or more) data sets need to be joined or merged, and for each record in data set A the corresponding record (or records) in data set B needs to be found. We illustrate the problem and some common approaches to solving it here through a short example, and formally describe the problem in Section 2.2 below.

Record ID	Name	Address	Employer	Spouse
r_1	J. Smith	Miller St. Canberra	Big W	r_{42}
r_2	John Smyth	42 Miller St. O'Connor	Woolworths	r_{42}
r_3	Jeffrey S. Fisher	42 Mills St. Campbell		r_{72}
...	...	...	...	
r_{42}	Mary S.	42 Miller St. O'Connor	Google	r_1

Table 2.1: Example data set to describe the entity resolution problem.

Consider the small data set shown in Table 2.1. The entity resolution problem can be formulated as follows: *do records r_1, r_2 and r_3 refer to one, two or three real-world people?* The data set in Table 2.1 also gives some indication why this might not be straightforward. Does the 'J.' in the value of the *Name* attribute for 'r_1' stand for 'John', 'Jeffrey' or something else? Is 'Smyth' a mispelling of 'Smith' or a different name? Are the values in *Address* and *Employer* up-to-date? Is r_3 unemployed or do we just not know his employer? In order to deal with difficulties such as these, many techniques have been developed and we describe three broad approaches here.

The first approach to be developed, and one still widely used in practice, is to consider the similarity of attribute values. In this example we see that the records have similar values in the *Name* attribute as well as the *Address* attribute (we look at exactly what similarity means below). An idea that began with early entity resolution techniques [47] is that similar attribute values are evidence that records may refer to the same entity.

However, attribute similarity may not be sufficient for all entity resolution problems so other approaches also make use of relationship information. In our example we can see that both r_1 and r_2 have the same spouse r_{42}. Being in the same relationship with another entity (in this case being married to) is also evidence that two records might refer to the same real-world entity. Collective entity resolution techniques [12, 39, 86, 150] typically make use of relationship information in the entity resolution process.

Finally, other approaches make use of domain knowledge (usually in addition to attribute and relationship similarity). Domain knowledge, in this case knowledge of Canberra's geography, informs us that 'O'connor' is a suburb of 'Canberra.' Additionally, knowledge of Australian corporations tells us that 'Big W' is a subsidiary of 'Woolworths.' As a result, even though these attribute values appear to have low similarity, they may in fact be the same address and employer.

In practice, a human expert can often interpret the context, make use of domain knowledge and other learned experience such as 'Smyth' being a possible variation or mispelling of 'Smith'

to perhaps come to the conclusion that for our toy example r_1 and r_2 refer to the same person, while r_3 refers to someone else. However, even human experts aren't always guaranteed to get it right. For example, simple experiments conducted by Smalheiser et al. [166], found that two experts did not always agree on the results of entity resolution, and that in over a third of cases it was not possible to be certain about the result. There were even a small number of cases, where one expert determined that the two records definitely referred to the same entity while the other expert determined that the two records definitely referred to different entities [166]. Similar results have been found in research on active learning with imperfect oracles [32, 65], and in population reconstruction work [148].

Capturing the same (or a superior) decision making process in an algorithm that runs on a computer is even harder, and defining each of the approaches described above in such a way that a computer can make an accurate assessment is a non-trivial task.

What does it mean for two attribute values to be similar and how can this be measured? Intuitively people might say that the name 'Smith' is more similar to 'Smyth' than it is to 'Fisher' but how can this concept be quantified. The general approach is to use a distance measure to assign a numerical score as a way of quantifying the similarity between two attribute values. Many distance measures have been proposed for this purpose including Levenshtein edit distance [105] or Jaccard similarity [119] for text attributes, and Jaro-Winkler [188] for proper nouns such as names. Other distance measures can be used for different data types in order to preserve the cyclical nature of date and time attributes (i.e. 'December' should be close to 'January') or to capture physical distance in the case of geographic attributes such as latitude and longitude.

In the toy example presented in Table 2.1 the relationship information was captured as the unique identifier of another record. However in practice, relationship information can also be ambiguous, missing, incorrect, or captured through a pseudo-identifier such as a name. Just as attribute similarity can be measured, so can relationship similarity. Measurements using common neighbours, random walks and weighted rules are all used to capture how similar the relationships between a pair of records are [1, 86, 165].

Finally, domain knowledge can be incorporated in many ways. Rules and mappings can relate nicknames to full names, ontologies can describe geographical hierarchies, such as 'O'Connor' is a suburb of 'Canberra,' and look-up tables can encode abbreviations and acronyms for places, companies or organisations. Such rules and ontologies can be created manually, or learned from training data when it exists [139, 152]. However, these rules and knowledge bases are typically very domain specific and cannot easily be translated from one problem domain to another.

The inherent difficulty of the entity resolution problem and the variety of ways to try and solve it have made entity resolution an active area of research. In the next section, we formalise the notion of entity resolution and provide a definition of terms and concepts that we will use throughout this thesis.

2.2 Definitions and Notation

We now provide a definition of entity resolution and relevant related terms that will be used frequently throughout this thesis. For the most part we make use of standard database terminology [44].

Attribute: An **attribute** is a property of, or fact about, an **entity type**. Examples include a *FirstName* attribute which corresponds to a person's first given name, or a *Title* attribute refering to the title of a research paper.

In some cases the distinction between an attribute and an entity type may depend on the data set(s) and problem in question. An address is often treated as an attribute of a person or organisation, but if the task is to perform **entity resolution** on geographic data, then address may be treated as an entity type and resolved as part of the entity resolution process.

There are also situations where **relationships** are modeled as attributes For example, in the data set shown in Table 2.1 the spousal relationship is captured as an attribute.

An **attribute value** is the value of the attribute for a particular **record** of the given entity type. For example, a particular person record might have an attribute value of 'John' in the *FirstName* attribute. The attribute values for a particular attribute are drawn from a set of valid values called the **domain** of the attribute. For example, 'John' and 'Mary' are in the domain of a *FirstName* attribute but '75' is not. Similarly, '2603' and '2902' are in the domain of an Australian *Postcode* attribute, but 'Canberra' is not.

Record: A **record** is a collection of **attribute values** about one or more real-world **entities**. For example a record might contain the *Name*, *Age* and *Address* of a person, or the *Title*, *KeyWords*, *Authors* and *Journal* of a paper. We denote the reference from a record to an entity with the $\mapsto$ symbol, e.g. $r_i \mapsto e_y$.

In some cases an attribute value is a refererence to another record. For example the *Authors* and *Journal* of a paper, can be foreign keys in a relational database, sub-nodes in an XML tree, etc. This normally implies a **relationship** between the entities to which each record refers.

Entity: An **entity** is 'something that has separate and distinct existence and objective or conceptual reality.'[1] Examples of entities include a particular person, business, academic paper, or product. An **entity type** is the type or class of entities to which a specific entity belongs. For example, individual people are entities of the person entity type.

In many cases it is obvious what constitutes an individual entity, and as a result what it means for two entities to actually be the same. For example, if a person's name changes, he or she is still the same person. Similarly, if the price and category of a product change, it is still the same product.

However, there are examples where it is less obvious what constitutes an entity and whether two entities should be treated as the same. Examples include academic institutions or commercial organisations. The Australian National University Medical School has a facility at the Canberra Hospital. When performing **entity resolution** of institutions, it is not clear whether this facility should be treated as a separate entity, as part of the Australian National University, part of the Canberra Hospital, or part of both parent institutions. Similarly, if two companies merge, should the new company be treated as the same entity as one or both of the original companies, or as a new entity entirely.

As a result, we distinguish between **atomic entities** and **compound entities**. An **atomic entity** is not divisible, i.e. a person, research paper, etc., while a **compound entity** is a collection of other entities of different entity types (which can be either simple or compound). For example, a household is a collection of people, an organisation is a collection of groups or divisions, etc.

In the case of compound entities they can change, and can split and merge as well as begin and end. For example a household in a census could split into two households as a result of a separation or children moving out of home. Similarly, two companies could merge to form a single new company. We must be careful when performing entity resolution between such compound entities to clearly specify exactly what constitutes an entity, and what our definitions are for a **match** and **non-match**.

Finally, we define the notion of equivalence between entities. We say that $e_x \equiv e_y$ if they are the same real-world entity and $e_x \not\equiv e_y$ if they are different real-world entities. In the case of atomic entities this is usually an easy assessment to make. Are e_x and e_y two different people or both the same person?, are they two different papers or the same one? In the case of compound entities, we may have to tailor the definition of equivalence to the specific problem we are trying to solve.

[1] https://www.merriam-webster.com/dictionary/entity

Relationship: A **relationship** is a connection or association between two or more **entities**. Examples include traditional familial relationships such as e_x is the father of e_y, but also includes other associations, such as e_x and e_y are coauthors of a publication, or live at the same address. As with entities, relationships also have a **relationship type** which characterises the nature of the relationship. Examples include coauthor, parent of, spouse of, and more.

Some relationships are symmetric, for example if e_x is a coauthor of e_y, then e_y is also a coauthor of e_x. In other cases, a relationship may be asymmetric, for example if e_x is the father of e_y, e_y is not the father of e_x. However, in such cases there is normally an inverse relationship which captures the same or similar information, for example if e_x is the father of e_y, then e_y is the child of e_x.

Just as with entities, relationships can also have attributes that are properties of, or facts about, the relationship. Particularly common are temporal attributes such as start and end points. For example, a marriage relationship begins when the couple get married and ends when they get divorced or one partner dies and these dates and times might be recorded as attributes of the marriage relationship. Other examples of attributes might indicate the strength or frequency of a relationship, for example the number of publications two people have coauthored, or the amount of money involved in a contractual business relationship.

Entity resolution: The **entity resolution** problem is formulated as follows: given two data sets $\mathbf{R}_1$ and $\mathbf{R}_2$ (with possibly $\mathbf{R}_1 \equiv \mathbf{R}_2$), for each pair of records $\langle r_i, r_j \rangle : r_i \in \mathbf{R}_1, r_j \in \mathbf{R}_2$, determine whether $\langle r_i, r_j \rangle$ is a **true match** or a **true non-match**. As such, entity resolution is a binary classification problem [23].

Match and **non-match**: In situations where we have **atomic** (i.e. not **compound**) **entities**, then then given a record pair $\langle r_i, r_j \rangle$ such that $r_i \mapsto e_x$ and $r_j \mapsto e_y$ we define $\langle r_i, r_j \rangle$ to be a **true match** if $e_x \equiv e_y$ and $\langle r_i, r_j \rangle$ to be a **true non-match** if $e_x \not\equiv e_y$.

The goal of **entity resolution** is to classify record pairs as **true matches** or **true non-matches**. However, most entity resolution models are imperfect, i.e. the classification result may not reflect the actual real-world situation. As such, we distinguish between **true matches** and **classified matches** and **true non-matches** and **classified non-matches** where **true matches** and **true non-matches** are the state of the real-world **entities**, i.e. $e_x \equiv e_y$ or $e_x \not\equiv e_y$, and **classified matches** and **classified non-matches** are the output of the entity resolution process. We discuss this distinction further in Section 2.3.4 when we look at ways of evaluating entity resolution.

As discussed above, when dealing with **compound entities**, it can be necessary to alter the definition of **match** and **non-match** (for both true and classified matches and non-matches) to suit the particular situation or problem. For example, we might say two households match if 75% of their members (i.e. individuals, atomic entities) match using the above definition of match for atomic entities.

We note that some entity resolution approaches (particularly historical ones) had a third category of **potential match** (or similar name) [66], which indicated the need for further analysis, often a manual clerical review [47]. However, this was an intermediate step in the process and the final outcome in these approaches was still a separation into the two classes, match and non-match.

Constraints: It is quite common for **entity resolution** to take place in the presence of **constraints** (i.e. not every pair of records can be a **match**, or some pairs of records must be a **match**). While some of these **constraints** are domain and problem specific, we define three common types of constraints.

Given two data sets $\mathbf{R}_1$ and $\mathbf{R}_2$, noting that possibly $\mathbf{R}_1 \equiv \mathbf{R}_2$, we define the following **constraints**:

- **One-to-Many** (1-to-M): A one-to-many **constraint** implies that given $r_i, r_j \in \mathbf{R}_1$ and $r_k \in \mathbf{R}_2$, then if $\langle r_i, r_k \rangle$ is a true match and $\langle r_j, r_k \rangle$ is a true match then we must have $i = j$. A real-world example where this constraint might occur is when performing **entity resolution** between a data set of birth certificates and a data set of marriage certificates, since an individual can (potentially) be married multiple times but is only born once. The many-to-one (M-to-1) case is symmetric.

- **One-to-One** (1-to-1): A one-to-one **constraint** implies that given $r_i, r_j \in \mathbf{R}_1$ and $r_k, r_l \in \mathbf{R}_2$, then if $\langle r_i, r_k \rangle$ is a true match and $\langle r_j, r_k \rangle$ is a true match then we must have $i = j$. We also have that if $\langle r_i, r_k \rangle$ is a true match and $\langle r_i, r_l \rangle$ is a true match then we must have $k = l$. A real-world example of this situation might be performing **entity resolution** between birth certificates and death certificates, where each individual person can only be born once, and can only die once.

- **Transitive Closure**: **Transitive closure** in **entity resolution** reflects the fact that **true match** is normally transitive. Formally, given three records $r_i, r_j, r_k \in \mathbf{R}_1 \cup \mathbf{R}_2$, then if $\langle r_i, r_j \rangle$ is a true match and $\langle r_j, r_k \rangle$ is a true match then we must have $\langle r_i, r_k \rangle$ is also a true match. The same does not necessarily hold for true non-matches.

Common Notation	
$\mathbf{R}$	A set of records. Individual records are denoted $r_i \in \mathbf{R}$.
$\mathbf{A}$	The set of attributes associated with each record $r_i \in \mathbf{R}$.
$r_i.a_j$	The value of attribute a_j for record r_i, where $r_i \in \mathbf{R}$ and $a_j \in \mathbf{A}$.
Italics	The name of an attribute or relationship type, e.g. *FirstName*, *MarriedTo*.
'Quoted'	The literal value of an attribute, e.g. 'John', 'S503'.

Chapter 4: Blocking	
$\langle a_i, f_j \rangle$	A blocking key, consisting of an attribute $a_i \in \mathbf{A}$ and function f_j.
$v_{i,j,y}$	A blocking key value, obtained by applying function f_j to the value of attribute a_i for record r_y. I.e. $v_{i,j,y} = f_j(r_y.a_i)$.
$\mathbf{B}$	A set of blocks. Individual block are denoted $\mathbf{b}_i \in \mathbf{B}$.
$V(\mathbf{b}_i)$	The set of blocking key values associated with block $\mathbf{b}_i$.
$\lvert\mathbf{b}_i\rvert$	The size of (number of records in) block $\mathbf{b}_i$.
$\zeta(v_1, v_2)$	The similarity of blocking key values v_1 and v_2.
$\varsigma(\mathbf{b}_1, \mathbf{b}_2)$	The overall similarity between blocks $\mathbf{b}_1$ and $\mathbf{b}_2$.

Chapter 5: Training Data Generation	
$\Psi(r_i, r_j)$	A similarity function that calculates a similarity measure for record pair $\langle r_i, r_j \rangle$.
ψ_{ij}	The similarity value of record pair $\langle r_j, r_j \rangle$ for $r_i, r_j \in \mathbf{R}$.
ψ_{min}	The minimum similarity threshold for matches in the bootstrapping.
α_i	The ambiguity value of record r_i.
α_{max}	The maximum ambiguity threshold for matches in the bootstrapping.
$\mathbf{G}$	Graph where each node n_i corresponds to a record $r_i \in \mathbf{R}$ and each edge $\langle n_i, n_j \rangle$ has weight ψ_{ij}.
$M_p(n_i)$	The p^{th} highest edge weight among the adjacent edges of node $n_i \in \mathbf{G}$.

Chapter 6: Parameter Tuning and Evaluation	
ψ_{ij}	The similarity value of record pair $\langle r_j, r_j \rangle$ for $r_i, r_j \in \mathbf{R}$.
ψ_{min}	Minimum similarity threshold for likely matches.
$\mathbf{G}$	Graph where each node n_i corresponds to a record $r_i \in \mathbf{R}$ and each edge $\langle n_i, n_j \rangle$ has weight ψ_{ij}.
$\mathbf{G}^+$	A subgraph of $\mathbf{G}$ such that $\{\langle n_i, n_j \rangle \in \mathbf{G} : \psi_{ij} \geq \psi_{min}\}$.
Δ_{co}	A consistent triangle.
Δ_{ip}	An incomplete triangle.
Δ_{ic}	An inconsistent triangle.

Table 2.2: Table of notation used in this thesis.

2.3 The Entity Resolution Process

The entity resolution process typically consists of a number of steps, as illustrated in Figure 2.1, and we provide a brief overview of the major ones here. In addition to the steps described, different techniques and situations may require additional stages, such as data standardisation [23], a clerical review of possible matches [47], active learning to generate training data [159],

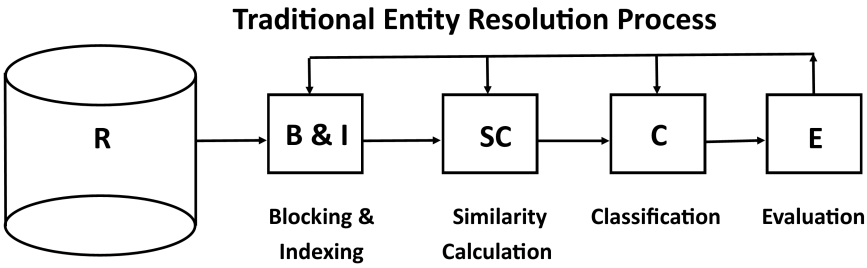

Figure 2.1: The traditional entity resolution process.

etc. It is also worth noting that while the ideal process is sequential (as shown in Figure 2.1), it is often neccessary to iterate and tune various parameters and models throughout the process which can require repetition and refinement of each step.

2.3.1 Blocking and Indexing

For most entity resolution tasks, the majority of record pairs will be true non-matches. Consider two data sets containing m records and n records that are individually clean (i.e. there are no duplicates in either data set). The maximum number of true matches is the lesser of m and n, while the number of potential matches is $m \times n$. A similar situation occurs when deduplicating a single data set with n records. The number of unique record pairs is $\frac{n \times (n-1)}{2}$. However, for true matches to outnumber true non-matches, over half the records in the data set need to refer to the same entity, which is not the case in most real-world entity resolution problems.

Because the entity resolution problem is inherently quadratic, i.e. $O(m \times n)$ or $O(n^2)$ in the deduplication case, but as described above, the majority of comparisons are (almost always) true non-matches, i.e. $O(m \times n - \min(m, n))$ or $O(n)$ in the deduplication case, the basic idea of blocking and indexing is to quickly eliminate record pairs which have very little or no possibility of being true matches, and limit the detailed comparisons to so-called *candidate pairs*, i.e. record pairs that might be true matches [23].

While we refer to this step as *Blocking and Indexing*, since these are the two approaches we use throughout this thesis, there are many other techniques that have been shown to be effective at selecting the candidate pairs for detailed comparison including locality sensitive hashing [107], canopy clustering [112], suffix arrays [2] and more. We summarise the two broad categories of

techniques that are used in this thesis, *blocking* and *indexing*, and give a further review of relevant techniques in Chapter 3.

Blocking: The idea of blocking is to subdivide the records into different **blocks**, and only compare pairs of records in each block. For example, traditional blocking [47] involves selecting one or more attributes and separating the records into blocks based on the values of these attributes. One block is created for each unique combination of values in the selected attributes and all records with the same combination of values are placed in the same block. For example, blocking a data set using a *FirstName* attribute, would generate one block for each unique value of *FirstName*, and all records with the same value of *FirstName* would be placed in the same block, i.e. all the records with *FirstName* 'John' in one block, 'Mary' in another block, and so on. Blocking a dataset on the combination of *FirstName* and *Surname* would create one block for each unique combination of *FirstName* and *Surname*, so the records with *FirstName-Surname* 'John-Smith' would be in one block, 'Mary-Miller' in another block, 'Mary-Smith' in a third block, and so forth. Once the blocks have been created, each pair of records in the same block becomes a candidate pair.

Beyond the traditional approach of blocking based on attribute values, several other techniques have been proposed. For example phonetic encodings such as Soundex [124], Phonex [104] and Double Metaphone [135] attempt to group similar sounding words (usually proper nouns like given names or street names) into blocks to account for potential mispellings and name variations.

Indexing: The idea behind indexing is to somehow order the records into an index. Then, two records become a candidate pair if and only if they are within a certain distance (using some distance metric) in this index. For example, sorted neighbourhood [73] works by using one or more attribute values as a sorting key, such as a concatenation of *FirstName* and *LastName*. This 'sorting key' is then used to order the records and all pairs of records within a fixed window size in this index become candidate pairs.

2.3.2 Similarity Calculation

Once blocking, indexing or another similar technique has been used to produce candidate pairs, the next step in the entity resolution process is to compare the records. This involves calculating a similarity score for each candidate record pair produced by the blocking step. There are many different techniques for this stage of the process, however the basic idea of them is to calculate a numerical measure of similarity between the two records.

The first step in this process is usually to calculate the similarity between the individual attribute values in the records. A number of different similarity functions have been developed for this process for use on different types of data. Each attribute similarity function takes as input two attribute values and returns a numerical value between 0 and 1. The higher the numerical score, the more similar the two attribute values. We describe four common attribute similarity functions here which we make use of throughout the thesis.

Edit Distance: Edit Distance [105] - sometimes called Levenstein edit distance - uses the idea that the distance between two text strings can be measured by counting the number of insertions, deletions and substitutions that are required to transform one string into the other. It is highly customisable and allows different costs to be associated with the different operations, or entirely new operations such as transpositions to be added in order to tailor it to a specific problem. One disadvantage of edit distance is that it is relatively slow, being quadratic, i.e. $O(|s_1||s_2|)$, where $|s_1|$ and $|s_2|$ are the lengths of the strings being compared.

Exact Match: Exact Match is a binary similarity function that tests whether two strings are identical or not. It is useful in domains where there are only a small number of values (for example a *Gender* attribute). It is calculated as follows:

$$S_{Ext}(s_1, s_2) = \begin{cases} 1.0 & s_1 = s_2 \\ 0.0 & s_1 \neq s_2 \end{cases} \tag{2.1}$$

Jaccard Similarity: Jaccard similarity (also known as q-gram similarity) makes use of q-grams, which are substrings of length q. The Jaccard similarity is computed as:

$$S_{Jac}(s_1, s_2) = \frac{|\mathbf{Q}_1 \cap \mathbf{Q}_2|}{|\mathbf{Q}_1 \cup \mathbf{Q}_2|} \tag{2.2}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are the sets of q-grams contained in strings s_1 and s_2 respectively [119]. It is fast ($O(|s|)$ in the length of the longer string if the set membership test can be done in constant time) and simple to implement, and works particularly well in longer multi-word strings of text where there is a possibility that the words might be in different orders. For example, the Jaccard similarity for s_1 = 'Jeffrey Fisher' and s_2 = 'Fisher Jeffrey' with $q = 2$ is computed as follows: $\mathbf{Q}_1$ = {'Je', 'ef', 'ff', 'fr', 're', 'ey', 'y ', ' F', 'Fi', 'is', 'sh', 'he', 'er'} and $\mathbf{Q}_2$ = {'Fi', 'is', 'sh', 'he', 'er', 'r ', ' J', 'Je', 'ef', 'ff', 'fr', 're', 'ey'}. Thus $|\mathbf{Q}_1 \cap \mathbf{Q}_2| = 11$, and $|\mathbf{Q}_1 \cup \mathbf{Q}_2| = 15$. Thus, $S_{Jac}(s_1, s_2) = 11/15$ or 0.73.

Jaro and Winkler: The Jaro string comparison technique [83], and the Winkler modifications [188], were specifically designed to account for errors that are commonly found in proper nouns. In particular, they are used when comparing the names of people. The Jaro similarity of two strings is calculated as:

$$S_{Jar}(s_1, s_2) \;=\; \frac{1}{3}\left(\frac{c}{|s_1|} + \frac{c}{|s_2|} + \frac{c-t}{c}\right) \tag{2.3}$$

where c is the number of agreeing characters between the two strings within half the length of the longer string, and t is the number of transpositions in the two strings.

The Winkler modifications [188] improves the Jaro similarity measure by giving increased weight to agreeing characters at the beginnings of the strings, long strings with multiple characters in common, and strings differing by commonly substituted characters.

In addition to the above measures that are designed to work on strings of text, additional comparison measures have been proposed for geographical attributes such as latitude and longitude, date and time attributes (which should be cyclical), age attributes and other data types. See Christen [23] or Naumann and Herschel [119] for other examples.

Just as the similarity of attribute values can be measured, so too can the similarity of relationships be measured. In some cases relationship similarity may be binary, for example two records either have the same spouse record or they don't, but in other cases the relationship information may be more complex and require measuring techniques that reflect this complexity, e.g. two author records may have only partial overlap in co-authors, and the relationship similarity should reflect the extent of the overlap. Relationship similarity is calculated for two records by looking at other records they have a relationship to, such as the coauthor records on a research paper or the members of a household in a census record. Like attribute similarity, relationship similarity is normalised to a value between 0 and 1 with higher values indicating higher similarity of relationships.

Common Neighbours: The common neighbours method of calculating relational similarity is analagous to the Jaccard similarity (described above) for attributes. It looks at the neighbours (i.e. related records) of a pair of records and is calculated as:

$$S_{CN}(r_1, r_2) \;=\; \frac{|\mathbf{N}_1 \cap \mathbf{N}_2|}{|\mathbf{N}_1 \cup \mathbf{N}_2|} \tag{2.4}$$

where $\mathbf{N}_1$ and $\mathbf{N}_2$ are the sets of related records of record r_1 and r_2 respectively. [12]. For exmaple, if r_1 is has a relationship with records r_3, r_4, and r_5 and r_2 has a relationship with

records r_4, r_5, r_6, and r_7, then $\mathbf{N}_1 \cap \mathbf{N}_2 = \{r_4, r_5\}$ and $\mathbf{N}_1 \cup \mathbf{N}_2 = \{r_3, r_4, r_5, r_6, r_7\}$, thus $S_{CN}(r_1, r_2) = 2/5$ or 0.4.

Adamic/Adar: The Adamic/Adar approach [1] extends the common neighbours approach to account for the fact that not all relationships are equally informative. A common neighbour with a large number of relationships is not as strong evidence for a match as a common neighbour with only two relationships (i.e. the records being compared). In this way it has many similarities with the idea of TF-IDF [169] used in information retrieval where agreement on rare terms is much stronger evidence than agreement on terms common in most documents.

The Adamic/Adar relationship similarity of two records is given by:

$$S_{Ada}(r_1, r_2) = \sum_{r_i \in \mathbf{N}_1 \cap \mathbf{N}_2} \frac{1}{log(|\mathbf{N}_i|)} \tag{2.5}$$

where $\mathbf{N}_1$, $\mathbf{N}_2$ and $\mathbf{N}_i$ are the sets of related records of records r_1 and r_2 and record r_i respectively.

Typically, the comparison step involves calculating the similarity between many different attributes, and potentially also relationships, which means the final output of a record pair comparison is a *weight vector* or similarity vector, containing the similarity scores of the individual attribute comparisons. The weight vector for records r_i and r_j could take the form $\Psi_{ij} = [\psi_1, \psi_2, \ldots, \psi_{|\mathbf{A}|}, \psi_r]$ where ψ_x is the attribute similarity of records r_i and r_j in attribute a_x for each $a_x \in \mathbf{A}$, and ψ_r is the relationship similarity of records r_i and r_j. As described in the next section, such weight vectors are used in the classification step to determine whether the record pair is a match or non-match.

2.3.3 Classification

Once the record pairs have been compared, they have to be classified as either matches or non-matches. As with the other steps in the entity resolution process, there are many different techniques that can be used and we describe three broad approaches here.

Simple Pairwise Threshold: The simplest approach to classification is to use a threshold ψ_{min} based on the average similarity values in the weight vector. For example if $\psi_{min} = 0.7$, all record pairs where the average similarity value in the corresponding weight vector is greater than or equal to 0.7 are classified as a match, and record pairs where the average similarity is less than 0.7 are classified as a non-match. Formally, record pair r_i and r_j, with corresponding

weight vector Ψ_{ij}, is classified as a match if:

$$\sum_{\psi_x \in \Psi_{ij}} \frac{\psi_x}{|\Psi_{ij}|} \geq \psi_{min} \tag{2.6}$$

and a non-match if:

$$\sum_{\psi_x \in \Psi_{ij}} \frac{\psi_x}{|\Psi_{ij}|} < \psi_{min} \tag{2.7}$$

This approach can be extended to incorporate a weighted average, and the weights can be probabilistically determined if information regarding the size of the domain of each attribute and the error rate of each attribute are known (or can be estimated) [47].

Machine Learning Methods: Since entity resolution is essentially a binary classification problem, any machine learning technique that has been developed for classification problems can be applied to entity resolution. Techniques such as support vector machines [54], neural networks [158], logistic regression [53], decision trees [35] and others can all be used. However, these techniques are usually supervised techniques meaning that training data is required in order to learn the features of the weight vectors or raw record pair data. In practice training data is often unavailable for real-world entity resolution problems, and not easy (or cheap) to create [23]. We discuss the problem of the lack of training data and how to partially overcome it in Chapter 5.

Clustering Methods: This group of techniques treats entity resolution as a clustering problem rather than a binary classification problem. The general idea of clustering is to partition a data set into one or more clusters, where the intra-cluster distance is low and the inter-cluster distance is high [68]. The goal of these approaches to entity resolution is usually to have one cluster per entity and group all the records associated with that entity into the cluster. Various approaches have been used including hierarchical clustering [186], density based clustering [18] and others.

The clustering result can be converted to the equivalent classification result as follows: for any record pair $\langle r_i, r_j \rangle$, if r_i and r_j are in the same cluster they are a classified match, and if r_i and r_j are in different clusters they are a classified non-match. It is important to note that unless the classification method guarantees transitive closure (described in Section 2.2), then a reverse transformation may not be possible (i.e. we may not be able to convert a classification result into a consistent clustering result).

		Classification Result	
		Match	Non-Match
Actual State	Match	TM	FN
	Non-Match	FM	TN

Table 2.3: Classification matrix for entity resolution, showing the four combinations of classifier output and actual real-world state.

2.3.4 Evaluation

Since entity resolution is usually treated as a typical binary classification problem, all the traditional measures such as *accuracy*, *precision*, *recall*, etc. are applicable [168]. In addition, other steps of the entity resolution process have different evaluation measures, for example the effectiveness of the blocking is usually assessed by measures such as *pairs completeness*, *pairs quality* and *reduction ratio* [134]. We describe the common evaluation measures for the entire process here, and then describe measures for individual steps in subsequent chapters where they are relevant.

We made the distinction between *true match* (TM) and *classified match* (TM + FM) in Section 2.2 and it is relevant here. As shown in the classification matrix described in Table 2.3, for each individual record pair $\langle r_i, r_j \rangle$ that is classified, there are four possible outcomes:

- **True Match** (TM): The output of the classifier applied to r_i and r_j is a **match** and this also reflects the actual real-world state, i.e. $e_i \equiv e_j$.
- **False Match** (FM): The output of the classifier applied to r_i and r_j is a **match**, but this is not the case in the real-world state, i.e. $e_i \not\equiv e_j$.
- **False Non-Match** (FN): The output of the classifier applied to r_i and r_j is a **non-match**, but the real-world state is a match, i.e. $e_i \equiv e_j$.
- **True Non-Match** (TN): The output of the classifier applied to r_i and r_j is a **non-match** and this reflects the real-world state, i.e. $e_i \not\equiv e_j$.

Based on these four combinations of classifier output and real-world state, we describe the following evaluation metrics:

Accuracy: Accuracy (Ac) is calculated as [23]:

$$\text{Ac} = \frac{TM + TN}{TM + TN + FM + FN} \tag{2.8}$$

While accuracy is a very common measure for traditional classification problems, it turns out to be less useful for entity resolution. As described in Section 2.3.1, entity resolution is typically an unbalanced problem, which means that it is generally easy to achieve very high accuracy by simply classifying everything as a non-match.

Precision: Precision (Pr) is calculated as [23]:

$$\text{Pr} = \frac{TM}{TM + FM} \tag{2.9}$$

Precision measures the proportion of classified matches that are actually true matches.

Recall: Recall (Re) is calculated as [23]:

$$\text{Re} = \frac{TM}{TM + FN} \tag{2.10}$$

Recall measures the proportion of true matches that are classified as matches by the classifier.

Both precision and recall are widely used in practice, however there is a trade-off between them. For example, it is possible to achieve perfect recall by classifying every record pair as a match, however this normally leads to very poor precision (since only a small proportion of the classified matches will actually be true matches). It is also usually possible to achieve very high or even perfect precision, by only classifying a very small number of record pairs as matches where there is exceptionally high confidence. Again, however, this usually leads to low recall (since most true matches will be incorrectly classified as non-matches).

F-measure: Because of the inherent trade-off between precision and recall, another widely adopted evaluation metric is the F-measure (F-m), also known as the *F*Score [168], which is the unweighted harmonic mean of *precision* (Pr) and *recall* (Re). It is calculated as [23]:

$$\text{F-m} = \frac{2PrRe}{Pr \ + \ Re} \tag{2.11}$$

Since it is possible to trade off between precision and recall, F-measure provides a balanced metric and in order to achieve a high F-measure, it is necessary to score well in both precision and recall. However for practical problems, it is quite possible for one of precision or recall to be more important. This can be achieved by weighting them accordingly in the equation in order to achieve the desired balance.

Alternatively, F-measure can be expressed as the weighted arithmetic mean of precision and

recall [70], according to the formula:

$$\text{F-m} = pRe \ + \ (1 \ - \ p)Pr \tag{2.12}$$

where:

$$p \ = \ \frac{FN \ + \ TM}{FN \ + \ FM \ + \ 2TM} \tag{2.13}$$

This provides another way the relative importance of precision and recall can be changed to suit the particular problem.

2.4 Common Entity Resolution Problem Domains

We describe three main domains where entity resolution is applied and note that there is some overlap between them.

2.4.1 Government Agencies

Entity resolution is very important to government agencies for a variety of different tasks. One example is identifying people who pose a risk to national security before they are allowed into the country. In such cases these people may be actively trying to hide their identity, which requires government agencies to try and detect them through alternate means. In other cases, simple data quality problems may prevent a person of interest from being noticed without suitable entity resolution processess [46].

In the health domain, individual people often have medical records stored with many different providers, e.g. doctors, hospitals, specialist clinics, etc. This presents problems for medical research, which often requires gathering and connecting records from the different data sets to build up a complete picture of an individual's health and medical record. In Australia, many state governments have record linkage centres[2] and similar examples exist in other countries.[3] Software tools such as FEBRL [26] were originally created for health record linkage, even if they now see wider application. While it is not currently a widespread practice, it is easy to imagine a future where emergency departments at hospitals might attempt to perform real-time entity resolution to determine a patient's current medications, co-morbidity or other complications that could have potentially life-threatening consequences if unknown.

[2]http://www.cherel.org.au/, https://www.santdatalink.org.au/resources
[3]http://www.statcan.gc.ca/eng/health/link

Finally, entity resolution is also widely applied by governments in areas of criminal investigation, fraud detection and tax evasion.[4] Examples include linking the owners of bank accounts to tax records to determine people who are paying too little tax, checking tax records against social security records to work out people falsely claiming unemployment benefits, or identifying bank accounts attached to criminal networks to enable detection of money laundering schemes. In Australia during 2011, the government used entity resolution to detect the fact that many recipients of unemployment benefits were on the high roller registers at casinos around the country and by doing this was able to bring down a major money laundering operation [7].

2.4.2 Commercial Organisations

Entity resolution also sees widespread use in the commercial world for a variety of different purposes. A common scenario is matching customer, supplier or product databases following a corporate merger. However, it is also common for companies to buy and sell data sets (or the meta-data from data sets) for advertising, market research or other business purposes [141].

Another corporate example which relies heavily on entity resolution is for credit verification. It is common in many countries that anyone applying for a loan, credit card, phone, or other long term contractual arrangement has their credit history verified to make certain they do not have a history of unpaid bills or bankruptcy.[5] In these cases it is necessary to match the customer details provided to existing records in order to determine their history. As with government fraud and compliance, it is adversarial in nature, in that people with poor credit histories may try and avoid detection by providing nicknames, out of date information, etc., which makes entity resolution more difficult. However, the results of incorrect entity resolution can also have significant negative consequences on unrelated members of the public [115] which makes it very important that the entity resolution is correct.

Companies can also make use of entity resolution for regulatory compliance and business reasons. One example of this is described by Jeff Jonas [85] in regards to Las Vegas casinos, where a casino has to identify patrons who are not legally allowed to be there because they are on various state or federal lists, problem gamblers who the casino has a responsibility to deny entry to and card counters or cheats, who the casino wishes to deny entry to for business reasons.

[4] Author experience.
[5] https://www.vedacheck.com/

Data Set	Number of Records	Number of Entities	Applicable Constraints
Cora	1295	112	Transitive closure
UKCD	155,888	151,082	One-to-one, transitive closure
NCVR-450	447,898	296,366	One-to-one
NCVR-full	8,237,785	8,087,018	Transitive closure
IOS	119043	54545	One-to-one, transitive closure

Table 2.4: Summary statistics and important characteristics of the data sets used in this Thesis.

2.4.3 Research Institutions

Entity resolution is also a frequent task in research institutions, as both part of data preparation in many research domains, and also for the institutions themselves. Many research domains require linking different data sets, and this often requires entity resolution. In the social sciences, studying populations, including family networks [10], occupations [91], cause of death [93] and similar topics requires population reconstruction [25] from historical birth, death, marriage and (potentially) census data, usually without unique identifiers. In addition, these data sets can be used in medicine, to study genetic factors, especially when linked with cause of death, or other risk factors that influence medical conditions and mortality. Compiling these data sets is a significant entity resolution challenge, but one which many countries are working with academia to try and solve, due to the social and medical benefits from doing so.[6]

At the institution level, bibliographic databases such as SCOPUS[7] and Thomson Reuters Web of Science[8] can have a significant impact on an institution's reputation. Publication and citation counts from such databases are a major contributing factor in many university ranking schemes [13]. Making sure that the databases are up-to-date and correctly reflect an institution's research output is a significant entity resolution task. However, failure to perform it correctly can have a major impact on an institution's reputation and ability to attract both students and academics and the task has been the subject of a variety of entity resolution work [166]. In addition, measures such as the H-Index [74] are widely used as part of promotion schemes within research institutions, which also requires research output to be correctly attributed in bibliographic databases.

[6] https://www.lscs.ac.uk/projects/digitising-scotland/, https://adrn.ac.uk/
[7] https://www.scopus.com/home.uri
[8] http://apps.webofknowledge.com

2.5 Data Sets and Experimental Details

Throughout this thesis we make use of three main data sets to evaluate our techniques. Two of the data sets are freely available (CORA and NCVR), while the UKCD is proprietry. All the data sets have been used to evaluate different entity resolution techniques in the past. A numerical summary of the characteristics of each data set is given in Table 2.4, and a more detailed description is provided below.

1. **Cora**: This is a public bibliographic data set of scientific papers that has previously been used to evaluate entity resolution techniques [165]. This data set contains 1,295 records and truth data is available.

2. **UKCD**: This data set consists of census data for the years 1851 to 1901 in 10 year intervals for the town of Rawtenstall and surrounds in the United Kingdom. It contains 155,888 individual records of approximately 32,000 households. A portion of this data (nearly 5,000 records) has been manually linked by domain experts. Fu et al. [53] have used this data set for household based group linkage where the task is to link households across time.

3. **NCVR**: This data set consists of voter registration data for the state of North Carolina in the United States of America [24].[9] It contains 8.2 million records consisting of the full name, address, age and other personal information of voters registered in the state. For most of our experiments we make use of a subset of this data set containing 447,898 records, named NCVR-450. For testing the scalability of our techniques we make use of the full data set.

In Chapter 7, we present an extension of our technique from Chapter 5 and we evaluate this extension on an additional data set:

4. **IOS**: This data set is made of birth, death, marriage and census records from the Isle of Skye in the United Kingdom. The data covers the period from 1861 to 1901, with census data recorded at 10 year intervals, and all birth, death and marriage records from the period. A portion of the records have been manually linked by historians [60, 148], however their approach favoured precision over recall so there are likely additional matches that are not recorded in the ground truth.

[9] ftp://alt.ncsbe.gov/data/

Experimental Settings and Hardware: All code for experiments was written in Python version 3.4 and run on a server with 6-core 64-bit Intel Xeon 2.4 GHz CPUs, 128 GBytes of memory running Ubuntu 16.04. Timing was done using the functionality from the built-in Python `time` module.

2.6 Summary

Entity resolution is a very important data preprocessing task, that involves determining which records in one or more data sets correspond to the same real-world entities. It has a wide variety of practical applications for government agencies, commercial organisations and research institutions. In the next chapter we summarise recent literature that is relevant to advanced entity resolution techniques, blocking, generating training data and evaluation of entity resolution results.

Chapter 3

Related Work

Entity resolution has been an active area of research since the 1950s. Recent advances in computing power have meant that more and more sophisticated techniques have become viable and the field has grown enormously in recent years. In this chapter we present a very brief overview of some important historical work on entity resolution, along with several surveys or books that describe the current state of the discipline. We follow this with an in depth description of various advanced entity resolution techniques. Finally, we present research related to the particular aspects of entity resolution that we focus on in this thesis.

The remainder of this chapter is structured as follows: in Section 3.1 we provide details of works that summarise the entire entity resolution process or provide a good overview of the entire process for those readers who desire more information than was provided in Chapter 2. In Section 3.2 we summarise the main advanced entity resolution techniques that are central to this thesis, along with recent developments or extensions that have been proposed. We then proceed to detail related work specific to each of the individual chapters in the thesis. Section 3.3 details recent research in blocking and constrained clustering, which is relevant for Chapter 4. Section 3.4 details work on active learning and generation of training data, which is relevant to Chapter 5 as well as the stable marriage problem which is relevant to our approach. In Section 3.5 we summarise research on evaluation of entity resolution and parameter tuning as well as correlation clustering which is relevant to Chapter 6. Finally, in Section 3.6 we summarise the related work and discuss the gaps we have identified which justify the need for the research described in the rest of this thesis.

3.1 Overview

The mathematical formulation of entity resolution began with Howard Newcombe in the late 1950s [120, 121]. The problem was given a probabilistic foundation by Felligi and Sunter [47] in

the late 1960s. In this seminal work, the authors provide many of the concepts and conventions that are still in use today. It contains a formal process for evaluating pairwise comparisons into three categories, 'link', 'non-link' and 'possible link' and a methodology for constructing a set of rules which minimise the size of the 'possible link' set for a given error level. The authors also discuss problems that can occur, such as sampling variability, and violation of indepedence assumptions that are critical to the probabilistic interpretation. Finally, there is a discussion of practical issues such as blocking methods to reduce the comparisons required. While the methodology has been expanded upon and refined by many subsequent works, the key ideas such as similarity in attribute values indicating two records are more likely to be about the same entity, and the concept that some attributes give a stronger indication of a match than others, are the foundation of most modern entity resolution techniques.

Subsequently, additional techniques have been developed for each aspect of the entity resolution problem. Since it is an active and broad field of research, there are many survey papers and even books that detail the entity resolution process as a whole. Survey papers from Garcia-Molina [58], Brizan and Tansel [15] and Winkler [189] surveyed the entity resolution problem around 2005. A book by Talbert [172] details the entity resolution process and how it relates to information quality. Christen [23] surveyed and summarised the state of current entity resolution techniques in 2012. Naumann and Herschel [119] provide a description of the entity resolution process along with a brief summary of some common techniques that are used at each stage. Getoor and Machanavajjhala [62] provide a tutorial on the entity resolution process along with a description of some outstanding challenges in the field. Köpcke et al. [99] provide a comparitive evaluation of different entity resolution frameworks.

In addition to the overarching problem of entity resolution, many smaller subdomains exist including privacy preserving entity resolution [155, 177, 178], real-time entity resolution [12, 27, 145, 173], parellel entity resolution [88, 94, 98], and more. Where the work in this thesis has applications in these subdomains we provide a brief summary of the ideas involved. However, a full discussion of these subdomains is outside the scope of this thesis.

3.2 Advanced Entity Resolution Techniques

We begin our treatment of advanced entity resolution techniques by first discussing collective entity resolution, a topic which became an active area of research around 2006. Collective entity resolution extends traditional pairwise matching in two ways. Firstly, it incorporates relation-

ships between entities in the matching process. Secondly, matching decisions are no longer made independently, i.e. the results from one decision can affect another. There were four approaches to collective entity resolution that were all developed around the same time and largely began work in the field. In addition to discussing collective entity resolution techniques, we also present recent research in the area of group linkage, temporal entity resolution and techniques for population reconstruction.

3.2.1 Relational Clustering Based Technique

The approach of Bhattacharya and Getoor [12] provides a formal definition of the entity resolution problem and describes three types of approaches to solving it. The pairwise (i.e. non-relational) approach, the naïve relational approach and the full collective approach. The pairwise approach corresponds to the traditional entity resolution approach of Felligi and Sunter [47]. Where relationships are incorporated at all, they are treated as attributes. For example, a *marriage* relationship might be incorporated through several attributes such as *Spouse's First Name*, *Spouse's Age*, etc. The naïve relational approach extends this to include references to other records, and a measure of relational similarity, but the calculated similarity values are not updated as the matching takes place. In other words, the matches are still performed independently. The full collective approach is an unsupervised clustering algorithm which iteratively matches the nodes with the highest combined attribute and relational similarity, and then updates the similarity with neighbouring nodes. As a result, earlier matching decisions can influence later decisions. From a practical perspective this means bootstrapping is required, and in their experiments the authors perform exact matching based on attribute values for this. Where there is ambiguity present, the attribute values of related entities are also incorporated into bootstrapping process.

Bhattacharya and Getoor [12] calculate relational similarity as:

$$sim(r_1,\, r_2) = (1 - \alpha) \times sim_A(r_1,\, r_2) + \alpha \times sim_R(r_1,\, r_2)$$

where $sim_A(r_1,\, r_2)$ and $sim_R(r_1,\, r_2)$ are the attribute and relational similarity scores respectively between records r_1 and r_2 and α is a weighting parameter with $0 \leq \alpha \leq 1$.

The paper tests the methodology on portions of three bibliographic datasets, the largest of which has approximately 60,000 references to 10,000 authors. The authors note good results for these domains, but importantly observe that the improvements are not substantial in well de-

fined and unambiguous domains. The experiments also show that increasing the neighbourhood size greatly increases computational complexity but has minimal impact on accuracy.

3.2.2 Random Walk Based Techniques

The second collective entity resolution approach we discuss uses random walks on graphs to perform entity resolution and was proposed by Kalashnikov and Mehrotra [86]. Similar to the approach of Bhattacharya and Getoor [12], the first step in the approach is using attribute based similarities to resolve easy matches to bootstrap the process. Once this is complete a graph is constructed where nodes represent entities and edges represent relationships between entities. Where there is ambiguity about the correct entities involved in a relationship, a choice node is created which incorporates the probability (either calculated based on similarity or evenly split) of the relationship being with each potential candidate. Entity resolution is performed by computing the chance of a random walk from one side of the choice node reaching the other side of the choice node. This is done iteratively, and after each iteration the probabilities on the choice nodes are updated. Because random walks contain a probabilistic component, this captures the intuition that a path through nodes that are highly connected has less importance than one through nodes with very few connections. As with the approach of Bhattacharya and Getoor [12], the authors also note that paths beyond a certain length contribute very little to the accuracy of the outcome, but come with significant computational cost.

The experiments are conducted on real and synthetic bibliographic datasets, that range in size up to 573,000 author references and 176,000 author entities. The authors note that after attribute matching was conducted, approximately 87% of the references were unambiguous, leading to 75,000 choice nodes being created. The authors also test the approach on a movie dataset with relationships between people (directors) and films. They demonstrate that the approach is useful for disambiguating references in both domains. However, the approach benefits from a certain proportion of unambiguous relationships to bootstrap the process. It also requires many user decisions about thresholds in the initial match phase in order to construct the graph as well as decisions about maximum path length, which optimisations to use, etc.

Nuray-Turan et al. developed an extension of this techique [123] which turns the approach into a supervised technique when training data is available. This allows for different paths (both in length and relationship types) to have different weightings in the random walk process. These weights can be learned from the training data and then applied to the full dataset to improve the results.

This technique improves the accuracy results over the original Kalashnikov and Mehrota [86] technique. They also compare it against a modified version of the Bhattacharya and Getoor approach [12], although this is not a fair comparison since they do not update the neighbourhood meaning it is much closer to the naïve relational method than the collective technique. However, in order to achieve a significant improvement in accuracy, they require training data, which is usually not available for entity resolution (see Chapter 5), and they also require a complex environment with variety in the types of relationships and the importance of each in the entity resolution problem.

3.2.3 Context and Information Propogation Based Technique

The third technique for collective entity resolution was proposed by Dong et al. [39]. It has many similarities with the above approaches in that it applies an algorithm iteratively to incorporate the results of previous decisions into subsequent decisions. It also makes use of relationship information to inform the matching process.

The authors describe three main ideas behind their approach. They firstly try and exploit context information, even in different attributes. For example, the fact that both the name value 'Stonebreaker, M.' and the e-mail value 'stonebreaker@csail.mit.edu' contain the sequence of characters 'stonebreaker' still provides positive evidence that the entities in question might be the same, even though the character sequence occurs in different attributes. The second feature of the approach propogates information from references. For example, an article entity is connected to a unique set of author entities. As a result, if two article entities are matched, the author entities should also be matched. The final feature of the approach is reference enrichment. If two different entities are matched, then the values for their attributes can be combined to create a more complete picture of the new, composite entity. These features are particularly useful for domains where entity types are defined in multiple places, with varying degrees of completeness of information in each place. For example, personal information management systems, or bibliographic databases that are created by crawling the web, often have incomplete representations of entities and are suited for this type of approach.

The authors test this approach on both a personal information management dataset and the Cora bibliographic dataset, in both cases achieving better results than the baseline systems. Of particular interest is the note that when working on the Cora bibliographic dataset, which contains very noisy venue information, the propagation of information from article entities to venue entities significantly increased the recall. However, it also incorrectly merged many dif-

ferent venue entities together. This result illustrates one of the risks of automated merging since entities that appear completely different end up being merged together because of an error in a reference.

3.2.4 Markov Logic

The fourth collective entity resolution technique we discuss is Markov logic networks (MLNs) which have been used to perform inference on various tasks including entity resolution [165], co-reference resolution [138] and information extraction [137].

Proposed by Richardson and Domingos [150], MLNs combine first order logic with probability theory with the aim of utilising the strong points of both fields. First order logic provides a compact and expressive representation for information including both entities and relationships. Complex rules can be represented as logical formulae, and the language of first order logic allows a concise and intuitive representation of domain knowledge. However, traditional first order logic is brittle since a single contradiction in a knowledge base invalidates it entirely. MLNs use probability theory to overcome this weakness. Probability theory and the treatment of uncertainty provides tools for performing inference with noisy or inconsistent data. It is combined with first order logic, by giving each logical formula a weight representing its likelihood of being true. Inference is then performed by a combination of Markov chain Monte Carlo simulation for resolving soft constraints or preferences, and logical satisfiability solvers for resolving hard constraints.

Aside from overcoming the brittleness of first order logic, Richardson and Domingos [150] suggest several advantages to this approach. The weights for the formulae can be learned from a database if training data is available. Hard constraints, such as transitive closure in entity resolution, or those being incorporated from domain knowledge, can be included as formulae with infinite (or negative infinite) weight. While there are many approaches to inference in domains with either hard or soft constraints, far less attention has been paid to the problem where they are combined [136]. MLNs partially address this deficiency through a combination of a logical walk-sat-solver to move between solution spaces, and Gibbs sampling to move within solution spaces [150].

The technique does have disadvantages. The predicates and formulae have to be specified prior to performing inference (as opposed to learning them directly from the data) and if there is no training data available, the results can be sensitive to the formulae weighting. Additionally, inference in MLNs is slow, even at an approximate level, and is intractible to solve exactly [150].

This means that for entity resolution, the challenge becomes one of reducing the size of the network in order to obtain reasonable running times. We propose a blocking technique that can control the size of blocks, and thus the network size, in Chapter 4.

A number of techniques have been proposed to determine the parameters of an MLN using a variety of different methods [79, 96, 163]. However, these techniques all rely on training data being available which is often not the case for entity resolution [23]. Moreover, due to the fact that the classes of matches and non-matches are typically unbalanced, it can become very expensive to generate training data [184] (see Chapter 5 for further details).

MLNs have been applied to the entity resolution problem by Singla and Domingos [165]. They provide some theoretical evaluations of the problem and conclude that one of the limitations of traditional entity resolution techniques is that they make independent match decisions. For this to be valid, the data must be independent and identically distributed which is not true in most cases. They provide potential candidates for evidence predicates for entity resolution using Markov logic and test their approach on the BibServ and Cora datasets.

Overall, their experiments show that MLN based entity resolution can achieve high quality results (in terms of the evaluation measures described in Section 2.3), however they also highlight the poor scalability of the technique. Even though their datasets are very small, at most 10,000 records, they still have to perform canopy clustering (essentially traditional blocking with overlapping blocks [22]) in order to make the inference tractible. We confirm this problem with some simple experiments of our own, the results of which are shown in Figure 4.1 in the next chapter.

Rastogi et al. [146] attempt to improve the scalability of entity resolution techniques. They treat the matching component as a black box, but make use of an MLN based approach in their experiments. Their approach extends the canopy clustering used by Singla and Domingos [165] by including message passing, where information is passed between canopies if it is relevant to the matching decisions. This approach is perfect for MLN based matchers, since they build up evidence through individual formulae. Where a matching decision has high evidence, but still insufficient for a match to be made, this accumulation of evidence can be passed between canopies so that in combination, sufficient evidence can be found.

The authors note that there are no theoretical guarantees of optimality in the matching process, but state that in their experiments, there was minimal or no loss in matching quality. Since MLN based techniques perform collective entity resolution (match decisions are influenced by other match decisions), it is possible for gradual evidence to accumulate across the canopies and

for one decision to influence another very far removed from it in the network. However, in practice this is extremely unlikely. Both Bhattacharya and Getoor [12] and Kalashnikov and Mehrotra [86] showed that in their experiments neighbours more than one link removed had little influence on the results, and in most domains it would require a very specific set of circumstances for matching decisions to propogate further than this.

The main criticism of the approach proposed by Rastogi et al. [146] however, is that while they improve scalability to the point where they can run an MLN based matcher on a dataset of approximately 58,000 records, this is still quite small in the context of modern datasets. While they don't provide the specifics of their implementation, it is likely they are making use of Alchemy [97], the same software used by Singla and Domingos [165] since they cite that work in the context of the entity resolution process. It is possible that using the more recent, and generally more efficient package of Tuffy [122] along with their canopy clustering and message passing technique could improve scalability further. However, it is still unlikely that the speedup would be sufficient to work with datasets of millions of records or more.

3.2.5 Group Entity Resolution

Group linkage can be used in domains where hierarchies of records exist. Since these hierarchies are usually one to many relationships, best results are achieved by matching not just between individual records, but between groups of records as well. An example of this is historical census matching, where only matching individuals may achieve suboptimal results. However, when results for individuals are combined with comparisons between households the overall results can be improved significantly.

The group linkage task has been studied previously. On et al. [126] formalises the notion of group linkage as matching a set of references, tied together in some way, with other sets of references. The process described extends traditional entity resolution by performing pairwise entity resolution on the individual records to determine potential matching pairs, and then matching groups, where a high proportion of records in each group are matching pairs. The process is computationally intensive, so they provide blocking techniques to perform this efficiently.

The approach by Fu et al. [54] uses multiple instance learning. Group links are treated as bags and record links are treated as instances within a bag. The problem becomes a bag level classification problem. The technique is supervised, making use of training examples of both positive and negative bags to train a support vector machine classifier. The experiments were performed on both real and synthetic data. The real data is the UKCD dataset used in this thesis

and described in Section 2.5.

The two works above treat groups of records as sets, but make no allowance for any structural information that might be present within groups. Fu et al. [53] further extend the group linkage approach by taking this structural information within groups into account in a graph matching technique. A graph is constructed for each group, containing relationship information between the different records as edge attributes. For example, in the experiments on household census data, the edges have attributes for *age difference*, *generational difference* and *role pair*, all of which are (in theory) static over time. The task then becomes a graph matching problem, where vertex similarity (pairwise similarity) and edge similarity (relationship structure) are combined to match the groups.

The approach of Christen et al. [34] extends the group linkage approach by also considering how groups evolve over time. It attempts to solve the problem by assigning a label to each node in the entity relationship graph (essentially a simple blocking approach) and then performs graph matching based on the labelled graphs. The approach also works in an iterative fashion whereby the high quality matches are identified first using strict matching criteria, and then in each subsequent iteration, the matching criteria are relaxed in order to increase the recall of the approach.

Mcconville et al. [114] propose an approach for entity resolution in graphs which starts by making use of string similarity to detect duplicate nodes. However, the approach makes use of community detection algorithms to determine and exploit the community structure in the similarity graph, meaning the approach can be applied to group linkage problems where the groups are not clearly defined in the data. Nanayakkara et al. [118] also employ a clustering based approach to generate groups for use in group linkage. The proposed approach makes use of temporal constraints, modelled as plausibility values, to capture both hard and soft constraints, in order to incorporate domain knowledge. For example, a minimum time between births of children for a single mother, minimum and maximum age of a mother at time of birth, etc.

3.2.6 Temporal Entity Resolution

At a conceptual level, temporal information does not change the entity resolution task. A timestamp attribute captures the date a recorded was created (in other words, the date the information in the record was current) and it is another attribute that can be used in the similarity calculation when determining whether two entities are the same. However in practice, this attribute can be incorporated in a more effective fashion. Rather than having its own similarity function,

differences in the values of the timestamp are used to modify the results of other similarity calculations.

There are various ways of dealing with this situation. The technique described in Li et al. [106] is to discount similarity scores based on the time difference between the records using measures of agreement decay and disagreement decay. The motivating idea is that the longer the time difference between two records, the more likely that some of the attributes or relationships have changed meaning the similarity scores should be adjusted.

A variation of this approach is used by Christen and Gayler [28] for query time entity resolution with temporal information. After potential matches have been determined, the similarity scores are adjusted based on the time difference between the query record and the potential match. Where the information is available, a time history for the entity can be constructed which can also be used to adjust the similarity scores. These factors are then combined to determine the most likely entity in the database to match to the query record. Similarly, work by Hu et al. [76, 77] extends the agreement and disagreement decay model to include supporting attributes which allows the authors to refine the parameters for the specific circumstances of individual records.

Chiang et al. [21] extend the approach of Li et al. [106] but add an additional phase to the process. Traditional (non-temporal) entity resolution is conducted first, and then a technique for detecting evolving entities is applied. This technique is effective, however it is very computationally intensive and does not scale well to large datasets. Along similar lines, the work of Althoff et al. [3] aims to build up time-lines of entities and incorporate such aspects as changes of name, occupation and other attribute values. This is very similar to the problem of entity tracking on the web [185] and building of temporal knowledge graphs [64] and many of the same approaches are used in both domains.

3.2.7 Population Reconstruction

Population reconstruction is an entity resolution problem that involves matching historical records from sources such as birth, death and marriage registers, land titles and other historical sources. The reconstructed populations can then be analysed to answer questions in areas such social sciences, genetics and medicine [92]. However, the entity resolution problem is more complex than a traditional pairwise matching problem. There is often a limited amount of information available from each individual record, for example a birth certificate might have the name and date-of-birth of the newborn, along with the name and possibly the age of each of

the two parents. With such a sparse attribute set for each individual it can be very difficult to unambiguously resolve each person on a certificate, particularly if the name happens to be a common one. As a result, it is extremely important to make use of the relationship information in order to solve this problem.

Eframova et al. [41] extend the traditional entity resolution approach by looking at common attributes across the different types of records (birth records, death records, etc.). In a subsequent work by the same authors [42], additional information such as name popularity, geographical distance, and coreference frequency are added to improve the entity resolution results. Rahmani et al. [142] attempt to solve the problem by weakly linking the records (i.e. by using a low similarity threshold for matches) and then by performing a random walk on the linked graph that is created in a similar manner to the collective entity resolution technique of Kalashnikov and Mehrotra [86].

Schraagen and Kosters [157] treat this problem as a graph consistency problem. The approach operates in two phases, firstly a seeding algorithm uses a strict condition to generate candidate matches, after which event consistency is used to detect problems in the seeded results or identify other candidates to be matched. Consistency checking is done through incorporating conditions that reflect the particular domain, for example a minimum of one year between births in a family or the requirement of a marriage record before a birth record. In this way, family records can be built up with few or no string comparison operations which improves the efficiency of the technique.

Kouki et al. [102, 103] applied a technique based on probabilistic soft logic (which is similar in many respects to Markov logic networks [150]) which uses logical rules to express the different roles, relationship types and possible links that can occur in population reconstruction. However the technique has improved scalability over Markov logic networks while still capturing the expressive power of first order logic to capture domain rules and the statistical inference from probabilistic graphical models.

The approach proposed by Malmi et al. [111] attempts to solve this problem using a combination of binary classification and collective classification techniques in order to construct family trees to study social demographics. The approach refines the binary classification approach by incorporating domain constraints on the number of spouses an individual is likely to have and adopts a greedy algorithm for choosing the most likely candidates.

Christen [25] proposed a technique that attempts to solve this problem in several stages, firstly by comparing individuals to generate initial matches, then by applying temporal and

one-to-one or one-to-many constraints (such as a person cannot be born after their death), and each person should have at most one death record. After the constraints have been applied, the author makes use of relationship similarity and group linkage techniques in order to further improve the quality of the results.

3.3 Blocking

One limitation of many advanced entity resolution techniques is their poor scalability. While blocking has been used to overcome this problem for traditional entity resolution (as have other techniques such as indexing [73], hashing [107], etc), for advanced entity resolution techniques, a greater degree of control over block sizes is important, in particular the size of the largest block. Many techniques such as collective entity resolution techniques also require disjoint blocks which eliminates techniques such as canopy clustering [112]. We provided a basic overview of blocking and indexing in Section 2.3, and several recent surveys of blocking techniques have been conducted [22, 43, 99, 132, 134, 170]. In the following we briefly describe some key prior research that relates to our own blocking techniques described in Chapter 4, in particular, blocking techniques that adopt an iterative approach or that aim to control the size of blocks.

Several iterative blocking techniques for entity resolution have been proposed in recent years [37, 145, 187]. Whang et al. [187] proposed an iterative blocking process in order to perform entity resolution. Rather than processing each block individually, the approach propagates the results from processed blocks (i.e. where records have been compared) to inform decisions in subsequent blocks. Once two records are determined to be a match, they are merged, and the resulting new record is propagated into other blocks where the combination of attributes may cause previously undetected matches to be found. The results of previous comparisons are stored so that comparisons are not repeated unnecessarily. However, these techniques give no control over the size of the blocks that are produced.

Das Sarma et al. [37] also developed an iterative blocking approach that combines splitting and merging to efficiently block large-scale data sets for entity resolution. The work makes use of labelled training examples to generate blocking schemas in an automated fashion. The technique includes a post-processing step where small blocks are merged to increase recall based on a heuristic. While this technique gives some control of the block sizes, it does not enforce hard size contraints and requires labelled training examples which are often unavailable for real-world entity resolution problems (see Chapter 5).

Papadakis et al. have proposed various iterative approaches to reducing redundancy in the number of comparisons required [128, 129, 130, 131, 133]. These approaches are based on the idea that in order to achieve high quality results, it is often necessary to block the data set several times using different functions or attributes. However, there may be considerable overlap between different blocking runs which typically results in repeated comparisons and so the authors have proposed various approaches to minimise or eliminate this duplication. While such approaches may increase the overall efficiency of the entity resolution process, they still do not give control over the size of the blocks being created, which limits their utility when combined with advanced entity resolution techniques with poor scalability.

Blocking and indexing techniques that place limitations on the number of comparisons that are permitted have been proposed in the context of real-time entity resolution where operational requirements may not permit large blocks. Ramadan et al. [145] modified the sorted neighbourhood approach [23] for real-time entity resolution to allow for updating a blocking key value tree in response to a query. The authors examined an adaptive window-size approach to vary the number of candidate records returned for comparison based on either a similarity or a size threshold. The similarity between neighbouring nodes can be pre-calculated to reduce query times. The same authors also propose a forest based approach [143] with similar properties that attempts to reduce the impact of typographical errors while still satisfying operational requirements. Vieira et al. [181] proposes a similar approach using a cluster index and a similarity index and which aims to reuse previous similarity calculations to minimise the query time. However, these approaches do not enforce minimum and maximum size constraints nor do they generate individual blocks which makes them unsuitable for applications such as collective entity resolution.

The approach of Kirsten et al. [94] aims to provide some degree of control of the block sizes for parallelisation of entity resolution. However, where blocks are too large for a single match task, their approach simply splits the matching process across multiple cores in the entity resolution step so that all records in a single block are still compared. This makes their approach unsuitable for advanced entity resolution techniques that utilise a graph based approach, where there is no clear method for partitioning the groups while still maintaining the entity resolution quality.

Verroios and Garcia-Molina [179] consider the problem of top-K entity resolution, i.e. finding the K entities with the most records in a data set. This approach is suitable where it is only important to resolve the most popular entities and the rest can be ignored. The approach relies

on locality sensitive hashing to place constraints on the size of the blocks generated. Although locality sensitive hashing can be computationally expensive, because the technique is only interested in the K largest entities, records that can be determined to be outside the top K entities, can be discarded very quickly, thus speeding up the overall processing time. While the authors demonstrate the effectiveness and efficiency of the approach, it is not applicable for general entity resolution problems due to its focus on a (relatively) small subset of the entities in the data set.

3.3.1 Constrained Clustering

Since there are many similarities between size constrained blocking for entity resolution and the problem of constrained clustering, we briefly summarise recent research on this topic as well. Size constrained clustering, extends the traditional clustering problem by requiring each cluster to also satisfy size constraints (such as a minimum and maximum number of records in the cluster). For further information on other aspects of constrained clustering see the work of Dinler and Teral [38] and Gupta [67]. However, we note that as a rule, current size constrained clustering techniques are only evaluated on data sets with a few thousand points or less due to poor overall scalability. Many of them also require a distance calculation to be performed for every pair of records. This means that while the ideas are relevant to blocking for entity resolution, the techniques themselves are of limited value.

Zhu et al. [194] examined the clustering problem under size constraints, although not in the context of entity resolution. The authors proposed an approach to produce clusters of a certain size, which can also be relaxed to a size range. Nevertheless, the authors only tested their approach on small data sets that have less than one thousand records or no more than three clusters. Their approach also requires computing the complete similarity between all pairs of records in a data set, which limits its usefulness for blocking in entity resolution tasks where the aim is specifically to avoid this complete pairwise comparison. Work by Malinen and Fränti [110] and Rebollo-Monedero et al. [147] have the same limitations.

Ganganath et al. [57] proposed a modification of the traditional K-Means clustering algorithm that uses prior knowledge to create the initial cluster centroids. Given such prior knowledge, they are able to create clusters of the specified size. As with other constrained clustering algorithms, their approach is only tested on relatively small data sets. In addition, the prior knowledge to create the clusters is not necessarily available in the context of entity resolution.

Klami and Jitta [95] examine a variant of size constrained clustering, where all clusters should

be approximately the same size and this size should be constant with respect to the number of data points. Solutions to this problem could be useful for blocking since larger data sets would produce more clusters rather than larger clusters. Similar to the blocking approach we discuss in Chapter 4, the authors use an iterative approach where clusters are formed and then modified or rejected if improved clustering arrangements can be found. However, the approach is only tested on data sets with less than 1,000 points and relies on extensive sampling in order to produce a clustering result, which limits its usefulness for entity resolution. Subsequent work by the same authors [84] expands on the techniques, but still suffers from the same limitations in the context of entity resolution.

3.4 Training and Bootstrapping Data

Another limitation of advanced entity resolution techniques is that most require either training data or a bootstrapping process in order to work. In Chapter 5 we propose a technique for generating such training data that relies on domain constraints such as one-to-one and one-to-many matching restrictions. In this section, we briefly summarise recent literature that is related to entity resolution techniques that can deal with a one-to-one matching constraint and the so-called *stable marriage problem* (defined in Section 3.4.2) which is a similar problem to entity resolution with one-to-one matching constraint. We also summarise recent research on active learning since it is an alternative method for creating training data for entity resolution and the technique we propose in Chapter 5 can incorporate an active learning component.

3.4.1 Entity Resolution with Constraints

As we discussed in Section 2.2, entity resolution often has to satisfy certain constraints. One common constraint is a so called *one-to-one* constraint, where each record in a data set can match to at most one record in another other data set. Because matching decisions are made independently, traditional threshold based entity resolution techniques cannot incorporate one-to-one matching constraints. However, collective entity resolution techniques exist which can deal with one-to-one constraints in entity resolution problems by making *collective* decisions.

For example, Markov logic networks [150] can incorporate a hard constraint such as one-to-one constraint using a formula with infinite weight. Similarly, other entity resolution processes that can incorporate constraints have been developed by Arasu et al. [6] which make use of a domain independent specification language in order to express any restrictions that may apply

on the matching process. The work of Burdick et al. [17] uses a similar approach and can also incorporate constraints such as one-to-one matching and transitive closure in the logical rules. The work of Shen and Wang [162] also makes use of rules, but adds weights in order to better resolve cases where constraints are conflicting.

3.4.2 The Stable Marriage Problem

There are many parallels between one-to-one matching constraints and the *stable marriage problem* [55, 82]. In the stable marriage problem a group of n men and n women each have a ranking (or preferences) of the other group in terms of their desirability of marriage. The problem is to find an optimal pairing of the men and women, given these preferences. To see how this relates to entity resolution, if we view the two data sets to be linked as the sets of men and women in the stable marriage problem, and the similarity between records as the preferences (i.e. we assign preferences for each record in order of decreasing similarity), then we end up with a very similar problem to the stable marriage problem. The fact that each person in the stable marriage problem can only be married to one person is equivalent to a one-to-one matching constraint in the entity resolution problem.

However, relaxations of the traditional stable marriage probem would normally apply to the equivalent entity resolution problem. The preference lists may be incomplete (due to blocking or other efficiency requirements) and we also may not have a complete pairing (i.e. not every record has a true match in the other data set). The case of incomplete preference lists has been studied in the context of stable marriage [81]. Additionally, the one-to-many constraint has been studied in the stable marriage context by looking at single preference list variants of the problem [80].

The algorithm for finding a solution to the stable marriage problem as proposed by Gale and Shapley [55] has quadratic complexity (i.e. $O(n)^2$ where n is the number of pairs) and relaxations of the problem such as incomplete preference lists, ties in the preference lists (which in the case of entity resolution is equivalent to identical similarity values) and incomplete pairings all increase the computational complexity to the point where it becomes largely intractible for problem sizes that are relevant to entity resolution tasks. Works by Király [90] and Munera et al. [117] both propose approximation algorithms for solving more difficult versions of the problem. Finally, Chen et al. [19] proposed an algorithm for another variant of the problem where each candidate can have multiple preference lists. In an entity resolution context this is equivalent to having multiple similarity scores between each record, which may be appropriate for some problems.

3.4.3 Active Learning

Active learning [160] has been used in many fields as a way to generate training data so that supervised classification techniques can be applied. Active learning belongs to the class of semi-supervised approaches to classification problems, and uses an oracle (e.g. a domain expert, crowd sourcing, etc.) to manually label examples in order to create a set of training data.

Active learning techniques vary in the way examples are selected for manual classification. Variance reduction based techniques aim to reduce generalisation errors by minimising output variance and have been applied to classification models such as neural networks [108] and conditional random fields [161]. Density based techniques such as those proposed by Xu et al. [191] and Settles and Craven [161] aim to ensure that the examples chosen for manual classification are not only uncertain, but also somehow representative of the underlying distribution. This means that the expert does not waste time classifying outliers, which may be uncertain, but have minimal impact on the overall classification quality.

Active learning has been successfully applied to the entity resolution problem [5, 11, 153, 184]. As described in Section 2.3, one particular problem when using active learning for entity resolution is that the classes of matches and non-matches are typically very unbalanced. As a result, a challenge with each of these techniques is to select representative examples of each class to present to the expert for manual classification, since randomly selecting record pairs for manual classification may produce no matches. Because of this, an active learning approach for entity resolution needs to employ a strategy to ensure that enough examples of both matches and non-matches are presented to the oracle for labelling, for the classifier to learn the characteristics of both classes.

This challenge has been addressed in several ways. The approach used by Arasu et al. [5] assumes monotonicity of precision (i.e. record pairs with higher similarity are more likely to be matches than record pairs with lower similarity). The active learning approach selects candidate pairs with decreasing similarity scores until a pre-defined precision threshold is reached. The monotonicity of precision assumption means that by starting with the most similar record pairs, the approach starts with the best ratio of matches to non-matches and stops selecting new candidate pairs for manual classification once the ratio drops below a certain threshold.

The approach proposed by Dal Bianco et al. [36] is similar to that of Arasu et al. [5] in that it relies on monotonicity of precision. However, the approach uses a two stage sampling strategy to focus the majority of the manual labelling effort on the so-called 'fuzzy zone', i.e.

those record pairs where the classifier has most difficulty in determining whether they are a match or non-match.

The approach proposed by Wang et al. [184] looks at clusters of record pairs (based on attribute similarity values) and attempts to determine whether each cluster consists of matches or non-matches by manually labelling some examples. If manually labelling record pairs from a cluster yields both matches and non-matches, the cluster is subdivided and more examples are manually labelled until a required level of purity (i.e. labelled examples are nearly all matches or nearly all non-matches) is reached. The work of Qian et al. [140] uses a rule learning based approach, which tries to make sure the rules are different enough to cover a wide variety of match scenarios in order to increase recall.

The concept of ambiguity is also applied in active learning [154], although it usually means something different to what we propose in Chapter 5. In the context of active learning, ambiguity usually refers to uncertainty about whether the classification should be a match or non-match, rather than which candidate to match to. For a threshold based classifier, it means that the similarity between two records is very close to the threshold, while for a probabilistic classifier it means that the probability that a record pair is a match is close to 0.5. While some approaches do look at the distribution [161], it is looking at the probability of each class label, rather than the probability of each potential matching candidate.

Finally, while traditional active learning techniques make use of an all-knowing domain expert to perform the manual classification, from a practical perspective this may not always be possible. Variations have been proposed which deal with noisy oracles, i.e. experts who return the wrong classification result [40]. Instead of a domain expert, other techniques make use of crowd-sourcing to perform the labelling [180, 183]. While the technique we propose in Chapter 5 does not explicitly deal with these variations, there is no reason why different forms of oracles could not be used instead of the traditional domain expert.

3.5 Evaluating Entity Resolution

Another limitation of entity resolution in general, and advanced entity resolution techniques in particular is that it can be very difficult to evaluate the quality of the entity resolution outcome. This is particularly problematic since entity resolution often requires iteration and refinement as parameters are tuned and techniques are tried, tested and discarded. This may result in an evaluation of each iteration, which can be very time consuming and costly. Measures that

are traditionally used to evaluate entity resolution include precision, recall, the F-measure, and accuracy [23, 30, 100] (described in Section 2.3). Menestrina [116] summarised various clustering measures for entity resolution and proposed the new measure *generalized merge distance* which can be customized to suit an individual entity resolution problem. Maidasani et al. [109] summarised a number of different pairwise, clustering and edit-distance based measures for evaluating entity resolution results. Recent work by Hand and Christen [70] describe problems with the interpretation and use of the F-measure, and propose how it can instead be mathematically expressed as the weighted arithmetic mean of precision and recall. O'Hare et al. [125] proposed a blocking evaluation metric that also takes into account the outcome of the linkage stage in the evaluation. However, all these measures rely on ground truth data being available. In the absence of such ground truth data, a costly manual evaluation of the results is required to determine whether the results from the entity resolution process are good enough. A fully unsupervised evaluation technique for entity resolution results remains an open challenge [23].

3.5.1 Correlation Clustering

There are many similarities between the problem of correlation clustering and the triagle counting approach for evaluating entity resolution we propose in Chapter 6. Bansal et al. [9] studied the problem of correlation clustering, i.e. given a graph with positive and negative edges, find a clustering that minimises disagreements or maximises agreements. Subsequent work has advanced this area in fields such as community detection in graphs [51] and image segmentation [89]. Further optimisations have been proposed that extend the algorithms to large scale problems that are more relevant to entity resolution [8]. Also in the entity resolution domain, Wang et al. [182] proposed a technique that makes use of graph consistency to ensure that group-wise entity resolution results are consistent with pair-wise entity resolution results. The authors proved the problem is co-NP-complete and proposed two approximate algorithms, one focusing on effectiveness and one on efficiency. Both algorithms aim to produce a consistent entity resolution result, but do not consider whether inconsistent results can provide additional information. For a complete summary of current work in correlation clustering see the survey paper by Pandove et al. [127].

Singla and Domingos [164] proposed an approach that uses propagation of information to improve the entity resolution process. The approach uses conditional random fields to make matching decisions and allows for results from one decision to be propagated to inform other match decisions. However, the approach still resolves transitive closure as a post processing step

to ensure consistency, rather than investigating whether any inconsistencies detected could be used to improve the entity resolution process.

3.6 Summary

Entity resolution is an active area of research. While many of the key ideas were developed in the 1950s and 1960s, recent advances in computing power, along with rapid growth in the amount of data being collected have seen entity resolution attract significant research effort. In this chapter we have provided a brief overview of the historical development of entity resolution, followed by a description of many advanced entity resolution techniques that are central to this thesis, including collective entity resolution, group linkage, temporal entity resolution and population reconstruction. We have also summarised selected works from those aspects of entity resolution that are particularly relevant to the research questions we address in this thesis, along with the techniques we propose in Chapters 4, 5 and 6.

As a general rule, most of the advanced entity resolution techniques we described suffer from at least one of the limitations that we presented in Section 1.2, and many suffer from all three. Most of the techniques have poor scalability, and in several cases, the scalability is very poor. When they are evaluated in an academic environment, the impact of the poor scalability is limited due to the size of the data sets chosen for evaluation purposes. However, in real-world applications, this is unlikely to be possible and the size of the data sets can be orders of magnitude greater than what the technique was originally demonstrated on. However, as the academic experimental evaluations show, if the techniques are applied on sufficiently small data sets, they can be very effective. As a result, blocking techniques that allow control of the block sizes could be very effective when used with advanced entity resolution techniques.

While some current blocking techniques provide a degree of control over block sizes, they often do not enforce hard size constraints, and in many cases do not produce disjoint blocks. This makes them unsuitable for advanced entity resolution techniques with poor scalability or which require disjoint blocks. While many size constrained clustering techniques do provide both features, the clustering techniques themselves have poor scalability which makes them unsuitable for blocking for entity resolution.

Many of the advanced entity resolution techniques described in Section 3.2 are supervised, meaning they require labelled training data before they can be applied. In addition, while some collective entity resolution techniques such as those of Bhattacharya and Getoor [12] and

Kalashnikov and Mehrotra [86] are unsupervised, they require a bootstrapping step in order to generate the initial relationship information used in the entity resolution process. In academic environments, problems with training data can again be overcome by appropriate choice of data sets (i.e. choosing a data set where ground truth data is available). However, for real-world entity resolution applications, this may not be possible.

Current approaches to generating training data often resort to active learning, and a suitable domain expert (or crowd sourcing, etc) may not be available in real-world applications. In addition, as we show in Chapter 5, the current approaches to bootstrapping collective entity resolution work well in some situations, but are not effective in others so more sophisticated techniques are needed before they can be used. Approaches to solving the stable marriage problem are relevant for this problem, however they also have poor scalability which again limits their appropriateness for entity resolution.

Finally, many of the advanced entity resolution techniques are very complex, with a number of choices and many parameters that require tuning in order to achieve good results. While domain expertise and experience can assist with this process, in the absence of evaluation data (see above), tuning becomes very challenging because it is difficult to determine whether the results of a particular setting are good enough. Current approaches to evaluation all rely on the ability to calculate (or estimate) the confusion matrix (described in Chapter 2) which in practice can be very difficult for entity resolution.

As a result, even though entity resolution has been the subject of a large amount of research, there are still improvements to be made, particularly with respect to the application of sophisticated entity resolution techniques in real-world scenarios. In the next three chapters, we present techniques to address some of these gaps.

Chapter 4

Addressing Scalability Through Size Constrained Blocking

As described in Chapters 1 and 3, one of the biggest practical limitations of many advanced entity resolution techniques is their poor scalability. Pairwise comparison of each record makes traditional entity resolution a quadratic problem, and many advanced entity resolution techniques have scalability that is worse (and often much worse). The motivation for the work in this chapter came from experiments testing the scalability of Markov logic networks [150], the results of which are shown in Figure 4.1. We observe that beyond a certain network size (in this case approximately 1,000 records), scalability becomes exponential. Given 1,000 records is very small by the standards of modern data sets, applying a technique like Markov logic networks on real-world entity resolution problems presents a significant challenge.

Many of the advanced entity resolution techniques discussed in Chapter 3 are only evaluated on data sets which are relatively small (tens or hundreds of thousands of records at most), meaning that the impact of poor scalability can be limited. It also means that the iterative nature of the entity resolution process (discussed further in Chapter 6), where parameters and techniques are tuned and refined through repeated application, testing and evaluation, can also be completed in a reasonable amount of time. However, in many real-world entity resolution problems, data sets are larger, and possibly much larger, than those typically used in academic experimental evaluations. This significantly reduces the feasibility of many advanced entity resolution techniques for real world applications.

From the early days of probabilistic record linkage [47], different techniques have been employed to get around the quadratic scalability of the entity resolution problem. We gave an overview of blocking and indexing in Section 2.3 and recent advances relevant to this chapter were summarised in Chapter 3. However, we note that there are two key features that are impor-

tant for blocking approaches specifically aimed at advanced entity resolution techniques, control of the block size (particularly the maximum block size) and the ability to create disjoint blocks. Indexing techniques such as sorted neighbourhood [73] provide the first but not the second, while many blocking techniques give the reverse [47, 124, 135]. In addition, a blocking technique still needs to ensure that both records in each true match pair end up in the same block, otherwise the blocks generated will not be useful in the rest of the entity resolution process. In other words, any blocking technique still needs to produce a sufficient level of block quality (as measured by pairs completeness - see Section 4.4).

The reason why control of the block size is important is easy to see. If scalability of a technique is quadratic or worse, then we need to be able to limit the worst-case scenario in order to ensure entity resolution can be performed in a time frame of practical use. However, it is also important not to have too small blocks, since this can reduce pairs completeness (see Section 4.4) to the point where too many true matches are missed in the blocking step. The block size distribution of many blocking methods is very skewed (see Figure 4.4 for examples of block size distributions and Figure 4.6 for examples of the largest block size), and this severely degrades the performance of even traditional entity resolution. In the case of many advanced entity resolution techniques, running them on blocks of tens or hundreds of thousands of records may simply never finish. However, simply adding additional attributes to the blocking approach to split blocks (or applying another scalability improving technique such as indexing or hashing to each block) may result in too many missed comparisons.

The reason why disjoint blocks are important is perhaps less obvious. However, many entity resolution techniques rely on a graph based model of the underlying data (exactly how the data is represented in the graph varies between techniques). Techniques in this category include collective entity resolution techniques [12, 39, 86, 150] and many extensions and adaptations [20, 45, 123]. As a result, even though an indexing technique such as sorted neighbourhood [73] limits the pairwise comparisons to records that are 'close' in the index, it essentially still creates a single graph or network for each of the above techniques. For a data set of any realistic size, this still gives a computationally intractable problem.

This second problem can be overcome by subdividing the index into sub-sections, however this simply creates different problems. If the sub-sections don't overlap, then subdividing the index potentially misses matches at the edges of a sub-section. If the sub-sections do overlap then there is repetition of work (since the same records are being compared multiple times) and there is also the problem of how to combine potentially inconsistent results since a record pair

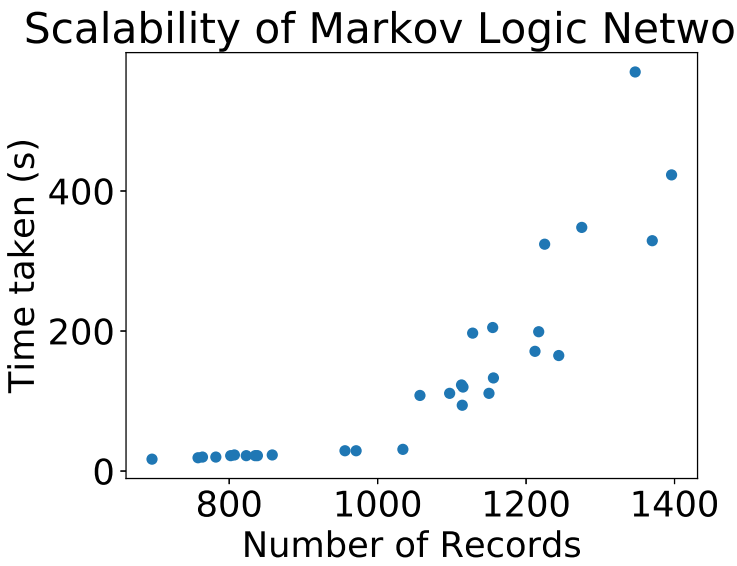

Figure 4.1: Experimental results showing the running time of entity resolution using a Markov Logic Network with different numbers of records.

might be classified as a match in one sub-section, and a non-match in another (for example due to the presence of different relationship information).

In this chapter, we present a blocking approach that aims to solve both these problems simultaneously. Our approach works in an iterative fashion, where we first split a data set into blocks using a blocking key (defined in Section 4.2), then cluster any blocks that are too small. Any blocks that are still too large are split again using a different blocking key, small ones are again clustered, and so on, repeatedly splitting and clustering until all the blocks are within a specified size range.

While the main purpose of this technique is to provide a practical blocking approach for advanced entity resolution techniques, there are several other problem domains where the technique would also be useful:

- In real-time entity resolution [12, 27, 145, 173], operational requirements mean that only a certain number of comparisons can take place within a specific (or limited) time-span (e.g. sub-second). Therefore, blocking is important to ensure that these comparisons are with the candidate records that most likely correspond to matches. In such cases, having a maximum block size ensures that operational requirements can be satisfied.

- In privacy-preserving record linkage [177], there may be privacy requirements on both the minimum and maximum block size. For example, to guarantee k-anonymous privacy [176]

it is necessary that each block contains at least k records. In addition, if all blocks have a similar size this reduces the vulnerability of the entity resolution process to frequency-based attacks [177]. In this situation it is important to produce blocks in the specified size range.

- Finally, for parallel or distributed entity resolution [88], it is important for load balancing that the blocks be of similar sizes, so that each processor or machine takes approximately the same amount of time. In such cases it is also important that blocks be disjoint so that each block can be sent to a single processor or machine.

It is also worth noting at this point what we are not trying to do with this technique, namely we are not trying to produce a blocking technique that improves upon traditional blocking evaluation metrics such as *Pairs Completeness* or *Reduction Ratio* (both defined in Section 4.4). Instead our aim is to produce a technique that achieves similar blocking quality to commonly applied blocking or indexing approaches, while also controlling the block size and satisfying the requirement for disjoint blocks.

Motivating Example: Throughout this chapter we make use of the data set in Table 4.1 to motivate the problem and illustrate our technique. The data set consists of seven (fictitious) records. Each record has four attributes: *Record ID*, *First Name*, *Surname* and *Postcode* (Zip-Code). Our goal is to split this data set into disjoint blocks, where each block has either two or three records in it.

Record ID	First Name	Surname	Postcode
r_1	John	Smith	2000
r_2	Johnathon	Smith	2009
r_3	Joey	Schmidt	2009
r_4	Joe	Miller	2902
r_5	Joseph	Milne	2902
r_6	Paul		3000
r_7	Peter	Jones	3000

Table 4.1: Example data set.

The remainder of the chapter is structured as follows: in Section 4.1 we provide the high level outline of our blocking approach. In Section 4.2 we present the notation, definitions and algorithms for our technique. In Section 4.3 we present a refinement of the technique that allows for more control of the blocking process by relating block size to block quality in situations where there is more flexibility in the size of the blocks that can be used in the subsequent

entity resolution process. In Section 4.4 we evaluate our approach on three real-world data sets and compare the results to several common blocking and indexing techniques. In Section 4.5 we discuss the computational complexity of our approaches as well as some practical considerations for their use. Finally, in Section 4.6 we present our conclusions and possible directions for extending our approach in the future.

4.1 Overview

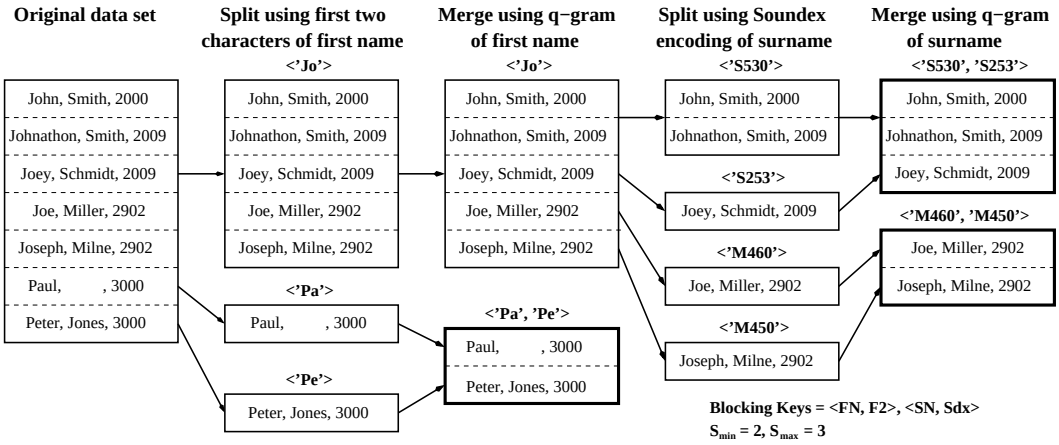

Figure 4.2: Example of algorithm flow using the data set from Table 4.1.

The idea behind our approach is to recursively split and merge blocks until we end up with all blocks in the desired size range. To do this we use a sequence of blocking keys to split a set of records into blocks, and a measure of similarity to merge small blocks back together. For example a blocking key might be the first two characters of a *Firstname* attribute. This would separate the records by placing all records with the same two characters of *Firstname* in the same block ('John' and 'Joe' together, 'Mary' and 'Madeline' together, etc.). Blocks that are in the desired size range are finished, and we do not modify them further. Blocks that are too small are clustered, based on the similarity between their blocking key values (as measured by some simple metric) and blocks that are too large are split again using a different blocking key. The newly created blocks are clustered if they are too small, split again if they are too large and so on. This continues until either all blocks are in the desired size range or we run out of blocking keys in our sequence to perform the splitting. The general process is shown in Figure 4.2 and described in more detail in the next section.

4.2 Approach

The idea behind our approach is to recursively split and merge blocks until we end up with all blocks in the desired size range. We present two versions of our approach which differ in how the merging is conducted during the clustering phases.

4.2.1 Definitions and Notation

We assume that a data set $\mathbf{R}$ consists of records, each of which is associated with a set of attributes $\mathbf{A}$. The value of an attribute $a \in \mathbf{A}$ in a record $r \in \mathbf{R}$ is denoted as $r.a$.

To split a set of records $\mathbf{R_x} \subseteq \mathbf{R}$ into blocks we make use of one or more blocking keys. A *blocking key*, $k_{i,j} = \langle a_i, f_j \rangle$ is a pair consisting of an attribute $a_i \in \mathbf{A}$ and a function f_j. The function f_j takes as input an attribute value and returns another value, such as a phonetic encoding, a substring, or a reversed string. For a given blocking key $k_{i,j} = \langle a_i, f_j \rangle$, we generate a *blocking key value* (BKV) for record $r_y \in \mathbf{R_x}$ by applying function f_j to $r_y.a_i$, denoted $v_{i,j,y} = f_j(r_y.a_i)$. For example, possible functions include the first two characters ($\mathrm{F2}$), exact value (Ext) or a Soundex encoding (Sdx) [43].

To illustrate this using the example in Table 4.1, we use the following blocking keys: the first two characters of the *First Name* attribute $\langle FN, \mathrm{F2} \rangle$, the Soundex encoding of the *Surname* attribute $\langle SN, \mathrm{Sdx} \rangle$ and the exact value of the *Postcode* attribute $\langle PC, \mathrm{Ext} \rangle$. The BKV of $\langle FN, \mathrm{F2} \rangle$ applied to r_1 is 'Jo' (the first two characters of 'John'), the BKV of $\langle SN, \mathrm{Sdx} \rangle$ applied to r_1 is 'S530' (the Soundex encoding of 'Smith') and the BKV of $\langle PC, \mathrm{Ext} \rangle$ applied to r_1 is '2000'. The result of each blocking key applied to each record in Table 4.1 is shown in Table 4.2.

Record ID	First Name	$\langle FN, \mathrm{F2} \rangle$	Surname	$\langle SN, \mathrm{Sdx} \rangle$	Postcode	$\langle PC, \mathrm{Ext} \rangle$
r_1	John	'Jo'	Smith	'S530'	2000	'2000'
r_2	Johnathon	'Jo'	Smith	'S530'	2009	'2009'
r_3	Joey	'Jo'	Schmidt	'S530'	2009	'2009'
r_4	Joe	'Jo'	Miller	'M460'	2902	'2902'
r_5	Joseph	'Jo'	Milne	'M450'	2902	'2902'
r_6	Paul	'Pa'			3000	'3000'
r_7	Peter	'Pe'	Jones	'J520'	3000	'3000'

Table 4.2: The output of sample blocking functions applied to the data set in Table 4.1.

To split a set of records $\mathbf{R_x}$ into blocks we use a blocking key $k_{i,j}$ to generate a BKV $v_{i,j,y}$ for each $r_y \in \mathbf{R_x}$ and we create one block for each unique BKV generated. We insert each record into the block corresponding to its BKV. This means two records r_y, $r_z \in \mathbf{R_x}$ will be

inserted into the same block if and only if they generate the same BKV using blocking key $k_{i,j}$, i.e. $f_j(r_y.a_i) = f_j(r_z.a_i)$. During our approaches we also need to merge blocks. This results in a single block being associated with multiple BKVs. We denote the set of BKVs associated with block $\mathbf{b}_i$ as $V(\mathbf{b}_i)$.

Based on a pre-defined list of blocking keys $\mathbf{K} = \langle k_{i,j}, k_{l,m}, \ldots \rangle$, we aim to adaptively split $\mathbf{R}$ into a set of blocks $\mathbf{B}$ by using the BKVs generated by one or more blocking keys in $\mathbf{K}$. However, we also need to control the size of the blocks we produce. The *size* of a block $\mathbf{b}$, denoted as $|\mathbf{b}|$, is the number of records in the block. To control the size of blocks, we use two size parameters: s_{min} and s_{max} with $s_{min} \leq s_{max}$, to specify the minimum and maximum block sizes that are permitted, respectively.

Problem statement. Given a data set $\mathbf{R}$, two size parameters s_{min} and s_{max}, and a list of blocking keys $\mathbf{K} = \langle k_{i,j}, k_{l,m}, \ldots \rangle$, the problem we study is to use $\mathbf{K}$ to partition the records in $\mathbf{R}$ into a set $\mathbf{B}$ of non-overlapping blocks such that for each $\mathbf{b} \in \mathbf{B}$, $s_{min} \leq |\mathbf{b}| \leq s_{max}$, and within each block the number of true matches is maximised and the number of true non-matches is minimised.

In practice, s_{min} and s_{max} can be set in accordance with operational requirements such as computational limitations for the entity resolution step. As is common with other blocking techniques, we can also improve the robustness of our approaches by running them multiple times with different lists of blocking keys for a single entity resolution task [22]. This reduces the impact that a single typographical error or incorrect value has on the entity resolution process [23]. In future work we intend to further investigate the impact of different blocking keys and whether the optimal list of keys can be discovered automatically.

We propose two recursive clustering approaches for generating blocks within a specified size range. The idea behind our approaches was illustrated in Figure 4.2. We iteratively split and merge blocks until the size parameters s_{min} and s_{max} are satisfied. The first approach processes blocks in order of decreasing block similarity (i.e., similarity-based), and the second approach in order of increasing block size (i.e., size-based).

4.2.2 Similarity Functions

During clustering we aim to merge small blocks, with the aim of bringing true matches together into the same block. We merge blocks based on their associated BKV(s). However, this merging step requires a way of measuring the similarity between two BKVs. It also requires that the functions in each blocking key which generate the BKVs, preserve some degree of attribute

similarity in the BKVs they produce, i.e. similar attribute values should result in similar BKVs. In addition, once blocks are merged, each block can be associated with multiple BKVs as shown in Figure 4.2, so we also require a way of combining the pairwise similarities between BKVs into an overall block similarity measure.

To calculate the similarity between two BKVs v_1 and v_2, denoted as $\zeta(v_1, v_2)$ we use traditional string comparison functions such as Jaro-Winkler or Jaccard similarity [23]. To combine the pairwise similarity between BKVs into an overall *block similarity* measure, denoted as $\varsigma(\mathbf{b}_1, \mathbf{b}_2)$, we make use of three traditional approaches for calculating the distance between clusters [190]: (1) *single link* $\varsigma(\mathbf{b}_1, \mathbf{b}_2) = \max(T)$, (2) *average link* $\varsigma(\mathbf{b}_1, \mathbf{b}_2) = \text{mean}(T)$ and (3) *complete link* $\varsigma(\mathbf{b}_1, \mathbf{b}_2) = \min(T)$, where $T = \{\zeta(v_1, v_2) : v_1 \in V(\mathbf{b}_1) \ \wedge \ v_2 \in V(\mathbf{b}_2)\}$.

However, traditional string comparison techniques do not always give good results for blocking keys using functions such as Soundex encodings or the first two characters of an attribute. For example, neither of the above similarity functions give a good indication of the similarity between the Soundex codes 'S530' and 'S550.' In order to obtain a better similarity measure, we use the original unencoded attribute values and apply a traditional string comparison function on them instead. For example, to calculate the similarity between BKVs 'S530' and 'S550', we take the attribute values that encode to 'S530' (such as 'Smith' and 'Smythe') and compute their similarity with attribute values that encode to 'S550' (such as 'Simon' and 'Simeon'), using a traditional string comparison metric. If possible, we calculate all pairwise combinations of such values in the data set(s) to get a weighted average similarity between pairs of Soundex codes.

However, if the full pairwise calculation is computationally infeasible, we randomly sample a selection of original values for each code and take the average similarity between these. In practice we found that even small sample sizes still produced results that were nearly identical to those of the complete calculation. We discuss this further in Section 4.4.

4.2.3 Similarity-Based Blocking Approach

The similarity-based blocking approach is described in Algorithm 4.1. To begin, we set $n = 1$ and take the set of records as $\mathbf{R}$. We generate a set $\mathbf{B}$ of blocks using the n^{th} blocking key in $\mathbf{K}$ (line 1). One block is created for each unique BKV. Next, $\mathbf{B}$ is partitioned into three disjoint sets $\mathbf{B}^-$, $\mathbf{B}^*$ and $\mathbf{B}^+$, with $\mathbf{b}_i \in \mathbf{B}^-$ if $|\mathbf{b}_i| < s_{min}$, $\mathbf{b}_i \in \mathbf{B}^*$ if $s_{min} \leq |\mathbf{b}_i| \leq s_{max}$ and $\mathbf{b}_i \in \mathbf{B}^+$ if $|\mathbf{b}_i| > s_{max}$ (line 2). We place each pair of blocks in $\mathbf{B}^- \cup \mathbf{B}^*$ into a priority queue Q, in order of their decreasing block similarity (lines 4-7). We retrieve from Q the pair of blocks ($\mathbf{b}_i$, $\mathbf{b}_j$) with maximum $\varsigma(\mathbf{b}_i, \mathbf{b}_j)$ (line 9). We merge $\mathbf{b}_i$ and $\mathbf{b}_j$ into $\mathbf{b}_{ij}$ where $V(\mathbf{b}_{ij}) = V(\mathbf{b}_i) \cup V(\mathbf{b}_j)$. We then

Algorithm 4.1: SimilarityBasedClustering(R, *K*, ς, n, s_{min}, s_{max})

Input:
- Set of records: **R**
- List of blocking keys: **K**
- Block similarity function: ς
- Current recursion depth: n // Set as $n = 1$ for first call to algorithm
- Minimum block size: s_{min}
- Maximum block size: s_{max}

Output:
- Set of correct sized blocks: $\mathbf{B}^*$

```
1:  B = GenerateBlocks(R, K[n])                    // Generate blocks using the n^th blocking key in K
2:  B^-, B*, B^+ = SizePartition(B, s_min, s_max)  // Partition B into too small, correct size, too large blocks
3:  Q = GeneratePriorityQueue()                    // Create empty queue, ordered by similarity
4:  for b_i in B^- ∪ B* do:
5:    for b_j in B^- ∪ B* \ b_i do:
6:      if |b_i| + |b_j| ≤ s_max then:
7:        Q.Insert(ς(b_i, b_j), b_i, b_j)        // Insert correct sized pairs into the queue
8:  while Q ≠ ∅ do:
9:    sim, b_i, b_j = Q.Pop()                      // Get the first pair from the queue
10:   b_ij = MergeBlocks(b_i, b_j)
11:   for b_k in B^- ∪ B*:
12:     if |b_ij| + |b_k| < s_max then:
13:       Q.Insert(ς(b_ij, b_k), b_ij, b_k)      // Put back in queue with new block similarity
14:   if |b_ij| < s_max then:
15:     B* = B* ∪ b_ij                           // Add to correct size blocks
16: for b_i in B^+ do:                             // Process the too large blocks
17:   B_i = SimilarityBasedClustering(b_i, K, ς, n + 1, s_min, s_max)   // Call recursively with n + 1
18:   B* = B* ∪ B_i
19: return B*
```

calculate $\varsigma(\mathbf{b}_{ij}, \mathbf{b}_k)$ for all $\mathbf{b}_k$ s.t. $|\mathbf{b}_k| + |\mathbf{b}_{ij}| \leq s_{max}$ and reinsert these new pairs of blocks into Q (line 13). We then proceed with the pair of blocks with the second highest block similarity (loop back to line 9), and continue this process until no more merges are possible. For each $\mathbf{b}_i \in \mathbf{B}^+$ (i.e. blocks that are too large, $|\mathbf{b}_i| > s_{max}$) we call the algorithm recursively with $\mathbf{b}_i$ as the new set of records and using the next blocking key in **K** to generate new BKVs (lines 16-18).

Figure 4.2 illustrates this process applied to the example data set in Table 4.1 with $\mathbf{K} = \langle\langle FN, \mathrm{F2}\rangle, \langle SN, \mathrm{Sdx}\rangle\rangle$ and $s_{min} = 2$ and $s_{max} = 3$. We start by splitting the records into blocks using the first blocking key $\langle FN, \mathrm{F2}\rangle$ (the first two characters of *FirstName*). The blocks that have a size smaller than two (s_{min}) are clustered and merged. Any blocks with size greater than three (s_{max}) are split using the second blocking key $\langle SN, \mathrm{Sdx}\rangle$ (the Soundex encoding of *Surname*). Then, in a second merging phase, blocks that are smaller than size two (s_{min}) are again clustered. In this case this finishes the algorithm since all blocks are now in the correct size range. However, if there were still blocks with size greater than three they would be split using a third blocking key, for example $\langle PC, \mathrm{Ext}\rangle$, any resulting small blocks would again be clustered and merged, and so forth. This continues until no blocks remain with size greater than three or we run out of blocking keys in **K**.

The main drawback of the similarity-based approach is the need to calculate $\varsigma(\mathbf{b}_i, \mathbf{b}_j)$ for

Algorithm 4.2: SizeBasedClustering(R, K, ς, n, s_{min}, s_{max})

Input:
- Set of records: **R**
- List of blocking keys: **K**
- Block similarity function: ς
- Current recursion depth: n // Set as $n = 1$ for first call to algorithm
- Minimum block size: s_{min}
- Maximum block size: s_{max}

Output:
- Set of correct sized blocks: $\mathbf{B}^*$

1: $\mathbf{B}$ = GenerateBlocks(**R**, **K**$[n]$) // Generate blocks using the n^{th} blocking key in **K**
2: $\mathbf{B}^-$, $\mathbf{B}^*$, $\mathbf{B}^+$ = SizePartition(**B**, s_{min}, s_{max}) // Partition **B** into too small, correct size, too large blocks
3: Q = GeneratePriorityQueue() // Create empty queue, ordered by block size
4: `for` $\mathbf{b}_i$ `in` $\mathbf{B}^-$ `do`:
5: Q.Insert($\mathbf{b}_i$) // Put the small blocks into the queue
6: `while` $Q \neq \emptyset$ `do`:
7: $\mathbf{b}_i = Q$.Pop() // Get the first block from the queue
8: $\mathbf{b}_j = Argmax(\varsigma(\mathbf{b}_i, \mathbf{b}_k)), \forall \mathbf{b}_k \in \mathbf{B}^- \cup \mathbf{B}^*$ `such that` $|\mathbf{b}_i| + |\mathbf{b}_k| \leq s_{max}$ // Get nearest block of correct size
9: $\mathbf{b}_{ij}$ = MergeBlocks($\mathbf{b}_i$, $\mathbf{b}_j$)
10: `if` $|\mathbf{b}_{ij}| < s_{min}$ `then`:
11: Q.Insert($\mathbf{b}_{ij}$) // Put new block back into the queue
12: `else`:
13: $\mathbf{B}^* = \mathbf{B}^* \cup \mathbf{b}_{ij}$ // Add to correct size blocks
14: `for` $\mathbf{b}_i$ `in` $\mathbf{B}^+$ `do`: // Process the too large blocks
15: $\mathbf{B}_i$ = SizeBasedClustering($\mathbf{b}_i$, **K**, ς, $n+1$, s_{min}, s_{max}) // Call recursively with $n+1$
16: $\mathbf{B}^* = \mathbf{B}^* \cup \mathbf{B}_i$
17: `return` $\mathbf{B}^*$

each pair of blocks in $\mathbf{B}^- \cup \mathbf{B}^*$ and store them in Q. In addition, as blocks are merged, the block similarity needs to be calculated between the new block and all remaining blocks. This reduces the scalability of the approach and also leads to high memory overhead since Q can become large, $O(|\mathbf{B}|^2)$. Next we present an alternative approach with better scalability that removes the need to store all pairwise combinations of blocks in memory.

4.2.4 Size-Based Blocking Approach

The size-based blocking approach is described in Algorithm 4.2. The initial setup for this approach is identical to that of the similarity-based blocking approach. However, in the size-based case the priority queue Q contains individual blocks, which are ordered by increasing block size (line 5). This is an important distinction since it significantly reduces the size of Q from $O(|\mathbf{B}|^2)$ to $O(|\mathbf{B}|)$. In the main loop of the algorithm (lines 6-13) we remove the smallest block $\mathbf{b}_i$ from Q (line 7), determine the block $\mathbf{b}_j$ such that $|\mathbf{b}_i| + |\mathbf{b}_j| \leq s_{max}$ and $\varsigma_1(\mathbf{b}_i, \mathbf{b}_j)$ is maximised (line 8). Essentially we find the most similar block to $\mathbf{b}_i$ such that their combined size would be less than s_{max}. We merge $\mathbf{b}_i$ and $\mathbf{b}_j$ into $\mathbf{b}_{ij}$ (line 9) and if $|\mathbf{b}_{ij}| \leq s_{min}$, we reinsert $\mathbf{b}_{ij}$ into Q. We then proceed to the next smallest block (loop back to line 6) and continue this process until no blocks remain with size less than s_{min}. As with the similarity-based approach, for each block in $\mathbf{B}^+$ the algorithm is called recursively with $n = n + 1$ and using the next blocking key in **K**.

4.3 Penalty Function

The basic idea behind our blocking approaches is to split and merge blocks using a sequence of blocking keys until all blocks are in the desired size range. While this is effective in most situations, there are circumstances where a greater level of control in the blocking approach is required.

Consider that for English names, both 'John' and 'Jonathon' are common. However, because 'John' is an abbreviation or nickname of 'Jonathon', we would likely prefer to compare the 'John Smiths' and the 'Jonathon Smiths', than the 'John Smiths' and the 'John Millers'. However, in order for this to happen, the total number of records with either 'John' or 'Jonathon' as a first name value must be less than s_{max}, since otherwise the corresponding blocks will not be merged. Since they are both common names, this may not be the case and true matches may be missed in the blocking process (since the 'Johns' and 'Jonathons' will end up in different blocks).

In order to prevent such situations, we now present a penalty function that combines block size and block similarity to determine whether or not to merge two blocks. This allows blocks to be merged even if their combined size is greater than s_{min}, provided the similarity (as measured by ς) is high enough.

The penalty function $\mathbf{\Phi}$ is as follows:

$$\mathbf{\Phi}\left(\mathbf{b}_i, \mathbf{b}_j\right) = 1 - \alpha^{-\left(\frac{|\mathbf{b}_i|+|\mathbf{b}_j|}{s_{scale}} - \beta\right)} \text{ for } \alpha \geq 1 \text{ and } \beta \in \mathbb{R}.$$

Two blocks $\mathbf{b}_i$ and $\mathbf{b}_j$ will be merged if they satisfy the inequality $\varsigma(\mathbf{b}_i, \mathbf{b}_j) \geq \mathbf{\Phi}(\mathbf{b}_i, \mathbf{b}_j)$. As the combined block size gets larger, the similarity threshold required for merging also increases, and vice versa.

The penalty function involves two parameters, α and β, which together with s_{scale} (related to s_{min} and s_{max}), produce the desired merging behaviour. We describe the purpose of each parameter here:

- α determines the trade-off between similarity and size. Higher values of α produce a stricter similarity threshold as the block size increases. In the limit as $\alpha \to \infty$, $\mathbf{\Phi}$ becomes a hard size restriction. In this case block similarity does not affect the merging decisions.
- β constrains the clustering by enforcing either a minimum block size ($\beta > 0$) or a minimum similarity threshold ($\beta < 0$). If $\beta = 0$ then there are no size or similarity restrictions on the merging. We note that β can only create one of these two types of restrictions for a given

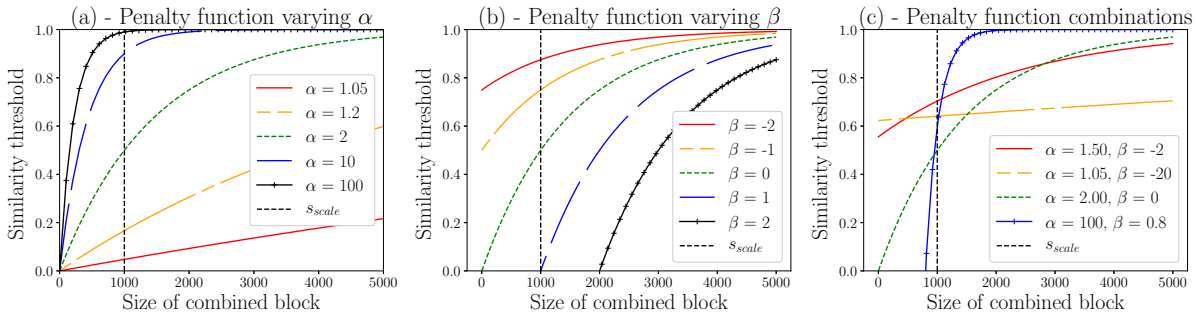

Figure 4.3: Penalty function example configurations.

clustering problem, since there may be no solution if both a minimum block size and a minimum similarity threshold are specified.

- s_{scale} is a scaling parameter that is useful for computational reasons. In practice, including s_{scale} removes the need to repeatedly calculate extremely large exponents of numbers very close to 1 when computing $\mathbf{\Phi}(\mathbf{b}_i, \mathbf{b}_j)$. Eliminating s_{scale} by changing α and β gives a mathematically identical function, but with α extremely close to 1 in practice. Including s_{scale} improves computational performance and prevents machine precision from influencing results. We explain how to set s_{scale} below.

We next provide the idea behind the penalty function with reference to the examples in Figure 4.3. In each example s_{scale} is set to 1,000 (the vertical dashed line). Consider the case where $\alpha = 2$ and $\beta = 0$ represented by the (green) curved dashed line from $(0, 0)$ to the top right corner in each example. Before merging two blocks $\mathbf{b}_i$ and $\mathbf{b}_j$ where $|\mathbf{b}_i| + |\mathbf{b}_j| = s_{scale}$ (i.e. $|\mathbf{b}_i| + |\mathbf{b}_j| = 1{,}000$), the similarity between the blocks must be at least $1 - \frac{1}{2^1} = 0.5$. Before merging two blocks with a combined size of $2 * s_{scale}$, the similarity must be at least $1 - \frac{1}{2^2} = 0.75$. A size of $3 * s_{scale}$ requires similarity greater than 0.875, and so on.

The value of α determines the rate at which the required similarity approaches 1.0, with higher values approaching more quickly than lower values as shown in Figure 4.3 (a). Changing the value of β has the effect of moving the curve to the left or right as shown in Figure 4.3 (b). For example, $\beta = -1$ and $\alpha = 2$ set a minimum similarity for merging to be $1 - \frac{1}{2^1} = 0.5$. If $\beta = 1$ and $\alpha = 2$, then blocks will be merged regardless of similarity until the combined size is at least 1,000 (equal to s_{scale}). By combining different values of α and β we can obtain a wide variety of merging conditions as shown in Figure 4.3 (c).

We now explain how best to choose the values of α, β and s_{scale} in order to achieve the desired

merging behaviour. In applications where minimum block size is not a hard requirement, the default we use on a data set is $s_{scale} = 0.5 * s_{max}$, $\alpha = 2$ and $\beta = 0$. This sets a similarity threshold of 0.75 to merge blocks with combined size greater than s_{max} and prevents blocks with very low similarity from being merged regardless of size. If minimum block size is important, then the default parameters we use are $s_{scale} = s_{min}$, $\alpha = (2 * s_{max})/(s_{max} - s_{min})$ and $\beta = 1$. This causes blocks to be merged regardless of similarity up to a combined size of s_{min}, and sets a similarity threshold of 0.75 to merge blocks with a combined size larger than s_{max}. In both cases, with some knowledge of the data, the value of α can be scaled to increase or decrease the similarity threshold of 0.75 as desired.

To incorporate the penalty function, Algorithms 4.1 and 4.2 have to be slightly modified. In Algorithm 4.1 we replace the size restrictions on $\mathbf{b}_i$ and $\mathbf{b}_j$ in lines 4 - 6 with the penalty function condition, and the same for $\mathbf{b}_k$ and $\mathbf{b}_{ij}$ in lines 11 and 12. In Algorithm 4.2 all blocks are inserted into Q in line 5, not just blocks with size less than s_{min}. Similarly $\mathbf{b}_{ij}$ is always reinserted into Q in line 11, regardless of size. Additionally, in line 8, we replace the size restriction on $\mathbf{b}_k$ with the penalty function condition on $\mathbf{b}_i$ and $\mathbf{b}_k$.

4.4 Evaluation

To evaluate our approaches we compared performance with standard blocking [47], Soundex encoding [43], and sorted neighbourhood based indexing [73], on each of the three data sets detailed in Section 2.5. For evaluation measures we used pairs completeness and reduction ratio [23] and a combination of the two measures similar to F-Measure: Pairs Completeness (PC) = $\frac{s_M}{n_M}$, Reduction Ratio (RR) = $1 - \frac{s_M + s_N}{n_M + n_N}$ and the combined F-Measure (FM) = $\frac{2 * PC * RR}{PC + RR}$, where n_M, n_N, s_M, s_N correspond to the total number of matched pairs, the total number of non-matched pairs, the number of true matched candidate record pairs and the number of true non-matched candidate pairs, respectively.

We do not explicitly model block quality. However, since merging blocks can only improve improve PC, we merge blocks until s_{max} is reached, regardless of block quality. If minimum block size is not a hard requirement, then block size and pairs completeness can be balanced using the penalty function, where a minimum similarity threshold will prevent blocks with a low likelihood of containing true matches from being merged, regardless of block size.

The experimental results on the Cora, UKCD, and NCVR-450 data sets are shown in Figure 4.4. For Cora we set $s_{min} = 50$, $s_{max} = 100$ and $K = \langle\langle Title, \text{Ext}\rangle, \langle Author, \text{Ext}\rangle\rangle$. For UKCD

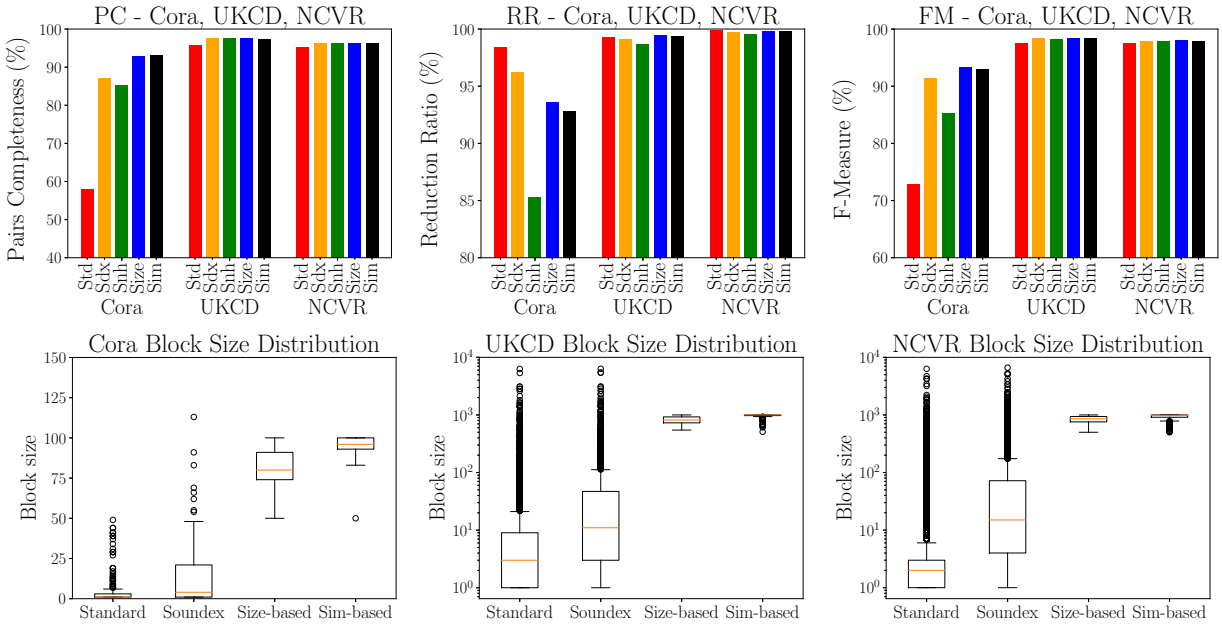

Figure 4.4: The distribution of block sizes produced by our techniques along with baseline comparisons.

we set $s_{min} = 500$, $s_{max} = 1{,}000$ and $K = \langle\langle \textit{Surname}, \text{Ext}\rangle, \langle \textit{First Name}, \text{Ext}\rangle, \langle \textit{Birth Parish}, \text{Ext}\rangle\rangle$. For NCVR-450 we set $s_{min} = 500$, $s_{max} = 1{,}000$ and $K = \langle\langle \textit{Surname}, \text{F2}\rangle, \langle \textit{First Name}, \text{F2}\rangle\rangle$.

The main focus of our approach was to improve scalability while still achieving high quality blocking. On all three data sets, we achieve equal or better F-Measure values than the three baseline approaches. This indicates that our approaches achieve comparable blocking quality to other common blocking techniques. To evaluate the impact on scalability, We show the distribution of block sizes generated by our approaches in Figure 4.4. As can be seen from the results, both our approaches produce blocks in the required size range, 500 - 1,000 records for UKCD and NCVR-450, and 50 - 100 records for Cora. While the size-based approach tends to distribute the block sizes throughout the interval $[s_{min}, s_{max}]$, the similarity-based approach tends to generate the majority of blocks with size close to s_{max}. This means it creates fewer blocks overall and makes it appropriate for parallel collective entity resolution applications since the time taken to process each block will likely be about the same.

We tested different parameter settings for our approaches to examine how sensitive they are to changing s_{min}, s_{max}, and the block similarity measure ς, and the results are shown in Table 4.3 (at the end of the chapter). In most cases, the choice of block similarity measure ς has minimal

effect on the results. However, the complete link block similarity measure did not work well with the size-based approach, particularly on the Cora data set. Changing s_{min} and s_{max} affects the trade-off between PC and RR as expected.

We tested the penalty function and the results are shown in Figure 4.5. For Cora we set $s_{scale} = 50$ and $s_{max} = 100$, and for UKCD and NCVR-450 we set $s_{scale} = 500$ and $s_{max} = 1,000$. When $\beta = 0$ (no minimum block size or minimum similarity threshold), the penalty function generally achieves the best combination of PC and RR values, the exception being for low values of α where the similarity threshold is very low, even for large blocks which results in poor RR values. High values of α and negative values of β mean the similarity threshold to perform any merging is high. This essentially negates the clustering steps of the algorithms, which results in poor PC values for data sets with lower data quality. High values of α in combination with positive values of β produce generally balanced blocks. We note that for the UKCD data set, setting $\alpha = 1.1$ performs very poorly. It repeatedly merges many blocks in each iteration of the algorithm and either runs out of blocking keys (resulting in poor RR values), or has to use attributes that have poor data quality (resulting in poor PC values). For the NCVR-450 data set, the penalty function produces very similar results regardless of the settings for α and β. The merging of blocks has less impact on the NCVR-450 data set, since it is relatively clean so merges do not increase PC values substantially, and also large enough that it requires many merges to reduce RR values significantly.

We also tested the scalability of our approaches using subsets of different sizes of the entire NCVR data set. We set $s_{min} = 500$, $s_{max} = 1,000$ and $K = \langle\langle Surname, \mathrm{F2}\rangle, \langle First\ Name, \mathrm{F2}\rangle\rangle$ and the results are shown in Figure 4.6. As can be seen, the scalability is nearly linear in practice. We discuss scalability and computational complexity further in Section 4.5.

We compared the total number of candidate pairs generated as well as the largest block generated by the different approaches and the results are shown in Figure 4.6. Controlling the maximum block size ensures that the total number of candidate pairs increases linearly with the size of the data set which means that once the data set becomes large, our techniques generate fewer candidate pairs than the traditional and Soundex based approaches. As a result, even though our approaches increase the time required for blocking compared to the baseline approaches, in general this will be more than made up for by a reduction in the time required to perform the comparison step. We show an example of this in Figure 4.6 (d) assuming 10,000 comparisons per second (approximately what we achieved in experiments in Chapter 7). The total running time is clearly dominated by the comparison step rather than the blocking step. As

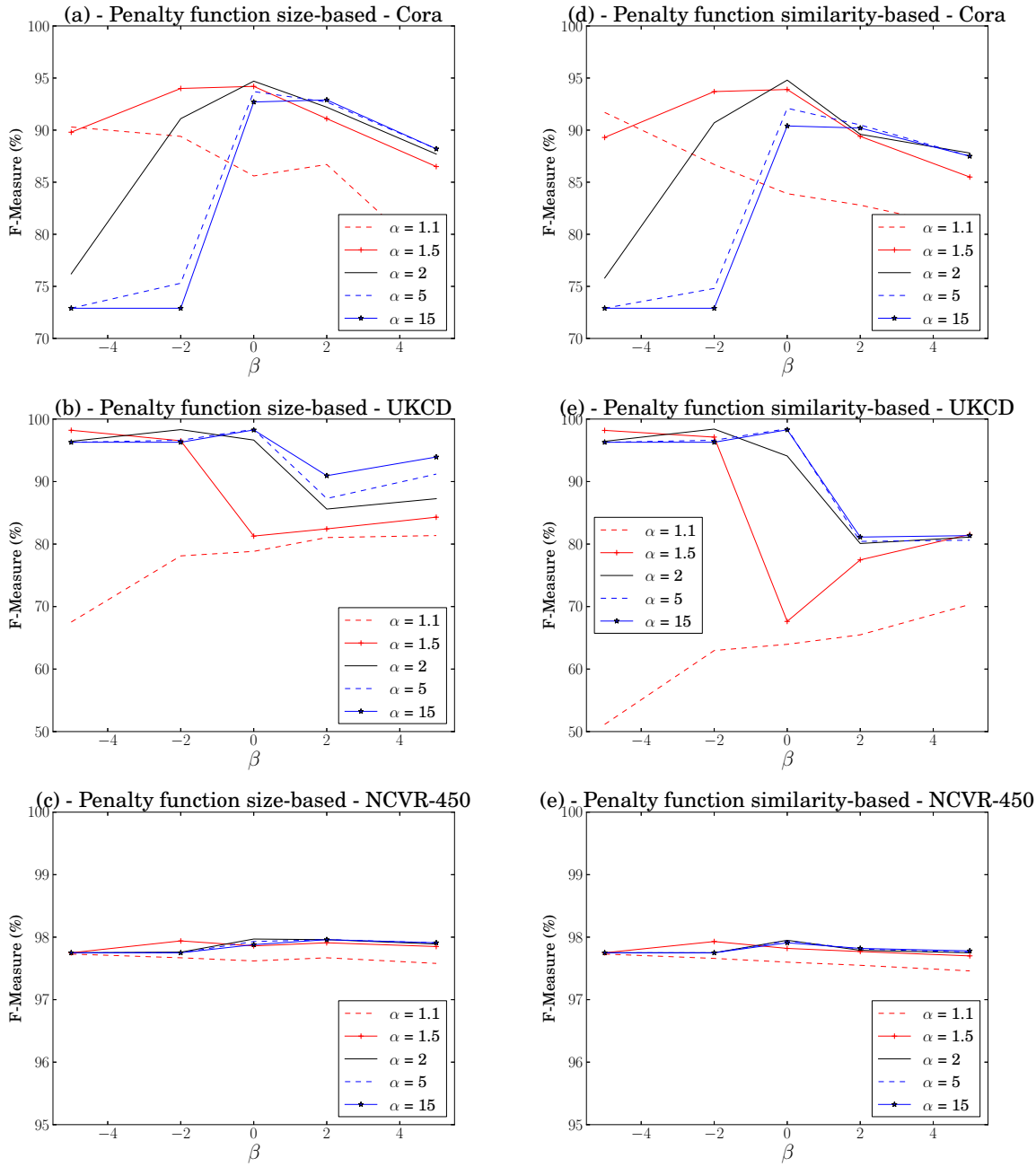

Figure 4.5: Penalty function results for Cora, UKCD, and NCVR-450. For each data set we display how different combinations of α and β affect the F-Measure values.

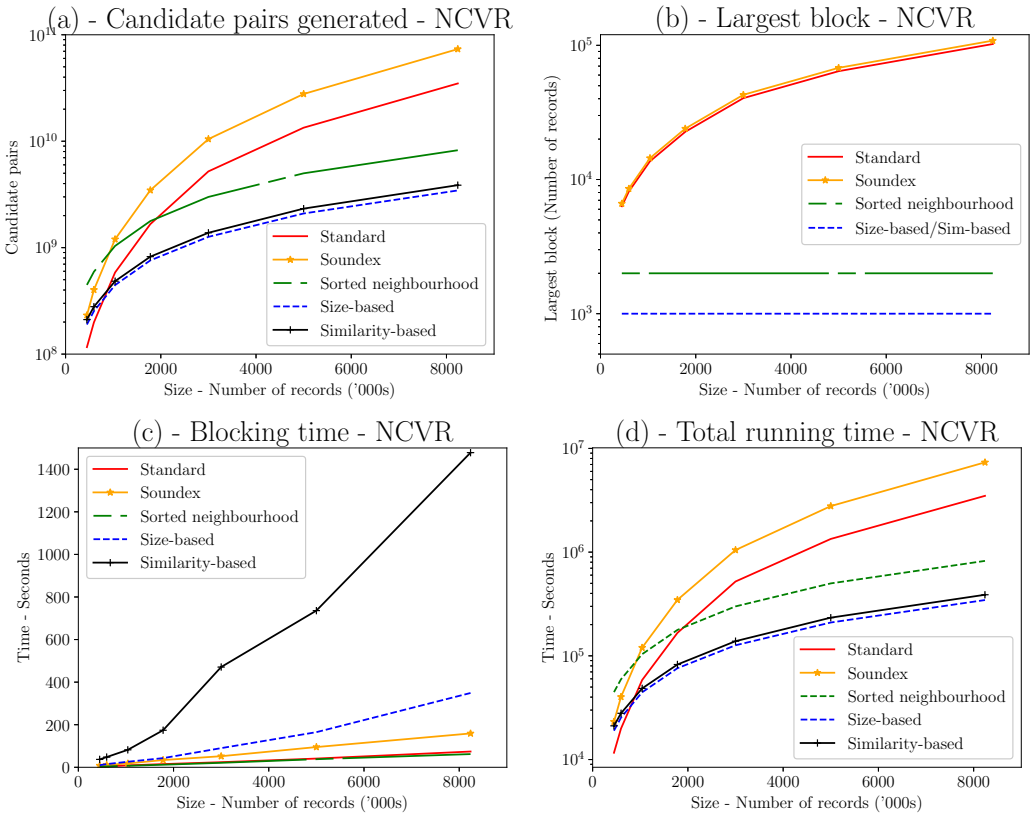

Figure 4.6: Scalability tests using different proportions of the NCVR data set.

a result, even though our techniques require additional time in the blocking step, this is more than made up for by the time saved in the comparison step.

In addition, the worst case block size is controlled by our approaches. Since many advanced entity resolution techniques have scalability worse than linear (i.e. worse than $O(n)$), the running time is heavily dependent on the size of the largest block. As such, the time saving will be even greater than that indicated by the reduction in the number of candidate pairs and shown in Figure 4.6 (d). In addition to the applications for advanced entity resolution techniques, controlling the worst-case block size means that our techniques are suitable for real-time entity resolution where operational requirements limit the number of comparisons that can be performed [145] and privacy preserving record linkage where an uneven block size distribution can lead to patterns that can be exploited by a potential attacker, even when data is encoded [33].

Finally, as we discussed in Section 4.2.2, for blocking key functions such as Soundex encodings or the first two characters of an attribute value, we calculate the similarity of the resulting BKVs by considering the original attributes values. Where possible, we calculated the weighted pair-wise similarity of all values of the attribute, however where this was computationally infeasible, we sampled from the attribute values instead. To determine whether this sampling process had a significant impact on the results, we investigated the impact of the sample size in the similarity calculations, using the NCVR-450 data set and Soundex encodings. Even with a sample size of 1, the clustering still produced similar results to the complete calculation and the reduction in F-Measure was less than 0.1% in all cases. As a result, we conclude that the sample size does not significantly affect the performance.

4.5 Discussion

We now discuss the characteristics of the two approaches and present their computational complexities. Depending on the settings of s_{min} and s_{max}, it is possible that our approaches may generate some blocks that are outside the desired size range. For example, if $s_{min} = 0.8 * s_{max}$, some blocks may have a size in the range $0.5 * s_{max}$ to $0.8 * s_{max}$. Merging any two of these blocks would result in a block size greater than s_{max}, so none of them end up being merged. However, if s_{min} and s_{max} satisfy $s_{max} \geq 2 * s_{min}$ then we are guaranteed that at most one block at the end will be smaller than s_{min} because if two blocks were left, they could be merged as their combined size would still be below s_{max}.

If blocks are left at the end of either algorithm which are larger than s_{max}, then there must

exist some unique combination of BKVs that occurs more frequently than s_{max} and our only option is to add another blocking key to **K**.

The similarity-based blocking approach ensures that pairs of blocks with high block similarity are merged together. In practice, the approach often creates many blocks that are close in size to s_{max} which makes it effective for load balancing in parellel ER applications [88]. However, if there is a block left at the end which is too small, it may be quite small in comparison to s_{min}, which may make this approach less suitable in applications such as privacy preserving record linkage [177] where s_{min} is particularly important. The running time of the similarity-based approach is also typically longer than that of the size-based approach.

In practice, if $s_{max} \geq 2 * s_{min}$, the size-based approach tends to produce blocks that are more evenly distributed within the size range, with potentially a single block that is too small. Since the merging is done iteratively from smallest to largest, if there is a block that is smaller than s_{min}, its size is typically close to s_{min}, although this closeness is not mathematically guaranteed. This means that for situations where minimum block size is important the size-based approach is a good candidate. However, the size-based blocking approach is not as successful when there are multiple large blocks with different BKVs from values that are quite similar. For example, depending on the blocking keys used, the first names 'John' and 'Jonathon' may generate different BKVs but should likely be combined into one block. However, because blocks are processed in order of size and both blocks may be large, neither block will be considered until late in the merging process. By the time they are compared one of them may have already grown too large (due to other merges) for them to be merged. This situation can potentially be overcome by the penalty function.

The selection of the blocking keys in K is important for both approaches and has a significant effect on the running time and the blocking quality. At present we rely on domain expertise to select the blocking keys, taking into account such factors as completeness, size of the domain, distribution of values and general data quality. As part of our future work we intend to investigate methods for automatically selecting blocking keys, such as those developed by Kejriwal and Miranker [87] and Ramadan and Christen [144].

In the worst case, the time complexity is $O(|\mathbf{R}|^3 \log(|\mathbf{R}|))$ and $O(|\mathbf{R}|^3)$ for the similarity-based approach and size based approach respectively. For the similarity-based approach, Q can contain $O(|\mathbf{R}|^2)$ blocks (line 7) and during the loop (lines 8 - 15) we have to perform $O(|\mathbf{R}|)$ insertions into Q of time complexity $O(\log(|\mathbf{R}|))$ (line 13). For the size-based approach the size of Q is at most $O(|\mathbf{R}|)$ (line 5) but calculating $Argmax(\varsigma(\mathbf{b}_i, \mathbf{b}_k))$ (line 8) is potentially $O(|\mathbf{R}|^2)$ so we end

up with an overall complexity of $O(|\mathbf{R}|^3)$.

In practice the similarity-based approach is significantly slower than the size-based approach. In addition the running time of both approaches is much more dependent on the number of unique BKVs generated by the blocking keys in **K** rather than the size of **R**. This is because we create one block for each BKV during clustering so the running time of the similarity-based approach becomes $O(|\mathbf{B}|^3 log(|\mathbf{B}|))$ and the size-based approach becomes $O(|\mathbf{B}|^3)$. In the worst case, each record generates a unique BKV and we end up with the asymptotic complexity above. However, we note that certain functions have hard limits on the number of unique BKVs they can generate. There are at most 676 (i.e. 26^2) combinations of two letters, so selecting the first two letters of an attribute value will create at most 676 blocks to consider in the clustering stage. Similarly, phonetic encodings such as Soundex and Double Metaphone [43], also have hard limits on the maximum number of unique BKVs they can create. Finally, some optimisation techniques such as pre-calculating and caching similarity values can be performed to improve the efficiency of both techniques.

4.6 Summary

In practice, poor scalability is a limitation of all entity resolution techniques and considerable research has been done in order to address this problem. Because the scalability of many advanced entity resolution techniques is even worse than traditional entity resolution, when using a blocking based technique, it is important to limit not only the average block size, but also the worst case block size. In addition, because of the graph based nature of many advanced enitity resolution techniques, it is important that any blocking technique applied produce disjoint blocks.

In order to address these problems we have developed two recursive clustering approaches which can generate blocks for entity resolution in a given size range. The blocks are also disjoint, meaning they can be evaluated independently, or in parallel, with only limited communication between blocks. Even in the case of Markov logic networks, where the scalability is worse than NP-Hard, they can perform in practice on blocks of a limited size in a few seconds. By setting an appropriate maximum (and minimum) block size, we can make sure that operational requirements are satisfied in terms of the running time of the entity resolution process. Our approaches also limit the size of the largest block, which means that we do not end up in the situation where a single block significantly worsens the efficiency.

We also proposed a penalty function which allows us to control the trade-off between block size and block quality, and fine tune either approach. This allows us to merge blocks beyond the maximum block size, and then split them on a second blocking key. This approach can be important for variations of a common name or attribute value, (a firstname of 'Jon' and 'John' for example) but can be tailored for other situations and requirements as well.

Finally, we have evaluated our approaches on three data sets. Our experimental results show that both our techniques perform well in comparison to the baseline approaches and give similar values of pairs completeness and reduction ratio while still satisfying our block size requirements. While there is a trade-off in terms of increased processing time on the blocking step, for most entity resolution techniques this will be more than made up for in the comparison and matching steps due to the reduced number of comparisons and in particular, the reduced size of the largest block.

4.6.1 Future Work

There are two main directions we intend to investigate in the future. At the moment we rely on domain expertise to select the blocking functions and blocking attributes. While this is common for most blocking techniques, poor choices in the blocking stage can lead to unusable entity resolution results (this is something we discuss in further detail in Chapter 6). Kejriwal and Miranker [87] and Ramadan and Christen [144] presented approaches for automatic selection of blocking keys for entity resolution and a similar approach would have value for our work as well.

Secondly, our work is likely of value for real-time entity resolution problems [12, 27, 145, 173], where only a fixed number of comparisons can be performed before a response needs to be provided. Because we can control both the minimum and maximum block size, we can ensure these operational requirements are met. However, this requires solving two additional problems: what to do with blocking key values we haven't seen before and how to update our blocks as new information arrives. We intend to investigate whether a balanced tree data structure would allow us to maintain both our size and our disjoint properties while being fast enough to update and query to be usable for real-time situations.

Cora										
s_{min} - s_{max}		20 - 50			20 - 100			50 - 100		
Block similarity measure ς		Single	Average	Complete	Single	Average	Complete	Single	Average	Complete
Size-based	PC	83.45	84.19	80.90	**92.95**	92.24	81.95	**92.95**	91.55	85.90
	RR	96.86	**97.03**	97.01	96.20	96.35	96.50	93.64	93.61	93.63
	FM	89.66	90.16	88.23	**94.55**	94.25	88.63	93.29	92.57	89.60
Similarity-based	PC	87.77	88.27	85.52	92.95	92.95	92.95	**93.07**	92.97	92.95
	RR	96.51	96.61	**96.71**	95.64	96.09	96.14	92.80	93.28	93.55
	FM	91.93	92.25	90.77	94.28	94.49	**94.52**	92.93	93.12	93.25
UKCD										
s_{min} - s_{max}		50 - 100			100 - 200			500 - 1,000		
Block similarity measure ς		Single	Average	Complete	Single	Average	Complete	Single	Average	Complete
Size-based	PC	89.64	88.72	87.49	93.65	93.32	91.57	**97.44**	96.77	95.92
	RR	**99.95**	**99.95**	**99.95**	99.89	99.89	99.90	99.47	99.48	99.48
	FM	94.51	94.00	93.31	96.67	96.49	95.55	**98.44**	98.11	97.67
Similarity-based	PC	90.43	90.24	89.33	93.76	93.82	93.18	97.32	97.27	**97.42**
	RR	99.94	99.94	**99.95**	99.88	99.89	99.89	99.38	99.44	99.45
	FM	94.95	94.84	94.34	96.72	96.76	96.42	98.34	98.34	**98.42**
NCVR-450										
s_{min} - s_{max}		500 - 1,000			2,500 - 5,000			5,000 - 10,000		
Block similarity measure ς		Single	Average	Complete	Single	Average	Complete	Single	Average	Complete
Size-based	PC	96.17	96.25	96.15	96.49	96.53	96.48	96.63	**96.64**	96.63
	RR	99.81	**99.82**	**99.82**	99.07	99.08	99.09	98.19	98.16	98.19
	FM	97.96	**98.00**	97.95	97.76	97.79	97.77	97.40	97.39	97.40
Similarity-based	PC	96.17	96.35	96.32	96.50	96.57	96.55	96.67	**96.68**	96.66
	RR	99.79	99.80	**99.81**	98.96	99.01	99.07	98.00	98.03	98.05
	FM	97.95	**98.04**	98.03	97.71	97.77	97.79	97.33	97.35	97.35

Table 4.3: Effects of parameter settings on PC, RR and F-Measure for Cora, UKCD, and NCVR-450, showing different configurations of s_{min}, s_{max} and the three different block similarity measures (ς) single link, average link, and complete link. The best value(s) in each row is shown in bold.

Chapter 5

Generating Training and Bootstrapping Data

Many advanced entity resolution techniques require bootstrapping or training data. Collective entity resolution techniques such as those of Bhattacharya and Getoor [12] and Kalashnikov and Mehrotra [86] start with an initial set of matches and then propogate the results of these and subsequent match decisions throughout the entity relationship graph in order to make subsequent match decisions. Markov logic networks [165] are a supervised classification technique and rely on training data to inform the logical formulae and their weights. Many group linkage techniques are fully supervised [52, 54] and temporal entity resolution techniques require training data to set the weights for agreement and disagreement decay [28, 106]. In academic environments, data sets can be chosen where ground truth data (and thus training data) is available, thus allowing such techniques to be developed and evaluated. However, for many real-world entity resolution problems, training data is not available, making many of these techniques difficult or impossible to apply. In this chapter we propose a technique that uses ambiguity to generate bootstrapping and training data in situations where one-to-one and one-to-many matching constraints exist. This allows supervised advanced entity resolution techniques to be applied, as well as providing a better performing bootstrapping approach for collective entity resolution techniques.

The remainder of this chapter is structured as follows: in Section 5.1 we describe the bootstrapping problem in more detail. We discuss why it is particularly important for collective entity resolution techniques and also how it applies to other advanced entity resolution techniques. In Section 5.2 we summarise our notation and provide mathematical definitions for our measures of ambiguity and formally present our approach. In Section 5.3 we detail how our notion of ambiguity can be applied to active learning to generate training data for other entity

resolution techniques. We perform an experimental evaluation of our techniques in Section 5.4. In Section 5.5 we discuss various aspects of our approach in more detail as well as some practical considerations, before providing our conclusions and some possible extensions of our work in Section 5.6.

5.1 Overview

Many advanced entity resolution techniques are fully supervised (i.e. they use labelled examples to train a classification model before using it to classify unseen or unknown examples). Markov logic networks [165], group linkage techniques [52, 54], and population reconstruction techniques [103, 111] all rely on having labelled training examples of matches (and often non-matches) on which to train the classification model. These techniques cannot be applied without training data.

In addition, as we discussed in Chapter 3, even though collective entity resolution approaches such as those of Bhattacharya and Getoor [12] and Kalashnikov and Mehrotra [86] are unsupervised, they still require a bootstrapping step, where an initial set of matches is used to generate the underlying graph that these techniques use for enitity resolution.

As we noted in Section 3.2, the defining features of collective entity resolution techniques are that match decisions are not made independently for individual record pairs, and relationships between entities are used in the matching process. However, incorporating relationships leads to a practical problem. Prior to entity resolution commencing, there may be no relationships present beyond those contained (explicitly or implicitly) within a single record. Consider the case where a bibliographic data set is being deduplicated. Each paper is related to its authors, and the authors are related to each other, but no relationship connects them to any other paper or author records. Similarly, when matching census data sets, the members of an individual household are related to each other as part of a single census record, however there are no relationships to individuals in different households, or different census periods.

To illustrate this point, consider the example shown in Figure 5.1 where we show part of two households from the UKCD data set, described in Section 2.5. We also show the entity relationship graph created by the two households in the initial state. Each record within a household is related to all others, either explicitly (solid line) or implicitly (dotted line). However, if we calculate an initial (i.e. before entity resolution commences) relationship similarity between two records, for example '1871_6718' and '1881_21469', it is 0 by every measure described in Sec-

Household ID 1422, 1871 Census

Record ID	Surname	First Name	Rel. to Head	Year of Birth	Occupation
1871_6718	Hyslop	Adam	Head	1834	Coal Dealer
1871_6719	Hyslop	Mary	Wife	1833	House Wife
1871_6720	Hyslop	Sarah	Daughter	1860	Scholar
1871_6721	Hyslop	John	Son	1861	Scholar
...					

Household ID 4452, 1881 Census

Record ID	Surname	First Name	Rel. to Head	Year of Birth	Occupation
1881_21469	Hyslop	Adam	Head	1834	Ag. Labourer
1881_21470	Hyslop	Mary	Wife	1833	
1881_21471	Hyslop	Sarah	Daughter	1861	Weaver
1881_21472	Hyslop	John	Son	1862	Stone Cutter
...					

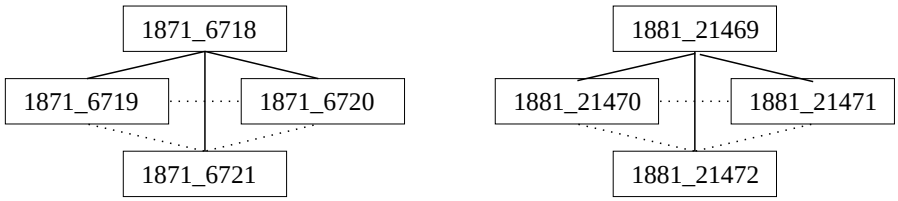

Figure 5.1: Parts of two households from the UKCD data set, and their corresponding representations in an entity-relationship graph.

tion 2.3. In order to have a non-zero measure of relationship similarity, two records must have a common neighbour (or neighbour of neighbour, etc.), which is not possible if they are in separate connected components of the graph.

If relational similarity between every pair of records in different households is initially 0, then for the approach of Bhattacharya and Getoor [12] to work, the weighting of the attribute similarity component has to be higher than the stopping threshold (essentially meaning that attribute similarity alone is sufficient to generate matches) and pairs of records with highly similar attribute values will be matched regardless of their relationship information. This either leads to many false matches, or the problem being solved is such that attribute similarity alone is sufficient to match records, so collective entity resolution is not required.

Similarly, when trying to apply the random walk based approach of Kalashnikoff and Mehrotra [86] no path exists between two records in different households, since they are in separate components of the entity relationship graph. Therefore a random walk starting at a node, has a probability of 0 of reaching any node in another household.

However, if it was possible to somehow match two records (for example the two records selected above, 1871_6718 and 1881_21469), this would allow both the above collective entity resolution techniques to match the remaining records in the two households.

This results in a chicken and egg situation, where in order to match any pair of records, we need to first match some other records, and so the collective entity resolution process never starts. The way collective entity resolution techniques such as those described above deal with this problem is by incorporating some form of bootstrapping or training data at the beginning of the entity resolution process. Essentially, they generate an initial set of matches (and in some cases also non-matches) from which to start the process.

The collective entity resolution approach of Bhattacharya and Getoor [12] solves this problem by essentially starting the process by merging records with identical attribute values (i.e. exact matches). Where the attribute values are common, e.g. a person named 'John Smith', then they also require exact values in the attributes of related records. In this way they can generate an initial set of matches from which to begin the clustering process.

The approach of Kalashnikov and Mehrotra [86] starts by performing traditional pairwise entity resolution down to a preset similarity threshold. Any reference that has only a single potential option with similarity above the threshold is treated as a match. Where there are multiple potential options with similarity above the threshold a choice node is created. This allows the process to start by matching some portion of the references to begin the process. Where this initial matching resolves a high proportion of the references, the results are very good, however they degrade as this proportion decreases suggesting that data sets with highly skewed attribute values may be problematic.

There are two conflicting requirements for the bootstrapping process and they mirror the trade-off between precision and recall described in Section 2.3. The first requirement is that the identified matches should actually be true matches (high precision). The initial bootstrapping results are propagated and used to infer additional matches. If the initial matches are incorrect, then anything inferred from them is more likely to also be incorrect. The second requirement is that the bootstrapping process needs to identify enough matches for the entity resolution process to start (high recall). Essentially, the two records in each true match pair need to be in the same connected component of the entity relationship graph after the bootstrapping process, or they can never be matched in the actual collective entity resolution process. These two requirements effectively mean that the bootstrapping process needs to identify a critical mass of definite matches to start the entity resolution process.

When it is possible to identify such a critical mass of definite matches, collective entity resolution techniques can be very effective. When deduplicating a bibliographic data set, there is no (theoritical) limit to the number of matches each paper or author may have. In addition,

since each author can write any number of papers, and each paper can have any number of authors, there is a possibility of building up long relationship chains. By iteratively making match decisions and propagating the resulting information, it is possible that records which initially start far apart in the entity relationship graph may eventually be matched.

By contrast, when linking the two census data sets shown in Figure 5.1, each individual only has (at most) one true match in the other data set. In addition, many of the households are relatively stable (i.e. they don't change significantly between census periods) so in these cases the individuals in a single household will all be in a single household in the other census period. This means there is no possibility of using a long path to gradually propagate information and infer matches. The bootstrapping process needs to match at least one pair of individuals in each household or they will never be matched in the entity resolution process.

A similar situation applies to population reconstruction, where a birth, death or marriage certificate might only include details of a small number of entities (usually three on a birth certificate, between two and four on a death certificate and six for a marriage certificate). Unless at least one entity on each certificate can be matched in the bootstrapping, none of the records on the certificate will ever be matched in the subsequent entity resolution process. However, the characteristics of the data sets and problems that make current bootstrapping processes less effective in domains such as group linkage and population reconstruction, also present additional features that can be used to improve the generation of the initial set of matches. We discuss these characteristics further in the next section.

5.1.1 One-to-One and One-to-Many Matching Constraints

Many entity resolution problems take place in the presence of one or more constraints (such as those defined in Section 2.2). A common constraint is that each record in a data set can match at most one (or sometimes must match exactly one) record in another data set. For example, when matching two customer databases that are individually clean (i.e. assumed not to contain duplicate records for the same entity), then each customer should be unique in a given database, meaning a record in one of the data sets can match at most one record in the other data set. This one-to-one matching constraint is not guaranteed by the traditional entity resolution process, nor is it guaranteed by collective entity resolution techniques such as relational clustering [12] or random walks [86]. Similar one-to-one or one-to-many constraints apply in other problem domains such as matching flight arrivals to immigration watchlists (one-to-one or one-to-many), matching two voter databases (one-to-one) [24], matching different census periods (one-to-one)

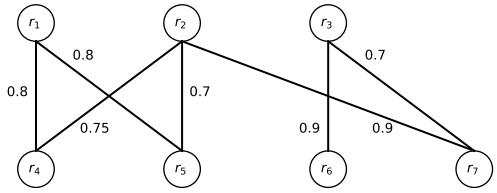

Figure 5.2: Ambiguity example as described in detail in Section 5.2.1.

Node	AmbMax	AmbRatio	AmbCount
r_1	1.0	1.0	1
r_2	0.85	0.83	2
r_3	0.8	0.78	1
r_4	0.95	0.94	1
r_5	0.9	0.88	1
r_6	0.0	0.0	0
r_7	0.8	0.78	1

Table 5.1: Ambiguity calculations (see Sect. 5.2.1)

[4], and matching birth, death and marriage certificates (one-to-one or one-to-many) [156].

In these situations, it is important to consider both the similarity between records and whether there are alternative matching candidates. In other words, is there *ambiguity* about which candidate a record matches. While this complicates the matching process, it also provides additional information that can be exploited to generate training and bootstrapping data. Using a very high attribute similarity threshold as a crude bootstrapping approach can lead to a small number of matches being identified, however, lowering the threshold enough that recall improves can lead to a large number of false matches, particularly in domains where important attributes are very skewed (see Section 1.2.1). However, when there is a one-to-one matching constraint, we can exclude records with multiple candidates and use a lower similarity threshold for the bootstrapping, provided ambiguity is low. This approach can also be combined with other bootstrapping techniques such as looking at the attribute values of related entities in order to identify unambiguous record pairs.

Consider the situation shown in Fig. 5.2 where the nodes in the graph represent records from two data sets, the edges represent potential matches, and the edge weights represent the similarity values between two records. Record r_1 has two potential matches, r_4 and r_5, and is equally similar to both. As such, it is ambiguous whether r_1 should be matched to r_4 or r_5 (and the one-to-one matching constraint means it can not be matched to both of them). Record r_6 however has only a single candidate (r_5), and a high similarity of 0.9 so the ambiguity is very low. In Table 5.1 we calculate the ambiguity for each record in Figure 5.2 using the three ambiguity measures described in Section 5.2.1.

In the remainder of this chapter, we show how we can incorporate one-to-one and one-to-many constraints when generating bootstrapping and training data. By excluding records with high ambiguity, we can consider record pairs with lower attribute similarity as potential matches in the bootstrapping.

5.2 Approach

We start by defining our terms and notation. We assume two data sets $\mathbf{R}_1$ and $\mathbf{R}_2$, and a function $\Psi(r_i, r_j)$ which calculates the similarity between a pair of records $\langle r_i, r_j \rangle$ where $r_i \in \mathbf{R}_1$ and $r_j \in \mathbf{R}_2$. We denote the output of Ψ applied to record pair $\langle r_i, r_j \rangle$ as ψ_{ij}, with $0 \leq \psi_{ij} \leq 1$ for all $\langle r_i, r_j \rangle$ s.t. $r_i \in \mathbf{R}_1$ and $r_j \in \mathbf{R}_2$.

Based on $\mathbf{R}_1$, $\mathbf{R}_2$ and Ψ, we construct a bipartite graph $\mathbf{G} = (\mathbf{N}, \mathbf{E})$, with $\mathbf{N} = \mathbf{N}_1 \cup \mathbf{N}_2$ where each node $n_i \in \mathbf{N}_1$ corresponds to a record $r_i \in \mathbf{R}_1$ and each node $n_j \in \mathbf{N}_2$ to a record $r_j \in \mathbf{R}_2$. An edge exists between every pair of nodes from the two data sets, formally $\mathbf{E} = \mathbf{N}_1 \times \mathbf{N}_2$. We denote the edge between nodes n_i and n_j as e_{ij}. Each edge $e_{ij} \in \mathbf{E}$ has weight ψ_{ij}, corresponding to $\Psi(r_i, r_j)$. In essence, $\mathbf{G}$ is the similarity graph for $\mathbf{R}_1 \times \mathbf{R}_2$.

In order to calculate the ambiguity of record $r_i \in \mathbf{R}_1$, we examine the edges adjacent to the corresponding node $n_i \in \mathbf{N}$. For each record $r_i \in \mathbf{R}_1$ we define $M_1(r_i) = \psi_{ij}$ where $e_{ij} \in \mathbf{E}$ and $\forall e_{il} \in \mathbf{E}, j \neq l, \psi_{ij} \geq \psi_{il}$. In other words, for record $r_i \in \mathbf{R}_1$, r_j is the most similar record in $\mathbf{R}_2$. The case for record $r_i \in \mathbf{R}_2$ is symmetric.

We can generalise this notion to $M_p(r_i)$ where $M_p(r_i)$ is the weight of the edge adjacent to n_i which connects it to the node in the other data set, corresponding to the record with p^{th} highest similarity. For example, $M_2(r_i) = \psi_{ik}$ where $e_{ik} \in \mathbf{E}$, r_j is the record such that $M_1(r_i) = \psi_{ij}$ and $\forall e_{il} \in \mathbf{E}, j \neq k \neq l, \psi_{ij} \geq \psi_{ik} \geq \psi_{il}$.

5.2.1 Measures of Ambiguity

When considering the case of one-to-one or one-to-many matching constraints, we use the ambiguity of a record to determine whether it is 'safe' (i.e. unambiguous) to match with the most similar record from the other data set, or whether there are other potential candidates. For a given record $r_i \in \mathbf{R}$, we consider its corresponding node $n_i \in \mathbf{N}$ and propose the following three measures of ambiguity:

1. The first measure of ambiguity considers the size of the difference between the highest and second highest edge weights among edges adjacent to n_i. The intuition behind this approach is that if the *best* candidate is significantly more similar than the *second best* candidate, then it is much more likely that the best candidate is the correct match. Formally:

$$\text{AmbMax}(r_i) = 1 - M_1(r_i) + M_2(r_i) \tag{5.1}$$

2. The second measure of ambiguity considers the ratio of the weights of the highest and second highest edges among those adjacent to n_i. The intuition behind this approach is similar to AmbMax, however it may be more appropriate for lower values of $M_1(r_i)$, for example if the similarity function Ψ is not well chosen. Formally:

$$\text{AmbRatio}(r_i) = \frac{M_2(r_i)}{M_1(r_i)} \tag{5.2}$$

3 The third measure of ambiguity counts the number of potential matches for n_i. Intuitively, the more potential candidates there are the more difficult it is to pick the correct one. Formally, given a minimum similarity threshold ψ_{min}:

$$\text{AmbCount}(r_i, \psi_{min}) = q - 1 \tag{5.3}$$

where q is the number such that $\forall p$ where $1 \leq p \leq q, M_p(r_i) \geq \psi_{min}$ and $\forall p$ such that $p > q, M_p(r_i) < \psi_{min}$. In other words, q is the number of edges adjacent to n_i with weight greater than or equal to ψ_{min}. We subtract one from the number of options so that an ambiguity value of 0 indicates no ambiguity (i.e. there is only a single option).

For all three measures of ambiguity, a higher value indicates that there are multiple candidates for a true match. The first two measures are normalised to values between 0 and 1. As opposed to AmbMax and AmbRatio, which only consider the two best candidates for matching, AmbCount considers how many candidates exist. It is possible to normalise AmbCount to a value between 0 and 1, such as by dividing through by the maximum value across the entire data set, however we did not find this particularly useful in practice.

Because AmbMax and AmbRatio only consider two edges whereas AmbCount considers the total number of edges satisfying a similarity condition, we found AmbCount ended up being useful in a different way to the other two measures. As such, we use AmbMax and AmbRatio for unsupervised generation of training data, whereas AmbCount was more applicable to active learning. We discuss normalisation, the selection of an ambiguity measure, and the various benefits of each measure in Section 5.5.4.

We illustrate these ambiguity concepts with reference to the example in Figure 5.2. Calculating $\text{AmbMax}(r_1) = 1 - 0.8 + 0.8 = 1.0$, in other words r_1 is highly ambiguous, which makes sense given its two possible candidates for matching are both equally similar. $\text{AmbMax}(r_7) = 1 - 0.9 + 0.7 = 0.8$, still ambiguous but less so than r_1. In practice an ambiguity of 0.8 may

Algorithm 5.1: Static Generation of Bootstrapping Data

Input:
- A bipartite similarity graph **G: N, E** - constructed from data sets $\mathbf{R}_1$ and $\mathbf{R}_2$;
- An ambiguity measure equation (5.1) or (5.2) A;
- A minimum similarity threshold ψ_{min};
- A maximum ambiguity threshold α_{max};

Output:
- Set of matching record pairs to use as training data **M**

```
1:  foreach r_i in R_1 ∪ R_2 do:
2:      α_i = CalculateAmbiguity(A, r_i)          // Calculate the ambiguity of each record
3:  M = ∅                                         // Create an empty set of matches
4:  foreach e_ij in E do:
5:      if M_1(r_i) ≠ e_ij or M_1(r_j) ≠ e_ij:
6:          continue
7:      if ψ_ij ≥ ψ_min and α_i ≤ α_max and α_j ≤ α_max:   // Filter the record pairs based on ψ_min and α_max.
8:          M.add(⟨r_i, r_j⟩)                     // Add similar and unambiguous record pairs to the training set.
9:  return M
```

be enough for us to be satisfied that we can perform the match, although this is dependent on the data set and the similarity function Ψ. Calculating $\text{AmbRatio}(r_2) = 0.75/0.9 = 0.83$ and $\text{AmbRatio}(r_4) = 0.75/0.8 = 0.94$. The result for r_2 may be usable, but the ambiguity of r_4 suggests that we likely cannot be confident about the correct match for this record.

5.2.2 Using Ambiguity to Generate Bootstrapping and Training Data

In order to generate bootstrapping data, we extend a threshold based technique by incorporating ambiguity. We present two versions of our approach. The first approach is static where the candidate pairs are filtered based on a maximum ambiguity threshold and a minimum similarity threshold. This approach is described in Algorithm 5.1.

The second approach is iterative and the candidate pairs are ordered according to some criteria (for example a combination of similarity and ambiguity), and kept in a priority queue. After selecting the first pair from the queue, we update the ambiguity values of any affected pairs before selecting the next pair. This allows us to start by selecting the most similar and unambiguous record pairs, and by resolving these first the ambiguity of other candidate pairs may reduce, allowing them to also be matched. This selection and update process repeats until there are no more pairs satisfying our criteria, or we have a sufficient number of matches for our purpose. This approach is described in Algorithm 5.2.

5.2.3 Static Generation of Bootstrapping Data

The intuition behind the static approach is that if each record in a candidate pair has only the other record as a good candidate for matching, then we can be confident that they should

Algorithm 5.2: Iterative Generation of Bootstrapping Data

Input:
- A bipartite similarity graph **G: N, E** - constructed from data sets $\mathbf{R}_1$ and $\mathbf{R}_2$;
- An ambiguity measure equation (5.1) or (5.2) A;
- A minimum similarity threshold ψ_{min};
- A maximum ambiguity threshold α_{max};

Output:
- Set of matching record pairs to use as training data **M**

```
1:  foreach r_i in R_1 ∪ R_2 do:
2:      α_i = CalculateAmbiguity(A, r_i)
3:  Q = CreatePriorityQueue()                    // Create an empty priority queue
4:  foreach e_ij in E do:
5:      π_ij = CalculatePriority(ψ_ij, α_i, α_j)   // Calculate priority of record pair ⟨i, j⟩
6:      Q.insert(⟨1 − π_ij, i, j⟩)                 // Insert into Q, ordered by priority
7:  M = ∅                                        // Empty set for the matches
8:  while Q not empty do:
9:      π_ij = Q.pop()                           // Get the first element in the queue
10:     if (ψ_ij < ψ_min) or (α_i > α_max and α_j > α_max):
11:       continue                               // Ambiguity too high or similarity too low
12:     else:
13:       M.add(⟨r_i, r_j⟩)                      // Add the record pair to M
14:       N.remove(n_i)                          // Remove n_i and adjacent edges
15:       N.remove(n_j)                          // Remove n_j and adjacent edges
16:       UpdateAmbiguityValues(A, r_i, r_j)     // Update the ambiguity values
17:       RemoveFromQueue(Q, n_i, n_j)           // Remove priorities for n_i and n_j
18:       UpdatePriorities(Q, n_i, n_j)          // Update remaining priorities in Q
19: return M
```

be matched. In order to determine which candidate pairs satisfy these conditions, the static approach filters the record pairs according to a similarity threshold ψ_{min} and an ambiguity threshold α_{max}. We begin by calculating the ambiguity for each record in $\mathbf{R}_1 \cup \mathbf{R}_2$. Then for each edge $e_{ij} \in \mathbf{E}$ we check a series of conditions. Firstly, we test that r_i is the most similar record to r_j, and vice-versa (line 5). Then we check that $\psi_{ij} \geq \psi_{min}$ and that the ambiguity of both r_i and r_j is less than α_{max} (line 7). Any record pair that satisfies all these conditions is added to the set of matches **M** (line 8).

It is important to note that provided α_{max} is strictly less than 1.0, the combination of conditions in lines 5 and 7, guarantees the one-to-one matching constraint holds for each record that is part of a record pair in **M**. For a record $r_i \in \mathbf{R}_1 \cup \mathbf{R}_2$, the only way its corresponding node can have multiple edges satisfying the condition in line 5 is if each edge has the same similarity value. However, by both the AmbMax and $\mathrm{AmbRatio}$ ambiguity measures described in Section 5.2.1, such records have ambiguity 1.0 since $M_1(r_i) = M_2(r_i)$ (and possibly $= M_3(r_i)$ etc.). As such, provided α_{max} is strictly less than 1.0, such nodes will be excluded by the conditions in line 7.

5.2.4 Iterative Generation of Bootstrapping Data

The iterative approach to generating boostrapping data is described in Algorithm 5.2. The idea is similar to that of the static approach in that we try to select record pairs that are similar and unambiguous to use as matches. However, there are key differences between the two approaches. The iterative approach begins by calculating the initial ambiguity score for each record in $\mathbf{R}_1 \cup \mathbf{R}_2$ (lines 1 and 2). Then, for each edge in $\mathbf{E}$, we calculate a priority score π_{ij} based on a combination of the similarity score ψ_{ij} and the ambiguity of r_i and r_j and insert these scores into a priority queue (lines 5 and 6).

The loop in lines 8 to 18 is the main part of the algorithm which iterates until the priority queue Q is empty. In each iteration of the loop we remove a record pair from the queue and check whether it satisfies the similarity and ambiguity requirements (line 10). If the requirements are not satisfied, we skip to the next pair in the queue. If the requirements are satisfied (line 12), we add the record pair to our match set (line 13). Once a record pair has been matched, the one-to-one matching constraint means that the two records in the pair should no longer be considered for other matches, nor should they affect the ambiguity of other records. The remainder of the loop (lines 14 to 18) is spent updating the queue and the ambiguity values of other records to reflect this. We remove the two nodes from the graph along with any other edges that connect to either of them (lines 14 and 15). We then recalculate the ambiguity of all records that have been affected by the removal of nodes n_i and n_j, and re-order the queue if required.

In practice, the iterative approach takes longer to run than the static approach, since ambiguity and priority values and the queue all have to be updated after a match is found. However, for entity resolution problems where the majority of records have a true match in the other data set, finding a match can reduce the ambiguity of other records, potentially allowing them to also be matched. This means the iterative approach tends to achieve higher recall than the static approach. We discuss these differences further in Section 5.5.

5.3 Applications to Active Learning

Active learning (described in Section 3.4) is a commonly used approach to generating training data. Unlike the bootstrapping approaches used by collective entity resolution techniques, and the approaches we presented in the previous section, active learning is a semi-supervised technique, meaning it requires a degree of manual classification. The idea behind active learning

Algorithm 5.3: Active Learning Based Generation of Bootstrapping Data

Input:
- A bipartite similarity graph **G: N, E** - constructed from data sets $\mathbf{R}_1$ and $\mathbf{R}_2$;
- An ambiguity measure equation (1) or (2) A;
- A minimum similarity threshold ψ_{min};
- A maximum ambiguity threshold α_{max};
- The maximum number of manual classifications b_{max};
- The number of manual classifications per iteration s;

Output:
- Set of matching record pairs to use as training data **M**

```
1:  M = RunStaticApproach(G, A, ψ_min, α_max)      // Run the static approach to get an initial set of matches
2:  b_current = 0                                  // Set the current number of classifications to 0
3:  while b_current < b_max do:                    // Run while there is budget remaining
4:      foreach r_i in R_1 ∪ R_2 do:
5:          α_i = CalculateAmbiguity(AmbCount, ψ_min, r_i)   // Calculate AmbCount for each node
6:      S = GetAmbiguousRecords(R_1 ∪ R_2, s)      // Get the s most ambiguous records
7:      foreach r_i in S do:
8:          m_i = ManuallyMatch(r_i)               // Manually select the appropriate true match
9:          M.add(m_i)
10:     UpdateGraph(G, M)                          // Remove from G matched nodes and adjacent edges
11:     b_current = b_current + s                  // Update the budget used
12:     M = M.union(RunStaticApproach(G, A, ψ_min, α_max))   // Rerun the static approach for newmatches
13: return M
```

is that an oracle (which could be a domain expert, a crowd sourcing initiative, etc. [160]) is presented with example pairs and asked to label them as *match* or *non-match*. In this way a training set is built up and used to learn the overall model. Because such manual classification is typically resource intensive, the aim of active learning is to try and minimise the number of manual classification decisions that are required in order to produce a satisfactory classifier. The problem can alternatively be posed as how to produce the best classifier given a fixed budget (i.e. a maximum number) of manual classifications.

As described in Section 2.3, entity resolution can be described as a binary classification problem. However, the two classes, *match* and *non-match* are typically very imbalanced with *non-matches* often outnumbering *matches* by several orders of magnitude, particularly for large data sets. We give a simple scenario showing this in Section 6.1. Any strategy that is used to select record pairs for manual classification needs to account for this class imbalance and ensure that enough *matches* are selected to provide training examples for both classes. This means that active learning techniques from other domains using density [161, 191], ambiguity [154], and other approaches (see Section 3.4), may require adaptation or may not work well at all for entity resolution.

For situations where one-to-one and one-to-many matching restrictions apply, we propose using AmbCount as a possible selection strategy. However, instead of only using manually classified record pairs to build the training data, we combine the active learning approach with our static approach. This mean that unambiguous pairs can be matched automatically, and

manual effort is reserved for those records where there is ambiguity.

The reason we only use $\mathrm{AmbCount}$, is because AmbMax and $\mathrm{AmbRatio}$ only consider the two best candidates for matching, whereas AmbCount measures how many candidates there are. This means that by choosing nodes with high values of $\mathrm{AmbCount}$, we can select nodes for manual classification which will remove many edges from **G**. In this way, we maximise the number of additional matches that can be gained from each manual classification, due to a reduction in ambiguity of the neighbouring nodes.

Our active learning approach is shown in Algorithm 5.3 and works as follows. We start by running our static approach to get an initial set of matches (line 1). Then the bulk of our work is done inside the loop (lines 4-12) which runs while there is manual classification budget remaining. We calculate the ambiguity for each node using $\mathrm{AmbCount}$ (line 5) and get the s most ambiguous nodes remaining (line 6). For each of these nodes we use our manual classifier to select the corresponding matching record (line 8) and add it to our set of matches (line 9). Note that in the event there is no matching record, we skip this step, but still remove the node from **G**. We then update our graph, by removing all nodes that have been successfully matched, along with their adjacent edges (line 10). We update our budget (line 11), before re-running the static approach to find any matches that have become unambiguous as the result of manual classification (line 12).

The idea behind our active learning approach is that by selecting the nodes with the highest value of $\mathrm{AmbCount}$, we achieve the greatest flow on effect on the ambiguity of other nodes which maximises the likelihood of additional matches being detected by our static approach.

Finally, for the sake of clarity, it is worth noting that our definition of ambiguity is different to the definition that is traditionally used in active learning [154]. We use the term *ambiguity* as we defined it in Section 5.2.1, in other words there is ambiguity about which record out of a number of possible candidates is the correct match. Active learning traditionally uses ambiguity to refer to examples that are very hard to classify, in other words, ambiguous examples are ones where the classifier has difficulty determining which class they belong to.

5.4 Evaluation

Of the three data sets that we use in this thesis, only the UKCD data set and the NCVR data set have meaningful one-to-one or one-to-many matching constraints. The CORA data set has the constraint that a single author cannot match with two different authors on the same paper,

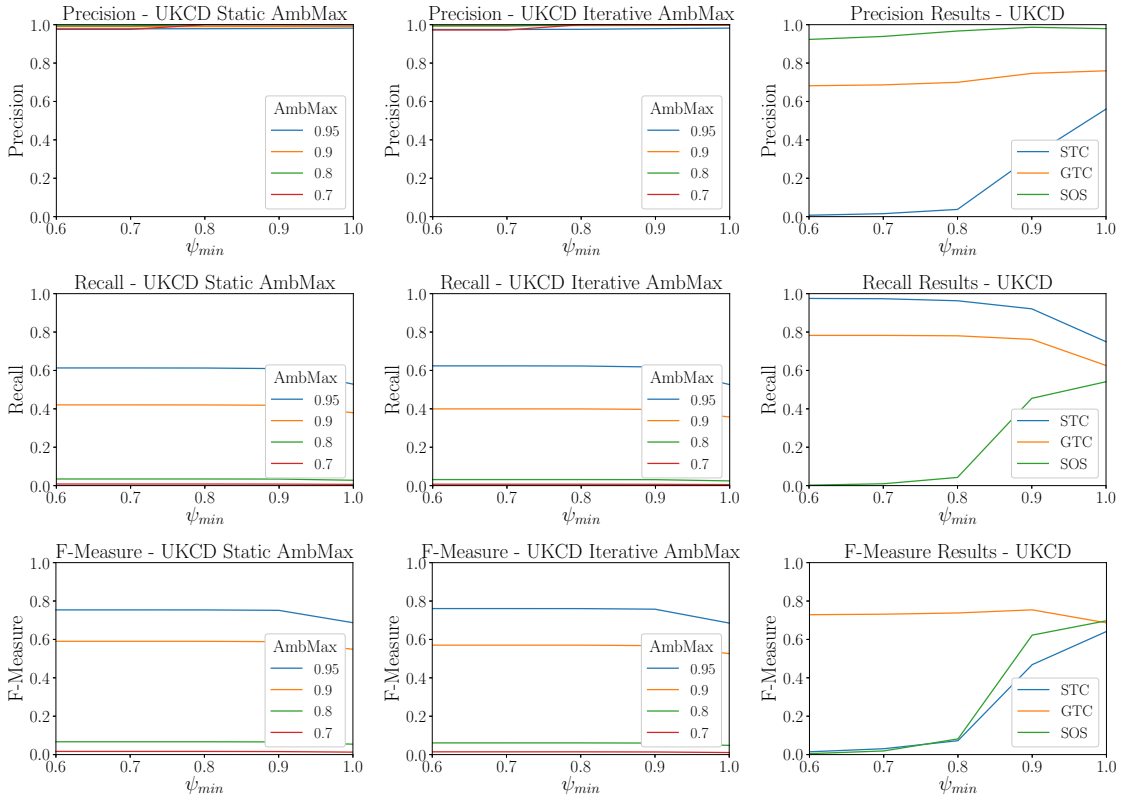

Figure 5.3: Precision (top row), recall (middle row), and the f-measure (bottom row) results for the UKCD data set. Shown are the results for the static approach (left), iterative approach (middle), and baselines (right).

however in practice this restriction is too narrow to allow our approaches to work.

As a result, instead of evaluating our approaches on CORA, we use a version of the NCVR-450 data set, which contains only those records that have a true match. This is referred to as NCVR-280 in the results. This allows us to evaluate whether the proportion of records with a true match has an impact on our results, and we discuss this further in the next section. We also evaluate our techniques on the UKCD and NCVR-450 data sets as in other chapters.

To evaluate our approaches, we compare them against three different baseline approaches. To ensure a fair comparison, we use the same blocking methods for our approaches and all three baselines. We also use the same attributes, weightings and string comparison functions to calculate similarity for each baseline approach as we do for our approaches. This means that our approaches and each of the baseline approaches have identical sets of candidate pairs and similarity values as inputs. The three baseline approaches are:

1. **Simple threshold based classifier (STC)**: the first baseline is traditional entity resolution

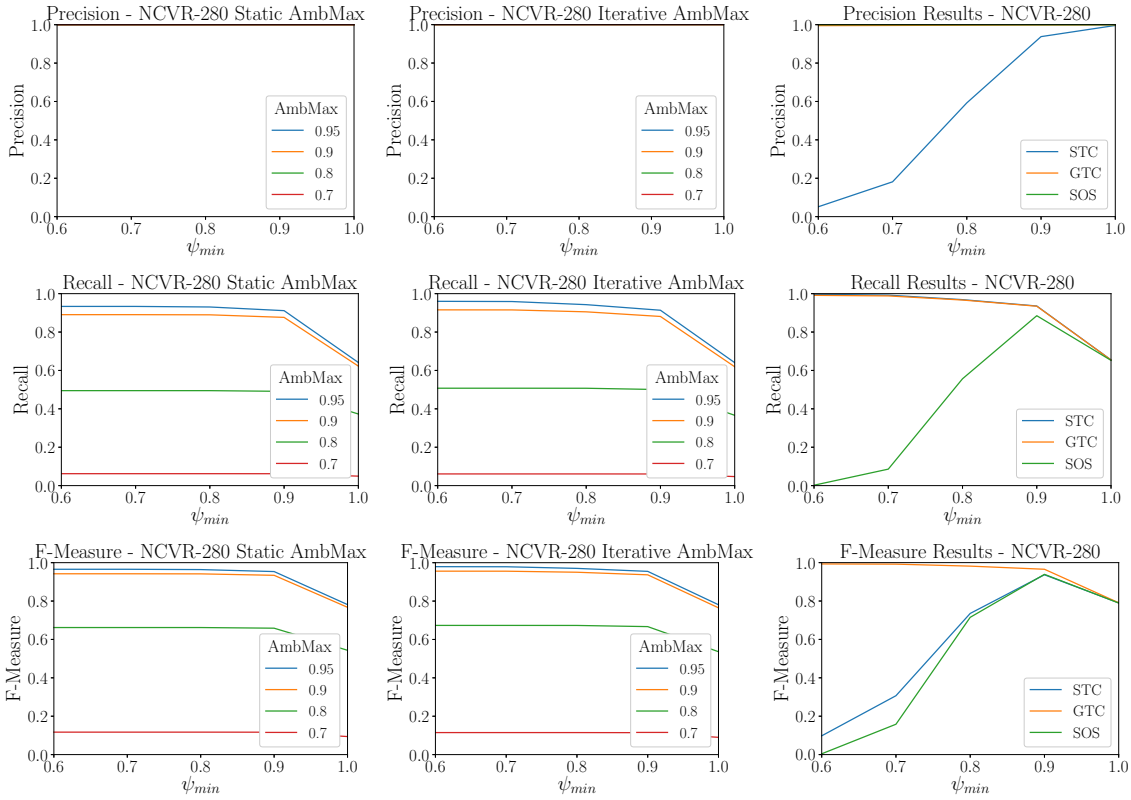

Figure 5.4: Precision (top row), recall (middle row), and the f-measure (bottom row) results for the NCVR-280 data set. Shown are the results for the static approach (left), iterative approach (middle), and baselines (right). We note that for both of our approaches, for all similarity values, the precision results are very close to 1.0.

where all candidate pairs with similarity above the threshold ψ_{min} are treated as matches.

2. **Greedy threshold based classifier (GTC)**: the second baseline is similar to the simple threshold based classifier except we also enforce the one-to-one matching constraint. This is done in a greedy fashion, i.e. by selecting record pairs in order of descending similarity (ψ_{ij}) with ties being broken arbitrarily.

3. **Single option set (SOS)**: our third baseline mirrors the bootstrapping approach of Kalashnikov and Mehrotra [86] in that for a candidate pair to be classified as a match, each record in the pair must have only a single option (the other record) with similarity above the threshold ψ_{min}.

We note that while we do not separately implement the bootstrapping approach of Bhattacharya and Getoor [12], the basic idea is captured in the first baseline approach when $\psi_{min} = 1.0$.

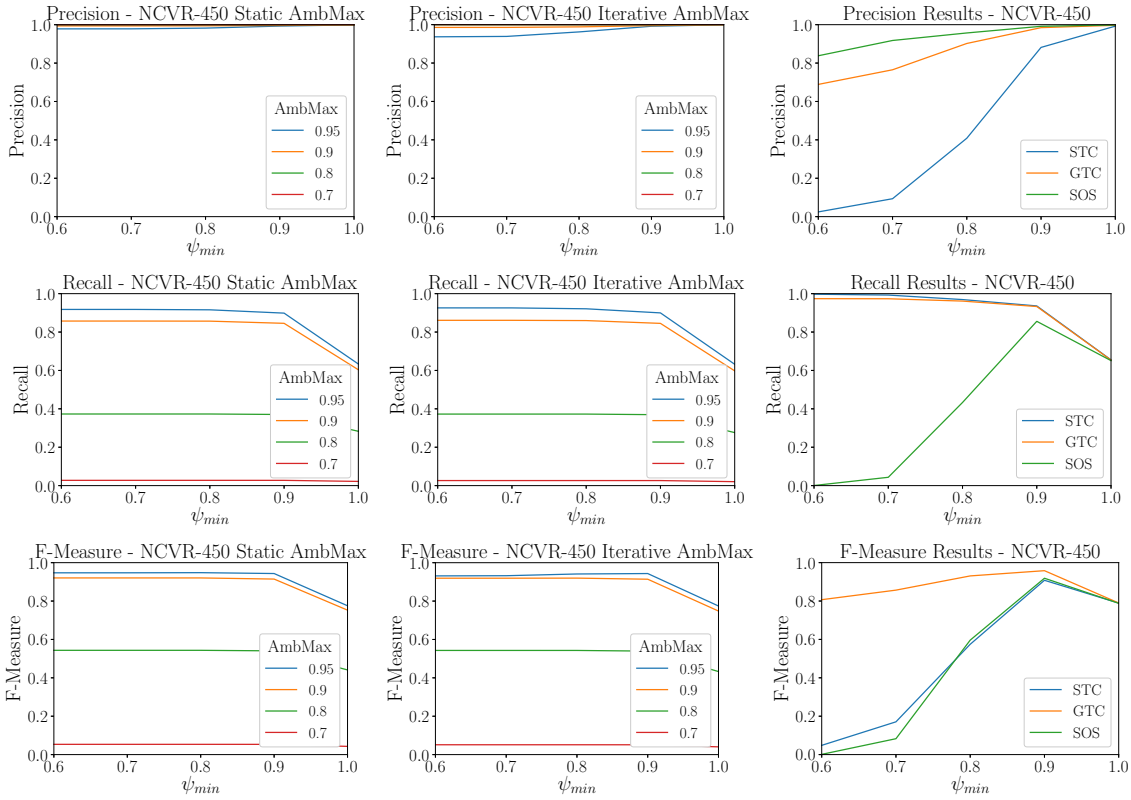

Figure 5.5: Precision (top row), recall (middle row), and the f-measure (bottom row) results for the NCVR-450 data set. Shown are the results for the static approach (left), iterative approach (middle), and baselines (right).

For our evaluation, we use precision, recall and the f-measure, which are described in Section 2.3. However, we note that as described in Section 5.1, the primary goal of generating both training data and bootstrapping data is to generate 'enough' high quality matches. This means that we prioritise precision in our evaluation, provided recall is not extremely low (we discuss this further in the next section). The results of both the static approach, the iterative approach and the baseline approaches are shown in Figures 5.3, 5.4 and 5.5 for the UKCD, NCVR-280 and NCVR-450 data sets respectively. We limit our reporting to just AmbMax, to avoid presenting several figures that are almost identical. We discuss the differences between the ambiguity measures in the next section, as well as describing some situations where the differences in results could be more significant.

For the UKCD data set, we also evaluate the proportion of households that have at least a single true match, since this is important for being able to bootstrap collective entity resolution techniques such as those of Bhattacharya and Getoor [12] and Kalashnikov and Mehrotra [86].

The results are shown in Figure 5.6.

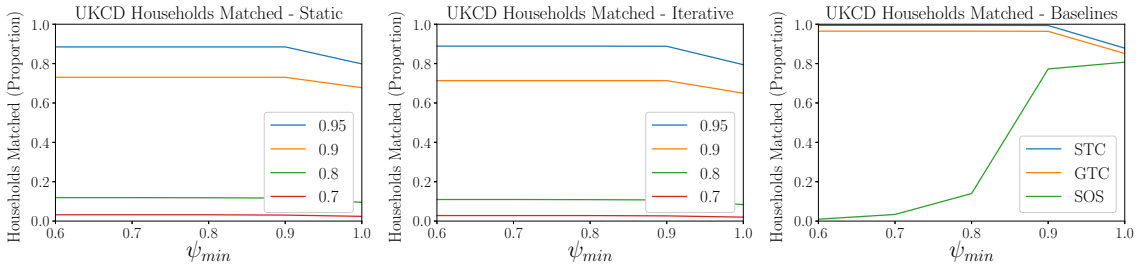

Figure 5.6: Proportion of households in the UKCD data set where at least one record is correctly matched.

In addition to testing the quality of our approaches, we also test their scalability with respect to queue size, similarity threshold and ambiguity threshold. The scalability results are shown in Figure 5.7.

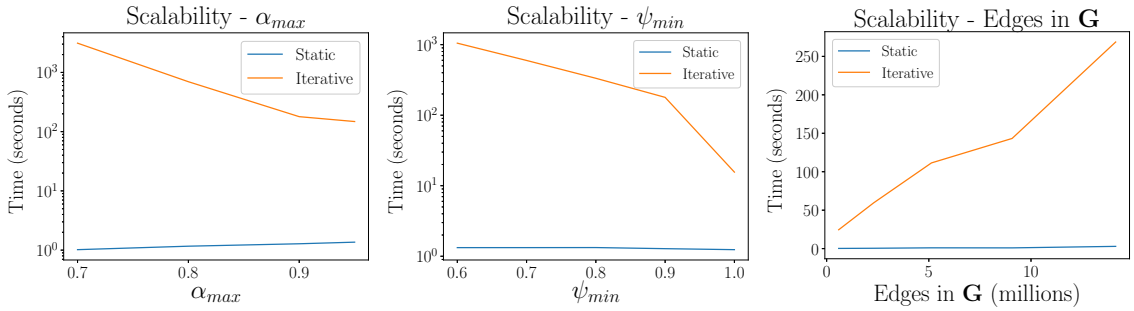

Figure 5.7: The scalability of our approaches, showing the impact on running time of α_{max}, ψ_{min} and the number of edges in **G** respectively.

5.4.1 Evaluation of Active Learning

In addition to testing our static and iterative approaches, we also evaluated our active learning approach by running it on the NCVR-280 and NCVR-450 data sets. We evaluated the numbers of true and false matches that occurred as the result of each manual classification, for different values of ψ_{min} and α_{max} and for different maximum budgets. This allowed us to test how effective our strategy was at overcoming the class imbalance problem that is inherent in active learning based entity resolution. It also allowed us to determine how much of an impact the manual classification process had on reducing the overall ambiguity of the remaining records (i.e. whether a manual classification leads to additional manual classifications due to the reduced ambiguity of other records). The results of our experiments are shown in Figure 5.8.

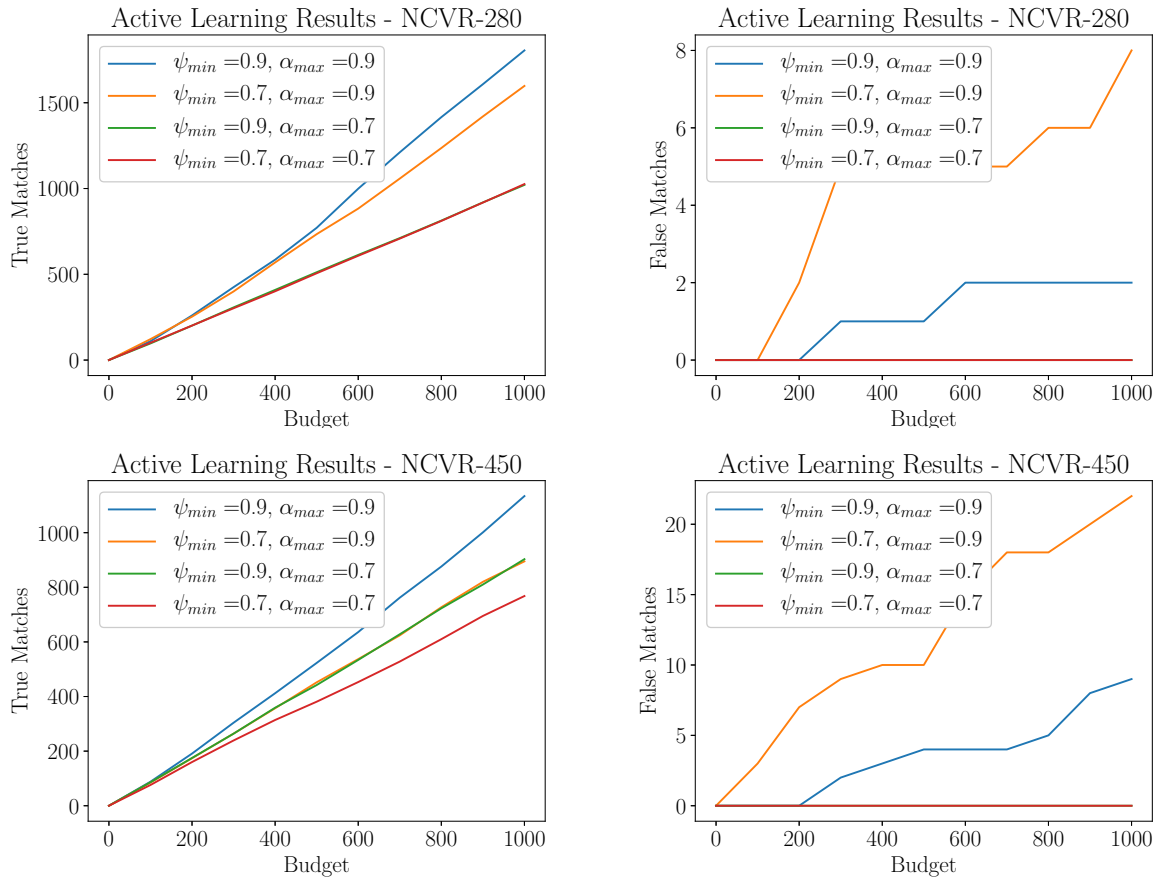

Figure 5.8: Number of true matches and number of false matches produced by the active learning approach for different manual classification budgets for the NCVR-280 data set (top row) and NCVR-450 data set (bottom row).

5.5 Discussion

Overall, our experimental evaluation demonstrates the effectiveness of using ambiguity as part of the training data generation process. On all three data sets we achieve very high precision values for all tested values of ψ_{min} and α_{max}. Recall depends on the choice of α_{max} to a much greater extent than precision. We test a range of values for α_{max}, however in all our experiments the most significant impact on precision comes from introducing any ambiguity threshold (i.e. $\alpha_{max} = 0.95$). Tighter restrictions on ambiguity (achieved by lowering α_{max}) do improve precision, but at the cost of a much greater impact on recall due to the removal of true matches.

The baseline approaches did well in some circumstances. The SOS baseline achieves a high precision in most cases, since it is effectively a binary measure of ambiguity. The GTC baseline does particularly well on the NCVR-280 data set, but it does very poorly on the UKCD data

set. Even though the F-Measure results for the UKCD data set are similar to our approaches, a precision between 0.6 and 0.8 is unlikely to be effective for bootstrapping or training data generation, due to the propagation of incorrect information from the false matches. The STC baseline does not work particularly well for anything except the NCVR-280 data set with ψ_{min} very close to 1.0.

It is worth commenting on the SOS baseline recall results, since they contradict the general entity resolution principle that a reduction in ψ_{min} should increase recall. Using the SOS baseline, a candidate pair is only matched if both records have no alternative option with similarity above ψ_{min}. However, lowering the value of ψ_{min} increases the number of alternatives, and thus actually reduces the number of candidate pairs that can be matched.

We also note that even though the differences in precision values are small in many cases, this can in fact be significant for bootstrapping purposes. Given the potential for incorrect matches to propagate throughout the subsequent classification process, even a small number of false matches can have a significant impact on the final results. For example, in the UKCD data set, incorrectly matching two records means that other records from the same household are also likely to be incorrectly matched (due to the relationship information generated by the initial incorrect match), so the impact on precision in the final matching process can be much greater than just the single incorrect match during bootstrapping.

The experiments evaluating the proportion of UKCD households where at least one member was successfully matched, shown in Figure 5.6, show a similar pattern to the recall results in Figure 5.3. This is to be expected since these experiments essentially measure group recall. Both the STC and GTC baselines manage to match at least one member of almost all the households. However, this is done at the expense of including many false matches. Despite the high recall, this likely makes them unsuitable for bootstrapping purposes, since the contradictory information will lead to poor results in the entity resolution stage. The SOS baseline approach performs better when ψ_{min} is close to 1.0, although it is still below the results achieved by our approaches with α_{max} equal to 0.95. As with the overall results, for our approaches recall reduces very quickly with a stricter ambiguity threshold (i.e. a lower value of α_{max}), with only very small improvements in precision.

The results for NCVR-280 are better than the results for NCVR-450 for all approaches, both of ours and the three baselines. NCVR-280 was created by removing the records from NCVR-450 that did not have a true match. Given these records cannot improve recall (they are not part of a true match) and they can only decrease precision (if they are incorrectly matched with another

record), then removing them will always result in an improvement in the results. However, this also demonstrates an important consideration for our approaches. For ambiguity to assist in choosing records that can be 'safely' matched, unambiguous candidate pairs need to actually be matches, rather than record pairs with very similar (and likely unusual) attribute values purely by coincidence. If the proportion of records with a true match is too low, then such coincidental matches are likely to make up a larger proportion of the matches.

The difference in results between the NCVR-280 data set and the NCVR-450 data set, illustrates the main consideration when choosing between the static approach and the iterative approach. The idea behind the iterative approach is that by resolving the unambiguous matches first, this will mean that other matches are less ambiguous and can be safely matched, which in turn reduces the ambiguity of further matches, and so on. However, this does not work if a large proportion of records do not have a true match. Such records cannot be (correctly) matched, so when they contribute to the ambiguity of other records, this ambiguity cannot be reduced except through a false match. Given that in all our experiments our approaches produce few (if any) false matches, the benefit of using the iterative approach on data sets with a low proportion of records with a true match is greatly reduced. In such cases, the static approach should be used due to its better efficiency.

5.5.1 Complexity and Optimisations

The results of our scalability experiments are shown in Figure 5.7. The experiments are conducted on the NCVR-280 data set, with control values of $\alpha_{max} = 0.9$, $\psi_{min} = 0.9$ and $|\mathbf{E}| = 14.15$ million. Both approaches scale approximately linearly with respect to the number of edges in $\mathbf{G}$, i.e. they are $O(|\mathbf{E}|)$. The choice of α_{max} and ψ_{min} has only a minor effect on the running time of the static approach. However, for the iterative approach, ψ_{min} directly affects the size of the queue that needs to be processed with corresponding impact on the processing and update time. In addition, α_{max} directly affects the number of edges that can be pruned from $\mathbf{G}$ in the iterative approach (as discussed below).

The complexity of the static approach is $O(|\mathbf{E}|)$. This means that in the absence of any blocking it is $O(|\mathbf{R}_1||\mathbf{R}_2|)$. Calculating the ambiguity for each node is $O(|\mathbf{E}|)$, since for each record $r_i \in \mathbf{R}_1 \cup \mathbf{R}_2$ we have to calculate $M_1(r_i)$ and $M_2(r_i)$, which requires examing every edge twice. Lines 4 to 8 of Algorithm 5.1 also require examining each edge in $\mathbf{E}$ and the conditions can all be checked in constant time, so this is again $O(|\mathbf{E}|)$.

In the absence of any blocking, and with no minimum similarity threshold set, Algorithm 5.2

has a space complexity of $O(|\mathbf{R}_1||\mathbf{R}_2|)$ (the length of the queue) and a time complexity of $O(|\mathbf{R}_1||\mathbf{R}_2| \times (|\mathbf{R}_1| + |\mathbf{R}_2|) \times log_2(|\mathbf{R}_1||\mathbf{R}_2|))$. The time complexity is due to each element in the queue potentially requiring a delete or update operation for every record in the two data sets, with both types of operation being logarithmic in the length of the queue.

While the asymptotic time and space complexities for Algorithm 5.2 are poor in the worst case, as we show in Figure 5.7, scalability is better than this in practice due to optimisations which were made to improve performance. As is common in entity resolution, blocking has been applied to reduce the number of edges in $\mathbf{G}$. The queue has also been reduced by removing any edge from $\mathbf{G}$ where the weight cannot impact the ambiguity of a node. Any edge with weight less than $\psi_{min} - (1 - \alpha_{max})$ cannot affect ambiguity and can safely be removed prior to constructing the queue. As a result, as α_{max} gets smaller, more edges have to be retained which increases the running time. In addition, if significantly larger data sets were being used than the NCVR data set, then further optimisations could be made. Alternatively, a tighter blocking approach such as sorted neighbourhood [73] or the size constrained blocking described in Chapter 4 can be used to further reduce the number of candidate record pairs.

5.5.2 Active Learning Evaluation

Our active learning approach is effective at overcoming the class imbalance problem and generating true matches to form part of the training set. However, the results for NCVR-280 are significantly better than those of NCVR-450. This is to be expected since every record in NCVR-280 has a true match so each manual classification will likely produce at least one true match (the record pair involved in the manual classification, and any additional record pairs that become unambiguous once the manual classification is performed). However, the number of true matches per manual classification increases with increased budget, so our approach requires a certain critical mass of manual classifications in order to reduce average ambiguity to the point where many additional true matches can be obtained.

This leads to two important considerations. Firstly, that a high value of α_{max} may be required, especially if the manual classification budget is low. Otherwise the active learning process may not reduce average ambiguity to the point where many true matches are found beyond those obtained through the manual classification process itself. Secondly, since our approach relies on selecting the true match (where it exists) from a set of possible matches, it is likely that each manual classification will be more resource intensive than for traditional active learning where the manual investigation only requires classifying a single record pair. As such, we suggest that

the practical applications of our approach are limited to entity resolution problems where most of the records have a true match in the other data set. This means that it is likely that each manual classification will obtain at least one true match. If this is not the case, the additional effort required for each manual classification is unlikely to be worthwhile.

5.5.3 Modifications for 1-to-Many Matching Situations

Algorithms 1 and 2 assume a one-to-one matching constraint, i.e. each record in either data set can match at most one record in the other data set. While this is a common situation [23], it may not always apply. For example, one of the data sets to be linked may be sufficiently dirty that we have to consider the possibility of duplicates and therefore a one-to-one matching constraint may not be appropriate. Alternatively, matching birth certificates to marriage certificates should capture the restriction that a person can only be born once but can potentially be married multiple times. As a result, while the bride or groom on a marriage certificate should match to the baby on at most one birth certificate, the baby on a birth certificate can match to the bride or groom on multiple marriage certificates.

We can incorporate one-to-many constraints in our static approach. Assuming the constraint applies to the records in $\mathbf{R}_1$ (the reverse is symmetric) we modify Algorithm 1 as follows. In line 1, we only calculate ambiguity for each r_i `in` $\mathbf{R}_1$. In line 5 we only test whether $M(r_i) \neq e_{ij}$ and in line 7 we remove the final part of the condition, i.e. $a_j \leq t_{amb}$ and only consider the weight of e_{ij} and the ambiguity of r_i.

While it is possible to modify the iterative approach in a similar fashion, there is no benefit to doing so. For one-to-many matching constraints, the ambiguity of a record does not change as the result of other matches so there is no advantage to processing the records in a certain order. This means that with the same parameter settings, the results from the two approaches will be identical. As a result, only the static approach should be used due to its better efficiency.

5.5.4 Choice of Ambiguity Measure and Normalisation

Based on our experimental evaluation, AmbMax and AmbRatio give very similar results on each data set. The top half of Figure 5.9 shows the f-measure results for the two different ambiguity measures using the iterative approach on the UKCD data set. The two sets of results are essentially identical and a similar pattern was observed for the results for precision and recall on the UKCD data set, as well as for all three measures on the other data sets. As such,

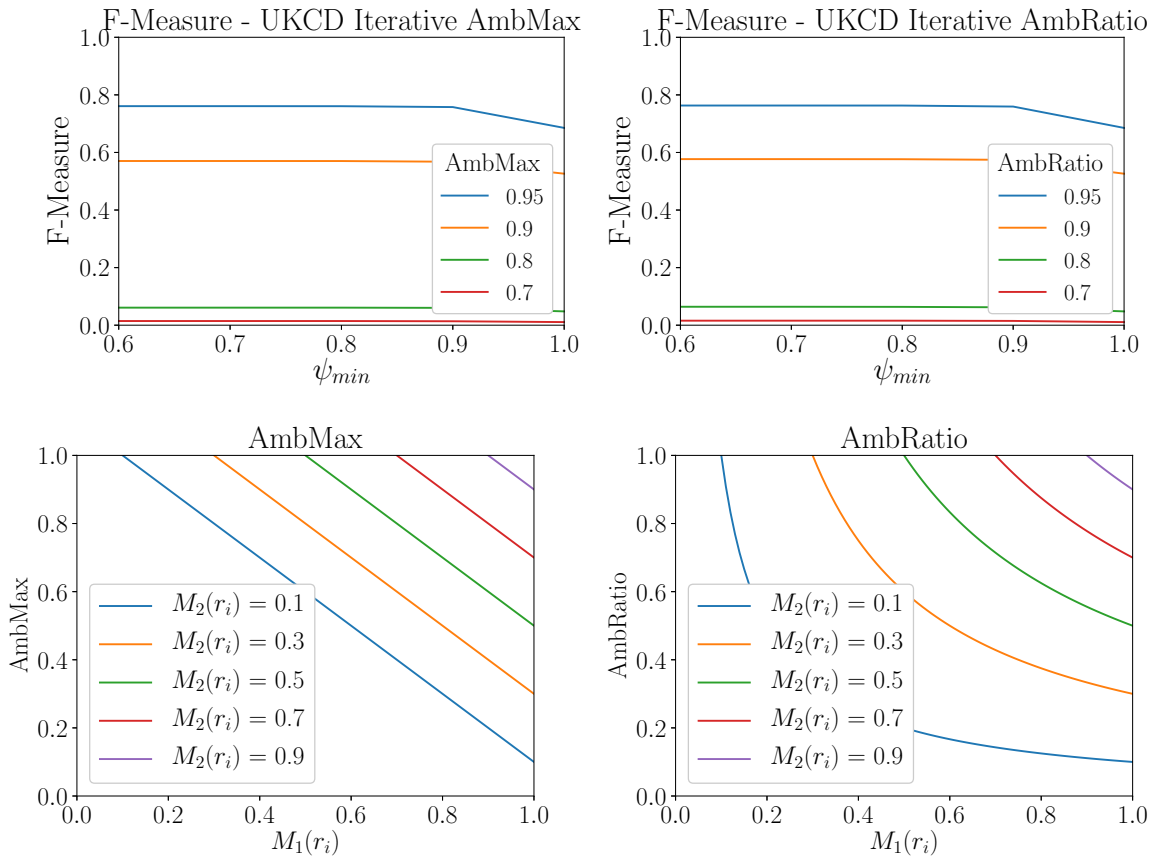

Figure 5.9: Comparison of `AmbMax` and `AmbRatio`. The top two figures show the f-measure results for AmbMax and AmbRatio respectively on the UKCD data sets with the iterative approach. The bottom two figures show how the values of AmbMax and AmbRatio change with respect to $M_1(r_i)$ and $M_2(r_i)$ (the two most similar records). If $M_1(r_i)$ and $M_2(r_i)$ are both greater than 0.7, the difference between the two ambiguity measures is relatively small.

we limit our reporting of results to AmbMax and conclude that either technique can be used in most situations.

However, as shown in the bottom half of Figure 5.9, there are combinations of $M_1(r_i)$ and $M_2(r_i)$ where the values of AmbMax and AmbRatio are further apart. This occurs when the similarity values are low, especially if they are below 0.5. While such situations are unusual in entity resolution, they can occur if the similarity functions are poorly chosen or the data set(s) are extremely dirty. As such, we retain both measures of ambiguity to provide an alternative option should this type of scenario occur.

We found that AmbCount was not useful in generating boostrapping data. In practice, any number of candidates beyond one, meant that there was ambiguity and the actual number of extra candidates was less important. This essentially meant that AmbCount worked as a crude

version of the other two ambiguity measures. We also considered whether some measure of the full distribution of similarity values would be useful for AmbMax or $\mathrm{AmbRatio}$, but for the same reason that $\mathrm{AmbCount}$ was unsuccessful, this also yielded poor results.

5.5.5 Generating Negative Training Examples

While the collective entity resolution techniques of Bhattacharya and Getoor [12] and Kalashnikov and Mehrotra [86] only require positive training examples (i.e. matches) to use as bootstrapping data, collective entity resolution techniques such as MLNs [150] also require negative training data (i.e. non-matches). Many other advanced entity resolution techniques are also supervised including group linkage techniques [52, 54], and population reconstruction techniques [103, 111]. In addition, traditional supervised classification techniques such as support vector machines [54], decision trees [35] or neural networks [158] can also be used on many entity resolution problems. Supervised techniques such as these typically require both positive and negative training examples.

While the class imbalance means it is generally very easy to find non-matches, if blocking, indexing or some other scalability improving technique is used and appropriately chosen, then these would normally need to be verified to check they are in fact non-matches. In order to automatically generate negative training data, we again make use of the one-to-one matching condition and take all the candidate pairs where (at least) one record in the pair was matched in the bootstrapping process, but the pair itself is not in $\mathbf{M}$. Formally, the set of non-matches $\mathbf{NM} = \{e_{ij} \in \mathbf{E} : \langle r_i, r_j \rangle \notin \mathbf{M} \wedge \exists r_k \in \mathbf{R}_1 \cup \mathbf{R}_2 \; s.t. \; (\langle r_i, r_k \rangle \in \mathbf{M} \vee \langle r_j, r_k \rangle \in \mathbf{M})\}$. In essence, if we determine that records r_i and r_j are a match, then the one-to-one condition means that each other candidate pair involving either r_i or r_j must be a non-match.

5.6 Summary

A lack of training data is a problem for entity resolution in general [23] and is particularly problematic for many advanced entity resolution techniques. Collective entity resolution techniques [12, 86] require bootstrapping to create the initial entity relationship graphs. Many other advanced entity resolution techniques such as Markov logic networks [150], group linkage techniques [54] and temporal entity resolution techniques [106] are fully supervised and require labelled training data (examples of matches and non-matches) in order to build the classification model. Getting such training and boostrapping data can be very difficult for many real-world

entity resolution problems and the unbalanced nature of the entity resolution problem (there are likely far more true non-matches than true matches) makes random selection of training records unfeasible.

However, many entity resolution problems contain domain constraints in the form of one-to-one or one-to-many matching restrictions. This can be because the data sets being matched do not contain duplicates, or it can be due to the nature of the domain itself, for example matching birth certificates to death certificates. In such cases, these constraints can be leveraged to assist in generating training data, by not only checking the similarity between the records in a candidate pair, but also checking whether there is any ambiguity, i.e. whether there are any alternative matching options for the records in a candidate pair.

In this chapter we proposed two approaches for generating bootstrapping and training data for entity resolution. Our approaches incorporate ambiguity into the decision making process thus allowing much greater confidence that selected record pairs are actually true matches. Our static approach is computationally fast (it has the same computational complexity as a simple threshold based classifier) and achieves very good results in most circumstances. For data sets where there is a high proportion of matching records, we have also developed an iterative approach, where records are processed in order based on their ambiguity and similarity values. As pairs are matched, the ambiguity values of remaining unmatched records are updated which can in turn lead to more matches being found. This iterative approach is computationally slower, due to the need to recalculate ambiguity values and update priorities, however it achieves improved recall in situations where a high proportion of records have a match.

We have also shown how ambiguity can be incorporated into active learning for entity resolution problems with one-to-one or one-to-many constraints. By selecting the most ambiguous records for manual classification and propogating the results, each manual classification can lead to multiple other matches becoming unambiguous, increasing the number of matches in the training data.

We have tested our approaches on multiple data sets and in comparison to three baseline techniques for generating training data. Our approaches achieve very high precision on all data sets. This allows the output of our approaches to be used to bootstrap or train a classification model with confidence that there are few or no false matches included in the training data. This is particularly important for collective entity resolution techniques where the impact of such incorrect matches can be widespread due to the results of initial match decisions being used to inform subsequent match decisions.

5.6.1 Future Work

In the future, there are several directions in which we hope to extend the work. We evaluate our approaches on the UKCD and NCVR data sets in this chapter, and we evaluate an extension of our approaches in Chapter 7, however the data sets are from a limited number of domains, so confirming the applicability of our approaches in other situations would further demonstrate their validity.

We also intend to investigate whether the idea of ambiguity can be incorporated into the entity resolution process along with attribute and relationship similarity. In our approaches, we use it as part of a training data generation process but assume that a further classification step will be run using a model that has been bootstrapped (or trained) on the results of our approaches. Incorporating ambiguity into the actual entity resolution process could be appropriate, particularly for classifiers that take an iterative approach, i.e. they resolve cases based on some ordering of increasing difficulty.

We also intend to investigate the optimal balance between precision and recall for different advanced entity resolution techniques. In our evaluation we have prioritised precision, since provided the training set is large enough, having false matches included in the training set is likely much more of a problem than a reduction in the overall number of matches. However, for a given problem and advanced entity resolution technique, there will be an optimal weighting of precision and recall, and calculating some examples of this will assist in setting parameters such as ψ_{min} and α_{max} on new problems, as well as fine tuning our overall approach.

A natural extension of ambiguity is to incorporate relationships and related entities into the definition. In Chapter 7 we propose an extension of our approach where we use relationships to validate our training data, and also to consider the ambiguity of groups of records.

Finally, as noted in Chapter 3, there are many similarities between one-to-one matching restrictions in entity resolution and the so called *stable marriage problem* [55]. We hope to investigate whether any of the algorithms used to solve the stable marriage problem and its variants, can be adapted to work in the entity resolution domain.

Chapter 6

Eliminating Parameter Settings through Unsupervised Evaluation

In Section 2.3 we described the entity resolution process, along with a some of the techniques that can be applied at each stage. While we described the process as being sequential, in practice it is often necessary to perform it iteratively as various techniques are tried, tuned, evaluated and discarded until the overall results are satisfactory. While this may be necessary from a practical perspective, each iteration of the entity resolution process can be very slow.

In this chapter we propose an unsupervised technique for evaluating entity resolution. Our technique relies on transitive closure to detect inconsistent entity resolution results prior to the classification step. This allows parameter settings and choices that will not produce good results to be discarded without spending resources on the classification and evaluation steps. While the primary purpose of our technique is to reject unsuitable parameter settings, it can also be employed as part of the overall evaluation to detect inconsistencies that need to be investigated further.

This chapter is structured as follows: in Section 6.1 we motivate the problem by presenting some examples and discussing problems that can occur. In Section 6.2 we present the main ideas behind our unsupervised evaluation technique, provide the notation and formal definitions, and the mathematical and algorithmic details of our approach. In Section 6.3 we perform a full experimental evaluation of our approach on the data sets described in Section 2.5. In Section 6.4 we discuss our results, as well as some additional aspects of our approach in more detail including computational complexity, normalisation and some limitations of the approach. Finally, in Section 6.5 we summarise our work and present some ideas of how it could be extended in the future.

6.1 Overview

While an iterative approach to entity resolution may be necessary, it leads to two problems in practice. Firstly, as discussed in Chapter 4, the scalability of entity resolution is generally poor. If each iteration takes a long time, then iterative parameter tuning can be extremely slow. The second problem, which we described in Chapter 5, is that for real-world entity resolution applications it is common that there is no training data available, and evaluating an entity resolution result is usually difficult and costly. The difficulty and cost of evaluation is made worse by an iterative approach, since the results of each iteration need to be evaluated separately.

6.1.1 Parameter Tuning

Each step in the entity resolution process typically requires many decisions, however we discuss the characteristics of two steps in more detail here.

The aim of the blocking and indexing step is to filter the record pairs, and remove as many pairs as possible, where there is little or no chance of them being true matches. As discussed in Chapters 2 and 3, numerous approaches for this have been proposed, which are effective in different situations and on different types of data. Each of these techniques require the choice of one or more attributes to use. Some attribute types also require additional processing such as truncation (selecting only some characters from an attribute value) or phonetic encoding [104, 124, 135]. Many techniques also require tuning of other parameters such as window size for sorted neighbourhood indexing [73], number and choice of hash functions for locality sensitive hashing based techniques [107], block size parameters for our own technique presented in Chapter 4, etc.

In the similarity calculation step, the aim is to compute a measure of similarity for each candidate pair. In practice this is done by applying one or more similarity functions to some or all of the attribute values of the two records. We described several similarity functions in Section 2.3, but there are many others. The entity resolution software package FEBRL [26] contains 21 different similarity functions, many of which have additional parameters of their own. It is also possible that customised similarity functions may be required (see Chapter 7 for a situation that required a custom similarity function).

While experience and domain expertise can assist in choosing techniques, blocking methods, similarity functions, parameters weights, and so forth, in practice it can be very difficult to select optimal (or even good) settings without significant experimentation and testing. When

		Classification Result	
		Match	Non-Match
Actual State	Match	9.5×10^5	5.0×10^4
	Non-Match	5.0×10^4	$10^{12} - 1.05 \times 10^6$

Table 6.1: Example classification matrix resulting from the comparison of two data sets of one million records each, with precision of 0.95 and recall of 0.95.

evaluation data is available, and on small data sets, this does not necessarily present a big problem, however for real-world applications this can be very time-consuming.

6.1.2 Evaluating Entity Resolution

We discussed the common entity resolution evaluation metrics in Section 2.3, and noted that three of the most widely used are precision, recall and the f-measure [23].

Calculating precision requires (at least an estimate of) the number of true matches and the number of false matches in the classification result. If each of the classified matches are evaluated manually by a domain expert, it may be possible to determine whether they are true matches or false matches, thus allowing precision to be calculated. If resources do not allow a manual analysis of all classified matches, a random sampling can be employed, which will lead to an estimate of precision, albeit with appropriate statistical confidence. It is also worth noting that beyond the problem of sampling error, experimental results of Smalheiser et al. [166], have shown that a manual evaluation is not guaranteed to be correct, which further affects the reliability of the precision estimate.

It is much harder to estimate recall due to the unbalanced classes (described in Section 2.3.1). In order to calculate recall, it is necessary to determine the number of true matches and false non-matches in the entity resolution result. As described above, it may be possible to estimate the number of true matches through a manual evaluation of the classified matches. However the false non-matches are much more difficult to estimate.

There are two situations which can lead to a false non-match in the entity resolution results, both of which are very difficult to detect. Firstly, the record pair may be classified as a non-match during the classification step. To understand why detecting these cases is a problem, assume two data sets are being compared where each consists of one million records and also assume that each record has a single true match in the other data set. If the classifier has precision of 0.95 and an actual (but unknown) recall of 0.95, then this results in the classification matrix shown in Table 6.1. Based on this classification matrix, the ratio of false non-matches to true non-matches

is approximately $1 : 2.0 \times 10^7$ or one to twenty million. This ratio means that it will almost certainly be impossible to manually evaluate enough randomly selected classified non-matches to get a statistically valid estimate of recall.

The common solution to this problem is to use a blocking or indexing technique, so that not all pairwise combinations of records are actually classified, thus making it easier to evaluate enough classified non-matches to estimate recall. However, this leads to the second situation where false non-matches can occur. Any pair of records that is not selected as a candidate pair by the blocking approach is treated as a non-match because it is never actually classified by the classifier. This means that in order to estimate recall, it is necessary to evaluate the pairs completeness (defined in Section 4.4) of the blocking approach, which again requires examining all record pairs from the two data sets.

6.1.3 Advanced Entity Resolution Techniques

In the context of advanced entity resolution techniques, both the scalability problem and the evaluation problem are even worse than for traditional entity resolution. The scalability of most advanced entity resolution techniques is worse than traditional entity resolution, meaning the classification step is likely to be slower. In addition, advanced entity resolution techniques often have more parameters and choices than traditional entity resolution, meaning more iterations are required. Finally, the types of data and problems where such advanced entity resolution techniques are appropriate, can make the evaluation very slow as well. Examining the context of each record to determine whether a record pair is a match or non-match requires an examination of all the related records and their circumstances, rather than just the record pair required for traditional entity resolution (see discussions in Section 7.1.2 about the creation of the ground truth data set used in Chapter 7).

What this means for entity resolution, and advanced entity resolution in particular, is that while an iterative approach to the entity resolution process may be neccessary, each iteration may be very slow, both in terms of computational time, and in terms of evaluation time. Any approach that can detect potential problems before the classification and evaluation steps, and thereby rule out various parameter settings, can significantly improve the speed of iteration and cut down on the time and cost of the entity resolution process. We present an approach for doing this in the remainder of the chapter.

R. ID	Surname	First Name	M. Init.	Date of Birth	Country of Birth	Postcode	Updated
r_1	Miller	Jane	S.	14-Feb-1984	United Kingdom	2600	2012
r_2	Miller	Jane	S.	14-Feb-1984	United Kingdom	2000	2013
r_3	Smith	Jane	S.	14-Feb-1984	United Kingdom	2000	2014
r_4	Stephens	Marie	P.	21-June-1977	New Zealand	2902	2012
r_5	Johnson	Marie	P.	21-June-1977	New Zealand	2001	2014

Table 6.2: Example of how transitivity can identify problems in the entity resolution process. The data set shows five records about two different people, both of whom have changed their surname and address.

6.2 Approach

In this chapter we present a technique to perform an unsupervised evaluation of the entity resolution process. The aim of our technique is to detect problems as early as possible in the process and without requiring a full evaluation. In order to do this we make use of the concept of transitive closure. Transitive closure (described in Section 2.2) reflects the idea that transitivity usually holds between matches. For example if records r_i and r_j are a match, and records r_j and r_k are also a match, then we must have that records r_i and r_k are a match. Since transitive closure usually holds in the ground truth (we discuss situations where it may not hold in Section 6.4), we can look for situations where transitive closure does not hold in the results (either results from intermediate steps or the final entity resolution output) and use this to tell us when we have problems in the blocking and indexing step or the similarity calculation step.

To demonstrate how this works, consider the data in Table 6.2. In it we show five records about two different entities (people). Both people have changed their surname and address over a two year period. The only difference is that the first person (records r_1, r_2 and r_3) updated their details after changing their address but before changing their name. Imagine the scenario where entity resolution is carried out using the following settings:

- Traditional blocking is performed twice. Once on the attribute *surname* and once on the attribute *postcode*. This results in each record being placed in two different blocks.
- Similarity calculation is done using the unweighted average of `exact match` applied to attributes *First Name*, *Surname*, *Date of Birth* and *Postcode*.
- Classification is done using a simple threshold based classifier with the similarity threshold set to 0.7.

The blocking approach means only records with either the same value of *surname* or the same value of *postcode* will be selected as candidate pairs. This means that r_1 and r_2 are a candidate

pair (since they both have the same value of *Surname* - 'Miller'), and that r_2 and r_3 are a candidate pair (since they both have the same value of *Postcode* - '2000'). However, records r_1 and r_3 are not a candidate pair since they do not match on either of the attributes used for blocking. Similarly, records r_4 and r_5 are not a candidate pair since they also do not match on any of the attributes used for blocking.

In the classification step, each of the candidate pairs produced by the blocking step will be classified by the classifier. This means $\langle r_1, r_2 \rangle$ will be classified as a match (the similarity is 0.75) and $\langle r_2, r_3 \rangle$ will be classified as a match (since the similarity is also 0.75).

The outcome of this entity resolution process does not reflect the ground truth for this example data set. The blocking step should have placed records r_1 and r_3 together in at least one block, and it should have also placed records r_4 and r_5 together in at least one block. The classification results are correct for all pairs actually classified by the classifier, since record pair $\langle r_1, r_2 \rangle$ is classified as a match, as is $\langle r_2, r_3 \rangle$, and no other comparisons are made.

Now consider a second (similar) scenario that uses *First Name* as a blocking attribute in place of *Surname*, and keeps all other settings the same.

For this scenario, r_1, r_2 and r_3 are all placed in the same block, since they have the same value of *First Name* - 'Jane'. Similarly, r_4 and r_5 will be placed in the same block, since they have the same value of *First Name* - 'Marie', thus each of record pairs $\langle r_1, r_2 \rangle$, $\langle r_1, r_3 \rangle$, $\langle r_2, r_3 \rangle$ and $\langle r_4, r_5 \rangle$ are candidate pairs. In the comparison step however, the similary value of $\langle r_1, r_2 \rangle$ is 0.75, the similarity value of $\langle r_2, r_3 \rangle$ is also 0.75, whereas the similarity of $\langle r_1, r_3 \rangle$ is 0.5 as is the similarity of $\langle r_4, r_5 \rangle$.

This means that again the output of the entity resolution process does not reflect the ground truth. Record pair $\langle r_1, r_2 \rangle$ is classified as a match (since the similarity of the two records is 0.75 which is greater than the 0.7 threshold). Record pair $\langle r_2, r_3 \rangle$ is also classified as a match for the same reasons. However, record pairs $\langle r_1, r_3 \rangle$ and $\langle r_4, r_5 \rangle$ are both classified as non-matches, since their similarity is only 0.5, which is less than the required threshold.

In both scenarios, transitive closure implicitly matches records r_1 and r_3 even though in the first scenario they did not occur in the same block together, and in the second scenario their similarity was below the required threshold. As such, we can tell that in the first scenario there must be problems with our blocking (since true matches are not being compared in the comparison step), and that in the second scenario there must be problems with our comparison or classification steps, since true matches are being classified as non-matches.

Short of performing an evaluation of every single record pair, there is little chance of de-

tecting the errors with records r_4 and r_5, and even with a such an evaluation, they may not be detected. However, the problems within the results for records r_1, r_2 and r_3 are much easier to detect, since we only have to look at the classified matches (in practice usually far fewer record pairs than the classified non-matches) and we can also focus on the cases where transitive closure does not hold, either in the blocking or the comparison and classification steps. Any parameter settings or choices leading to a high level of inconsistency in the results can be investigated or discarded, allowing us to focus on those options which may lead to good results.

It is worth noting however, that our technique doesn't determine what is wrong with the entity resolution result, only that *something* is wrong. The situation described in the first scenario indicates a matching pair is not included in the candidate pairs generated by the blocking and indexing step. The situation described in the second scenario indicates that at least one of the match decisions relating to the three records does not reflect the ground truth. If the number of such cases is relatively small, then investigating the instances where they occur can help to fine tune the blocking and indexing and similarity calculation steps. If the number of such cases is large, then this likely indicates the need to significantly revise at least one aspect of the entity process, for example choose different similarity functions, a different blocking approach, different attributes, etc.

6.2.1 Definitions

We assume a data set $\mathbf{R}$ made up of records $r_i \in \mathbf{R}$. Each record is associated with a set of attributes $\mathbf{A}$. We assume the traditional entity resolution process shown in Figure 2.1. We treat the blocking and indexing step and the similarity calculation step as black boxes. We assume the output of the blocking and indexing step is a set of (possibly overlapping) blocks $\mathbf{B}$, where each block $\mathbf{b}_n \in \mathbf{B}$ is a subset of $\mathbf{R}$. We assume the output of the similarity calculation step is a set $\mathbf{S}$, with each $s_{ij} \in \mathbf{S}$ being a triple $\langle i, j, \psi_{ij} \rangle$ where ψ_{ij} is the similarity between records r_i and r_j for each pair of records $r_i, r_j \in \mathbf{R}$ where there exists a $\mathbf{b}_n \in \mathbf{B}$ such that $r_i, r_j \in \mathbf{b}_n$.

Similarity values are calculated during the similarity calculation step using traditional string comparison functions (see Section 2.3.2) applied to one or more attributes in $\mathbf{A}$. Relational similarity can also be incorporated through the use of appropriate relational similarity measures (again see Section 2.3.2). We assume all similarity values are normalised, i.e. $\psi_{ij} \in [0, 1]$ for all $r_i, r_j \in \mathbf{R}$. If blocking or indexing has been applied then $\mathbf{S}$ may not be complete, i.e. for some (and in practice most) pairs of records $r_i, r_j \in \mathbf{R}, s_{ij} \notin \mathbf{S}$.

For advanced entity resolution techniques the classification step is generally computationally

expensive so we aim to detect potential problems with transitive closure prior to this stage. We use a similarity threshold ψ_{min} (where $0 \leq \psi_{min} \leq 1$) to determine likely matches, in essence a simple threshold based classifier (as described in Section 2.3.3). If we have existing knowledge about a suitable value for ψ_{min}, we can set the threshold accordingly, otherwise we can test many different values and assess the results of each (like we do in our evaluation in Section 6.3). For $r_i, r_j \in \mathbf{R}$, if $s_{ij} \in \mathbf{S}$ and $\psi_{ij} \geq \psi_{min}$ then we assume r_i and r_j have a high likelihood of being classified as a match during the classification step. In the simplest case this may be the actual classification model used as well [47].

We construct a graph $\mathbf{G}$ where each node $n_i \in \mathbf{G}$ corresponds to a record $r_i \in \mathbf{R}$, and an edge $\langle r_i, r_j \rangle$ in $\mathbf{G}$ corresponds to the similarity value $s_{ij} \in \mathbf{S}$, with the edge weight being the value of ψ_{ij}. For a given ψ_{min}, we also define a sub-graph of $\mathbf{G}$, $\mathbf{G}^+ = \{\langle r_i, r_j \rangle \in \mathbf{G} : \psi_{ij} \geq \psi_{min}\}$. In essence, $\mathbf{G}^+$ corresponds to the graph of likely matches in $\mathbf{R}$.

Finally, we define three types of triangles (shown in Figure 6.1) that exist on the graph $\mathbf{G}^+$ with similarity threshold ψ_{min}:

- A *consistent triangle* - denoted Δ_{co} - is a triple of records $\langle r_i, r_j, r_k \rangle$ such that $r_i, r_j, r_k \in \mathbf{R}$, $s_{ij}, s_{ik}, s_{kj} \in \mathbf{S}$ and $\psi_{ij} \geq \psi_{min}$, $\psi_{ik} \geq \psi_{min}$ and $\psi_{kj} \geq \psi_{min}$.
- An *incomplete triangle* - denoted Δ_{ip} - is a triple of records $\langle r_i, r_j, r_k \rangle$ such that $r_i, r_j, r_k \in \mathbf{R}$, $s_{ij}, s_{jk} \in \mathbf{S}$ but $s_{ik} \notin \mathbf{S}$ and $\psi_{ij} \geq \psi_{min}$ and $\psi_{jk} \geq \psi_{min}$. Note that because $s_{ik} \notin \mathbf{S}$, ψ_{ik} is never calculated.
- An *inconsistent triangle* - denoted Δ_{ic} - is a triple of records $\langle r_i, r_j, r_k \rangle$ such that $r_i, r_j, r_k \in \mathbf{R}$, $s_{ij}, s_{ik}, s_{jk} \in \mathbf{S}$, $\psi_{ij} \geq \psi_{min}$, $\psi_{jk} \geq \psi_{min}$ but $\psi_{ik} < \psi_{min}$.

In the next section we describe how the proportions of the different triangle types in the entity resolution result can give us insight into where we may have problems in the process, without relying on ground truth or a full evaluation.

6.2.2 Triangle Counting Approach

The basic idea of our approach is to perform simple threshold based entity resolution and then look at situations where transitive closure does not hold in the final result in order to detect problems. In order to assess transitive closure, we make use of the three triangle types described in the previous section: *consistent triangles*, *incomplete triangles* and *inconsistent triangles*. The three different triangle types are shown in Figure 6.1. Assuming that transitive closure holds in the

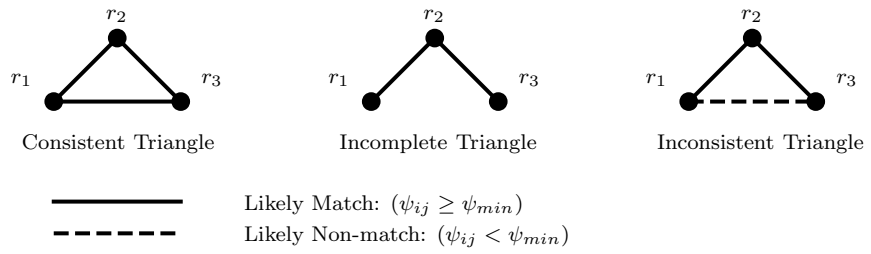

Figure 6.1: Three triangle types for evaluating entity resolution.

Algorithm 6.1: Unsupervised evaluation of entity resolution

Input:
- A set of records $r_i \in \mathbf{R}$
- A set of blocks $\mathbf{b}_k \in \mathbf{B}$
- A set of similarity values $\langle r_i, r_j, \psi_{ij} \rangle \in \mathbf{S}$
- A minimum similarity threshold ψ_{min};

Output:
- Normalised proportions of Consistent (CO), Incomplete (IP) and Inconsistent triangles (IS).

1: **G** = ConstructGraph(**R**, **S**)
2: $\mathbf{G}^+$ = GetSubGraph(**G**, ψ_{min}) // Limit to likely matches
3: **C** = BreadthFirstSearch($\mathbf{G}^+$) // Get connected components
4: `foreach` c_i `in` **C** `do`:
5: Δ_i^{co}, Δ_i^{ip}, Δ_i^{is} = CountTriangleTypes(c_i) // Label each component with the triangle counts
6: CO, IP, IS = NormaliseCounts(**C**)
7: return CO, IP, IS

ground truth, then for the output of the entity resolution process to reflect the ground truth, it should only contain consistent triangles. If the output contains a high proportion of incomplete triangles, then there are problems with the blocking process, while if the output contains a high proportion of inconsistent triangles, then there are problems with the similarity calculations.

Our approach is detailed in Algorithm 6.1. We start by constructing **G**, based on the set of records **R**, the set of similarity values **S**, and the specified similarity threshold ψ_{min} (line 1). We next construct the sub-graph $\mathbf{G}^+$, which only contains edges $\langle r_i, r_j \rangle$ where $s_{ij} \in \mathbf{S}$ and $\psi_{ij} \geq \psi_{min}$ (line 2). We then use a breadth-first search to find the connected components of $\mathbf{G}^+$ (line 3). For each component in $\mathbf{G}^+$, we count the number of consistent (Δ_{co}), incomplete (Δ_{ip}), and inconsistent (Δ_{ic}) triangles in **G** (line 5). These counts are then normalised for each component (line 6).

Data set	Approach	Type	Attributes
UKCD	1	Traditional Blocking	*First Name + Birth Parish*
UKCD	2	Traditional Blocking	*Gender + Surname*
UKCD	3	Traditional Blocking	*Gender + Birth Year*
NCVR	1	Sorted Neighbourhood Indexing	*First Name*
NCVR	2	Sorted Neighbourhood Indexing	*Surname + First Name + Zip Code*

Table 6.3: The blocking approaches used in our evaluation.

Data set	Approach	Similarity Functions
CORA	1	⟨*Author*, `qgram`, 0.2⟩, ⟨*Title*, `qgram`, 0.4⟩, ⟨*Publication*, `qgram`, 0.2⟩, ⟨*Year*, `qgram`, 0.2⟩
CORA	2	⟨*Title*, `qgram`, 1.0⟩
CORA	3	⟨*Author*, `qgram`, 0.5⟩, ⟨*Title*, `qgram`, 0.5⟩
CORA	4	⟨*Publication*, `qgram`, 0.5⟩, ⟨*Year*, `qgram`, 0.5⟩
UKCD	1	⟨*Surname*, `qgram`, 0.3⟩, ⟨*First Name*, `qgram`, 0.3⟩, ⟨*Occupation*, `qgram`, 0.15⟩, ⟨*Birth Year*, `age_difference`, 0.25⟩
NCVR	1	⟨*First Name*, `Jaro`, 1.0⟩
NCVR	2	⟨*First Name*, `Jaro`, 0.25⟩, ⟨*Surname*, `Jaro`, 0.25⟩, ⟨*Age*, `age_difference`, 0.25⟩, ⟨*Zip Code*, `qgram`, 0.25⟩

Table 6.4: The combinations of ⟨*attribute*, `similarity function`, weighting⟩ triplets used in our evaluation.

We discuss normalisation further in Section 6.4. However, the standard approach we use for normalising is as follows: given the set of connected components $\mathbf{C}$, for each component $c_i \in \mathbf{C}$, let Δ_i^{co}, Δ_i^{ip} and Δ_i^{is} be the numbers of consistent, incomplete and inconsistent triangles, respectively, let $|c_i|$ be the number of nodes in c_i and let $|e_i|$ be the number of edges in c_i. Let $\mathbf{C_3} = \{c_i \in \mathbf{C} : |c_i| \geq 3\}$. The normalised proportions of each triangle type are calculated as follows:

$$\text{Consistent triangles (CO)} = \frac{1}{p} \sum_{c_i \in \mathbf{C_3}} \frac{\Delta_i^{co}}{|e_i|} \tag{6.1}$$

$$\text{Incomplete triangles (IP)} = \frac{1}{p} \sum_{c_i \in \mathbf{C_3}} \frac{\Delta_i^{ip}}{|e_i|} \tag{6.2}$$

$$\text{Inconsistent triangles (IS)} = \frac{1}{p} \sum_{c_i \in \mathbf{C_3}} \frac{\Delta_i^{is}}{|e_i|} \tag{6.3}$$

where p is a scaling factor to ensure that CO + IP + IS = 1. We note that components with 1 or 2 nodes do not appear in the results, since they do not contain any triangles.

6.3 Evaluation

We evaluated our technique on each of the three data sets described in Section 2.5. Since CORA is a very small data set (1295 records) we have used CORA to show the impact of different similarity functions without blocking or indexing being applied. For the UKCD data sets, we used multiple blocking methods but kept the similarity functions unchanged. For the NCVR data set, we tested the combinations of two different indexing parameters and two different similarity functions. We also note that some settings are deliberately chosen in such a way as to be unsuitable in order to demonstrate the effect this has on the results. The blocking or indexing

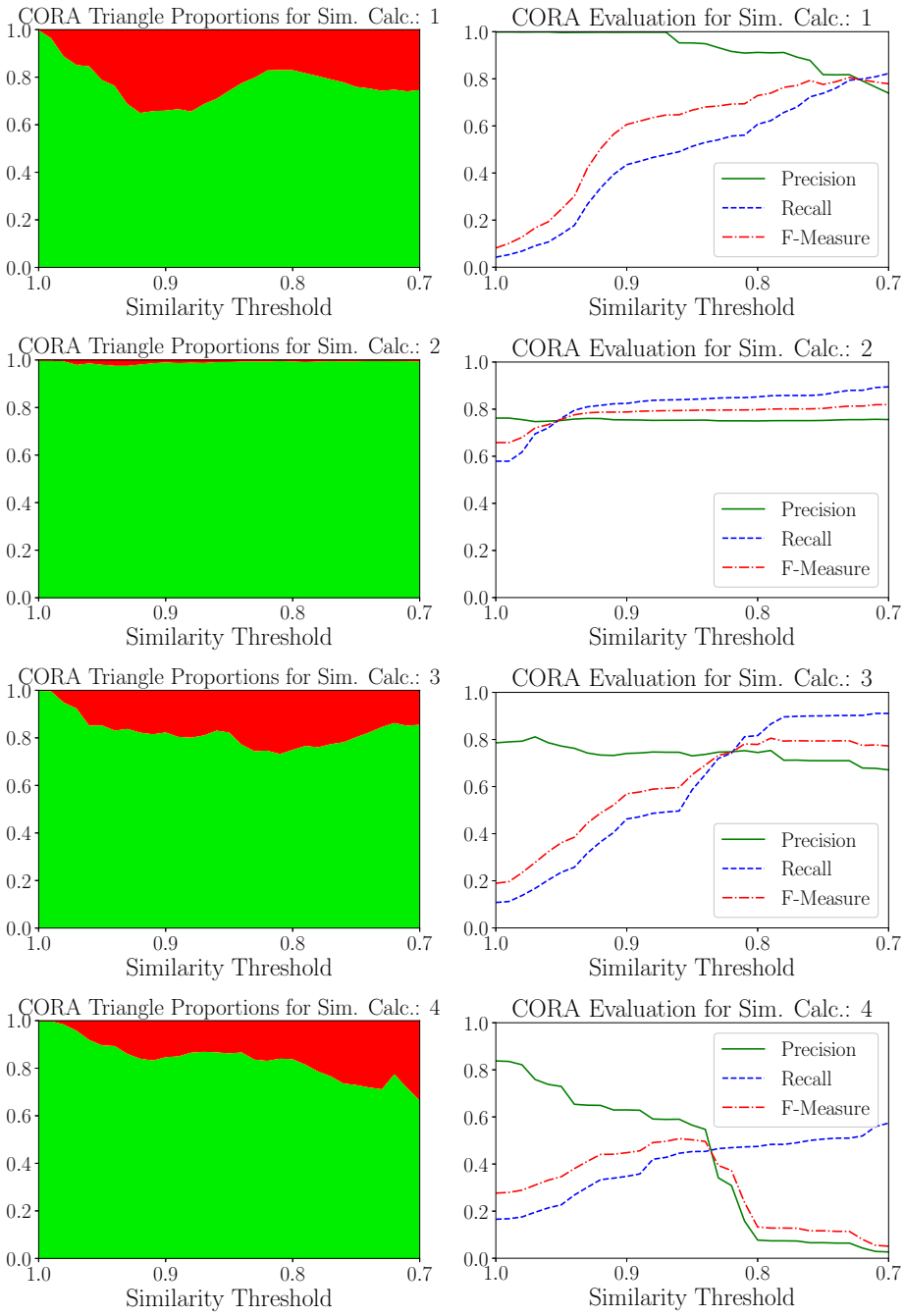

Figure 6.2: Normalised proportions of *Consistent Triangles* (green), *Incomplete Triangles* (yellow) and *Inconsistent Triangles* (red), as well as the corresponding values of precision, recall and the f-measure for the CORA data set.

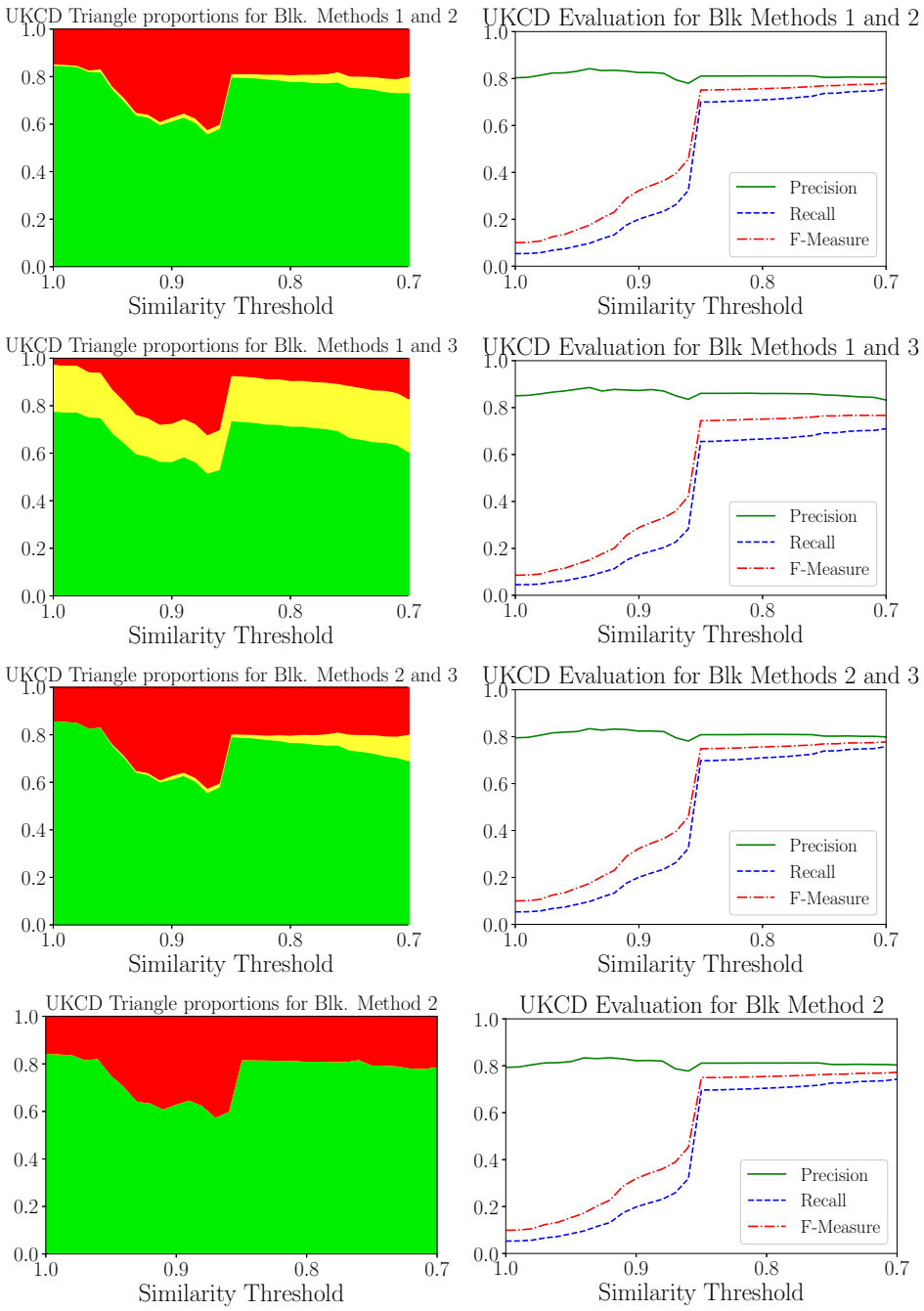

Figure 6.3: Normalised proportions of *Consistent Triangles* (green), *Incomplete Triangles* (yellow) and *Inconsistent Triangles* (red), as well as the corresponding values of precision, recall and the f-measure for the UKCD data set.

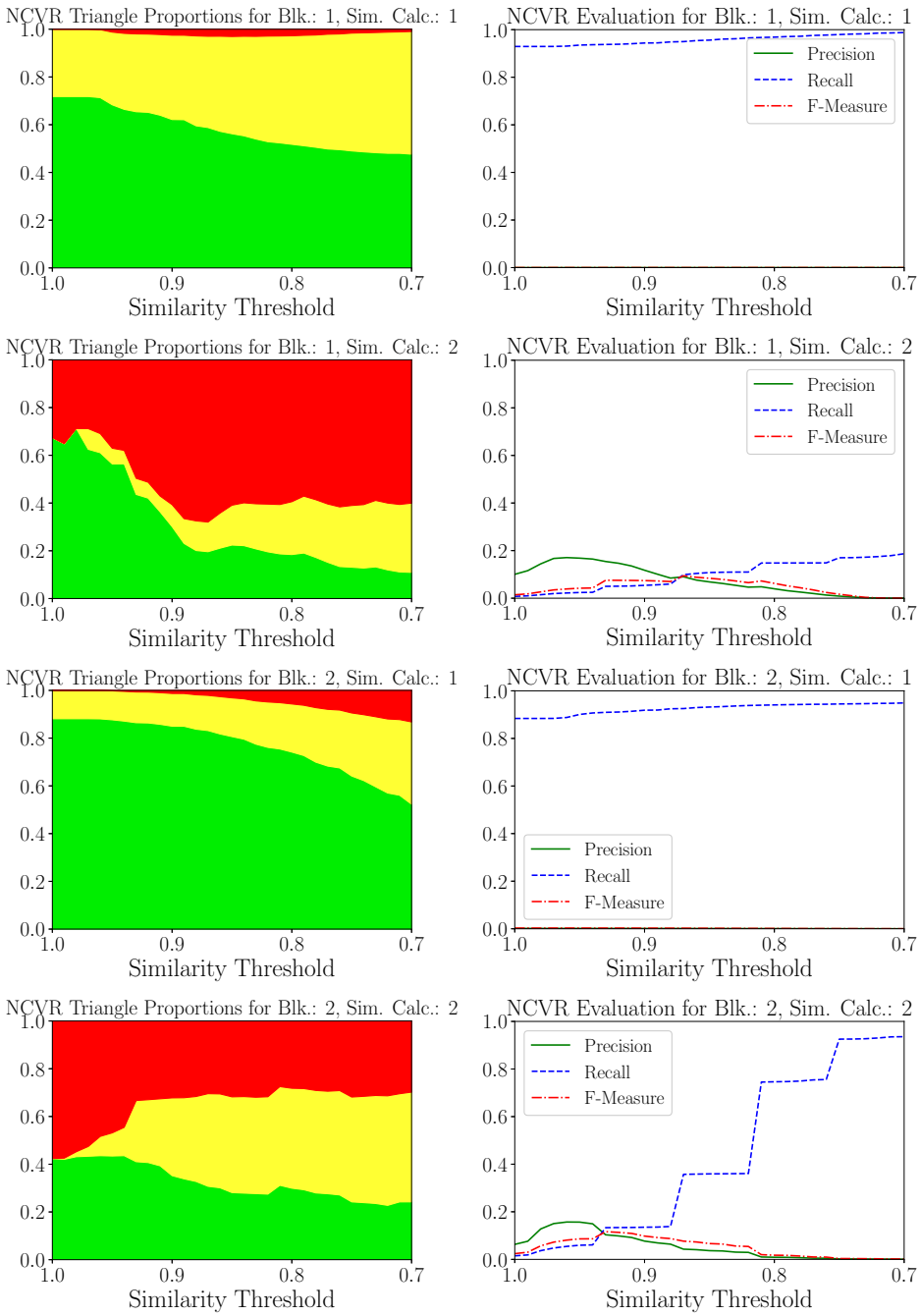

Figure 6.4: Normalised proportions of *Consistent Triangles* (green), *Incomplete Triangles* (yellow) and *Inconsistent Triangles* (red), as well as the corresponding values of precision, recall and the f-measure for the NCVR (full) data set.

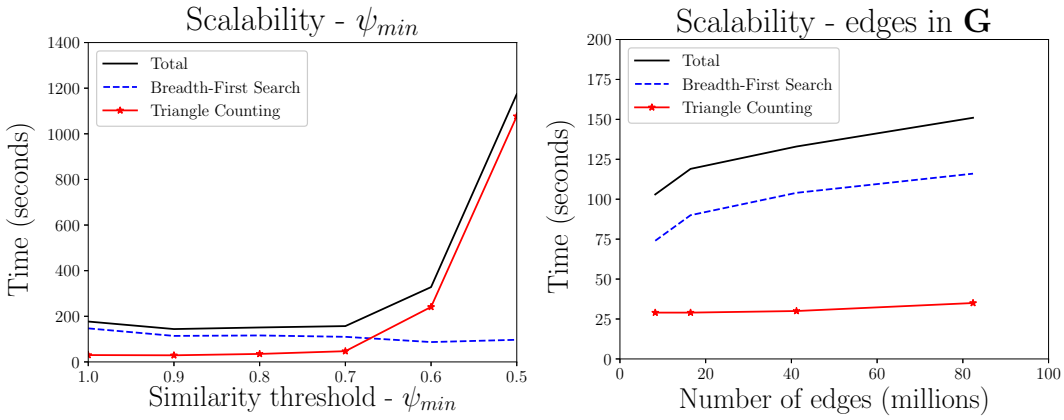

Figure 6.5: Scalability - evaluated on the NCVR data set, showing the impact of ψ_{min} and the number of edges in **G** on the running time.

approaches are shown in Table 6.3 and the similarity functions are shown in Table 6.4 - where they are described as a triplet of $\langle attribute, similarityfunction, weighting \rangle$.

For our evaluation, we present the normalised proportions of consistent, incomplete and inconsistent triangles for different values of ψ_{min}, and for different combinations of blocking parameters and similarity calculations. The results are shown in Figures 6.2, 6.3 and 6.4. We also show the corresponding values of precision, recall and the f-measure (described in Section 2.3).

In addition, we evaluated the scalability of our approach using the NCVR data set. We examined the impact of the density of the graph by changing the threshold ψ_{min}, shown in Figure 6.5, and also the number of edges in **G**. Increasing the number of edges in **G** resulted in a small increase in running time. For most settings of ψ_{min} the algorithm was efficient, but for values of ψ_{min} below a certain point (in this case 0.7), the presence of very large components slows the algorithm down significantly. However, the algorithm is efficient for the values of ψ_{min} that are of practical interest (typically > 0.7). We discuss this further in the next section.

Finally, we make two notes about the experimental setup. Firstly, since transitive closure does not apply to the NCVR-450 data set, (see discussion in the next section,) for the evaluation we instead made use of the full NCVR data set.

Secondly, for the UKCD data set, there is a one-to-one matching constraint between records in each census period (i.e. a record from the 1851 census can match to at most one record from the 1861 census and so forth). However, because there are six census periods in the data set (every ten years between 1851 and 1901), transitive closure is still applicable provided the three records are all from different census periods. Incorporating this required a minor modification

to the `GetSubGraph` function called in line two of Algorithm 6.1 in order to ensure the one-to-one matching constraints were not violated (either directly or through transitive closure). As such, when constructing, $\mathbf{G}^+$, we adopted a greedy approach starting with the edges of $\mathbf{G}$ with the highest value of ψ and only adding them to $\mathbf{G}^+$ if they would not violate the one-to-one matching constraint. Ties were resolved arbitrarily (i.e. based on a randomly assigned record id).

6.4 Discussion

As with most aspects of the entity resolution problem, there is no one-size-fits-all rule such as 'consistent triangle proportions above a certain threshold are guaranteed to produce the best results.' However, the proportions of the different triangle types provide another tool for evaluating entity resolution results and we can eliminate several of the results presented above very quickly. We can also make some additional judgements to help refine the entity resolution process going forwards.

CORA: Each of results 1, 2 and 3 are potentially worth investigating further. We make a note that for number 2, the number of matches only increases from approximately $12,000$ to approximately $20,000$ as ψ_{min} decreases from 1.0 to 0.7, and from approximately $18,500$ to approximately $20,000$ between ψ_{min} values of 0.9 and 0.7. This suggests there may be a tipping point up to which the majority of triagles are consistent, and it may be worth experimenting with lower values of ψ_{min} to determine where this point occurs.

The results of number 4 are problematic, since it is not immediately obvious that they are significantly worse than numbers 1, 2 and 3. However, for values of $\psi_{min} \leq 0.72$, a single large component includes over half the records in the data set. This suggests that even though the output is reasonably consistent, it is likely that is has matched far too many records to be of practical value.

UKCD: The most obvious point with respect to the UKCD results is the high proportion of inconsistent triangles when blocking techniques 1 and 3 are combined. This suggests that these two blocking approaches do not significantly overlap. Furthermore, when each of them is combined with blocking approach 2, there is a very low proportion of inconsistent triangles for most values of ψ_{min}. This suggests that blocking approach 2 is capturing almost all the matches that are included in 1 and 3, which is borne out by the final set of results (which was run subsequently to the first three), where only blocking approach 2 is used. Going forward,

this suggests that blocking approaches 1 and 3 can be discarded, since they do not add anything that is not already captured by blocking approach 2.

Another observation about the results from the UKCD data set relates to the significant rise in consistency for values of ψ_{min} in the range 0.9 to 0.85. This coincides with the weighting of 0.15 associated with the *occupation* attribute in the similarity calculations. It suggests that there are a large number of records which are identical except for their values in this attribute. This means that for the classification step, it is very important to determine whether such records should be a match or not, since it will have a significant impact on the results.

NCVR: The main observation is that even at values of ψ_{min} close to 1.0 there are large proportions of incomplete triangles. This suggests that there are many clusters forming that are larger than the window size of the sorted neighbourhood indexing (20 records for each approach). If these large clusters are realistic with respect to the problem then the window size needs to be increased to make sure that all such record pairs are actually compared. Otherwise, it suggests that the similarity functions are inadequate to solve the particular problem. In the case of the NCVR data set it is most likely to be the latter, since the size of the data set and the frequency of the most common values of each of the reliable attributes *First Name*, *Surname* and *Age* are more likely to produce a false match than a true match, even when they are identical. This suggests that capturing some measure of relational similarity or performing group linkage on households may be required to achieve good results.

It is also worth making a note about the high proportion of inconsistent triangles in the second and fourth rows of Figure 6.4. This is due to the use of the `age_difference` function (which we define in Chapter 7). Because age changes, for all data sets we evaluate, we convert the *Age* attribute to *Year of Birth* prior to comparison. However, this means that the value of *Year of Birth* can be incorrect by one year (since we do not know whether the record comes from before or after the individual's birthday). To account for this, the `age_difference` function returns similarity of 1.0 even for a difference in *Year of Birth* of one year. However, this leads to the situation where the results are inconsistent - even when ψ_{min} is set to 1.0, since records are being matched with other records both one year older and one year younger than themselves, and which are identical in the other attributes involved in the similarity calculation. More sophisticated matching techniques are required to resolve these cases.

6.4.1 Complexity Analysis and Scalability

We briefly analyse the computational complexity of our approach. Breadth-first search is $O(|\mathbf{R}| + |\mathbf{S}|)$ and triangle counting is $O(|\mathbf{S}|^{\frac{3}{2}})$ [171]. In practice triangle counting is dependent on the degree of nodes in the graph (both average and worst-case degree). This matches our practical evaluation in our Section 6.3, where the running time is shown to be very dependent on the choice of ψ_{min}, (i.e. how many edges are in $\mathbf{G}^{+}$). As such, while a superlinear complexity is not ideal, for most values of ψ_{min} that are of practical use (i.e. $\psi_{min} \geq 0.7$), the algorithm scales very well.

6.4.2 Normalisation

Since a triangle is formed by three points, we observe that the maximum number of triangles in a component c_i is $\binom{|c_i|}{3}$. This means that the number of triangles in a component is $O(|c_i|^3)$. As such, without normalisation a single large component can completely dominate the results. While the presence of a large component in and of itself may be a sign of problems (since in most problem domains it is rare to have hundreds or thousands of records that refer to the same entity in a data set), it can still lead to problems. A component with ten nodes potentially contains 120 times as many triangles as a component with three nodes.

We considered two possible methods of normalisation. The first, and the one presented in the results, is to make the contribution of each component proportional to its size. As described in Section 6.2, this can be done by dividing the contribution of each component by the number of edges it contains (which is $O(|c_i|^2)$) to end up with a weighting that is proportional to the number of nodes in the component, i.e. $O(|c_i|)$.

However, an alternative method of normalisation is to have each component carry the same weighting regardless of size. This can be achieved by dividing the contribution of each component by the total number of triangles in it. The normalisation factors shown in Equations 6.1, 6.2 and 6.3 instead become:

$$\text{Consistent triangles (CO)} = \frac{1}{|\mathbf{C_3}|} \sum_{c_i \in \mathbf{C_3}} \frac{\Delta_i^{co}}{\Delta_i^{co} + \Delta_i^{ip} + \Delta_i^{is}} \tag{6.4}$$

$$\text{Incomplete triangles (IP)} = \frac{1}{|\mathbf{C_3}|} \sum_{c_i \in \mathbf{C_3}} \frac{\Delta_i^{ip}}{\Delta_i^{co} + \Delta_i^{ip} + \Delta_i^{is}} \tag{6.5}$$

Household ID	Entity IDs
h_1	e_1, e_2, e_3, e_4
h_2	e_1, e_2, e_3, e_5
h_3	e_1, e_2, e_5, e_6

Table 6.5: An example of three households where transitive closure doesn't apply for household matches in the ground truth, based on a match definition of 75% or greater overlap in the entities in the two households.

$$\text{Inconsistent triangles (IS)} = \frac{1}{|\mathbf{C_3}|} \sum_{c_i \in \mathbf{C_3}} \frac{\Delta_i^{is}}{\Delta_i^{co} + \Delta_i^{ip} + \Delta_i^{is}} \tag{6.6}$$

where $|\mathbf{C_3}|$ is the number of components in $\mathbf{C_3}$.

6.4.3 Limitations

There are some limitations to our approach and we briefly discuss some situations where it is not appropriate.

Firstly, for some entity resolution problems, transitive closure does not apply. In some cases this is due to the presence of one-to-one matching constraints, which we defined in Section 2.2 and discussed in more detail in Chapter 5. If there are only two data sets to be matched and a one-to-one matching constraint applies to all records, then there is no possibility of a single record being part of multiple true matches, so transitive closure cannot apply.

As discussed in Section 2.2, there are also some situations where transitive closure may not hold in the ground truth. Consider the three households shown in Table 6.5. If the definition of a match for two households is that at least 75% of the individual entities (in this case people) in each household are also in the other household, then transitive closure does not hold in the ground truth when matching these three households. Household h_1 matches to household h_2, since they both contain entities e_1, e_2 and e_3, and household h_2 matches household h_3, since they both contain entities e_1, e_2 and e_5, but household h_1 does not match household h_3, since the only common entities are e_1 and e_2, which is a 50% overlap and does not meet the required threshold of 75%. For entity resolution problems like this, we cannot assume that a high level of inconsistency in the results means there are problems in the entity resolution process.

Finally, while a high level of inconsistency in the results implies problems with the entity resolution process, the reverse is not true, i.e. a low level of inconsistency does not always imply a lack of problems. For example, setting ψ_{min} to 1.01 will result in no triangles being formed (since $\mathbf{G}^+$ has no edges), which is a perfectly consistent result that is likely of no practical

value. Similarly, in the absence of blockingand indexing, setting ψ_{min} to 0.00 will result in $\mathbf{G}^{+}$ being a complete graph, so all the traingles are consistent triangles. This again is a perfectly consistent result which is likely of no practical use. While these are extreme cases, less extreme examples are easy to construct and we show some in Section 6.3 along with a discussion of how to potentially identify them. Nevertheless, we recommend using our approach to test and validate choices in conjunction with traditional expertise.

6.4.4 Incorporating Into Advanced Entity Resolution Techniques

Our approach uses a simple threshold based classifier ψ_{min} as a means of determining likely matches. This is a reasonable first approximation since all the advanced entity resolution techniques described in Chapter 3 make use of similarity in some way. However, those techniques incorporate additional features which go beyond a simple threshold based classifier, and the output of such classification steps may not mirror the results we obtain.

Nevertheless, there are a number of reasons why it is important that the blocking and similarity calculation steps are well chosen, even when used as part of more advanced entity resolution techniques. From a computational perspective, resolving inconsistencies in transitive closure using techniques such as Markov logic networks [165] is extremely expensive computationally. For other techniques such as the random walk based collective entity resolution technique of Kalashnikov and Mehrotra [86], inconsistencies in the output from the initial stage increase the number of choice nodes in the graph which increases the running time of the technique. From a blocking perspective, if each block is processed independantly, then any deficiencies in the blocking may never be corrected in the classification step. In addition, techniques such as the group linkage technique of Fu et al. [54] rely on a threshold based classifier to provide the potential entity-to-entity matches, before trying to classify the groups, so any problems in the initial stage will likely impact the final results.

Finally, when using our approach in conjuction with advanced entity resolution techniques, our approach can also be applied to the output of the classification step where classified matches are used in place of similarity values - thus removing the need of ψ_{min}. While this has the downside of requiring the (potentially expensive) classification step be run before problems are detected, many of the same general rules apply in that a high level of inconsistency in the final results, likely indicates something has gone wrong in the process. As such it gives us another tool to evaluate an entity resolution result without resorting to a full manual evaluation.

6.5 Summary

Entity resolution is often performed in an iterative fashion. Techniques are chosen, parameters set, results are obtained and evaluated, and then the whole process is repeated until the final outcome is good enough for whatever the task is. While this may be necessary from a practical perspective, the process can be computationally very slow. In addition, if the process is performed without evaluation data, a common scenario for real-world entity resolution problems, then a manual evaluation to determine whether the results obtained are good enough can be very time consuming, and thus expensive.

While this situation holds true for traditional entity resolution, it can be even worse for many advanced entity resolution techniques where the classification step of the entity resolution process can be extremely time-consuming. Evaluating the results for complex entity resolution problems involving groups of entities or multiple entity and relationship types can also require examining the context of records which further increases the effort and cost associated with the evaluation for each iteration.

In this chapter we presented a technique based on transitive closure to detect entity resolution results with a high degree of inconsistency in either the blocking results or the similarity calculation results. This technique can be run prior to the classification step in order to detect potential problems and eliminate parameter settings or choice of techniques that will not produce consistent results. By examining the proportions of different types of triangles in the results graph, we can detect high levels of inconsistency. Where a pair of records are being implicitly matched through transitive closure but were never compared in the similarity calculation step, there must be a problem with the blocking or indexing approach used. Where they were compared but were below the required similarity threshold, there must be a problem with the similarity calculation. If either situation occurs frequently in the entity resolution results then this can be quickly detected and the parameters or techniques changed until a satisfactory level of consistency is obtained.

Finally, we reiterate that although a high level of inconsistency indicates problems with the entity resolution process, a consistent output is not a guarantee of good results. A fully consistent result can be obtained by matching every pair of records or by matching no pairs of records, however neither result is likely to be of value. We discuss some less extreme examples in the evaluation along with potential ways of detecting such situations, but still suggest that our approach should be used to test and validate choices in conjunction with traditional expertise.

6.5.1 Future Work

While the approach for evaluating entity resolution in an unsupervised fashion is effective, there are a number of possibilities for extending the work or to look at similar types of measures as a way to minimise the manual evaluation effort that is required.

In our current approach, the measure ψ_{ij} is treated as a single value, where for those situations that are producing inconsistent results, it might be useful to see the breakdown across different attributes and similarity functions. This would make it easier to detect situations like the one present in the UKCD experiments, where the *Occupation* attribute was causing problems in the similarity calculation step. The same approach could be applied to the blocking and indexing step, where investigating which combination of techniques was producing the largest proportion of the incomplete triangles would allow for a better assessment of why the problems were occuring.

Just as it would be possible to separate ψ_{ij} into different components based on the attributes used, so it would also be possible to have different values of ψ_{min} in different parts of a graph. Along similar lines, the entity resolution problem can be reframed as a community detection problem in graphs [51], and while many of the algorithms for this are computationally expensive, there are potentially approaches from this area that could be adapted to our problem.

While our approach was effective at detecting a high degree of inconsistency, as we mentioned in Section 6.4, consistent results are not guaranteed to be useful. In the extreme cases this is relatively easy to detect, however, we found that for the less obvious cases two additional pieces of information were very effective in spotting problems, namely the total number of matches and the size of the largest cluster. While they can be evaluated separately as we did in our experimental evaluation, we intend to investigate whether they can be incorporated into the measures, potentially through the normalisation process. This would maintain the relatively simple output of our approach while also allowing us to identify less obvious problems.

Chapter 7

Generating Relational Training Data - A Case Study on the Isle of Skye

As we discussed in Chapter 1, a significant limitation of many advanced entity resolution techniques is that they require training data or a boostrapping process. In many real-world entity resolution applications, such training data is unavailable and it can be expensive and time consuming to produce. In Chapter 5, we presented an approach that made use of one-to-one and one-to-many matching constraints, as well as the concept of ambiguity, to produce training data consisting of similar and unambiguous record pairs. We showed that this approach is very effective at producing high quality training data (our precision results were all very close to 1.0) and worked better than the baseline approaches. However, as discussed in Chapter 3, many advanced entity resolution techniques also make use of relationship information. Collective entity resolution techniques [12, 39, 86, 165], group linkage techniques [34, 53, 54, 126] and population reconstruction techniques [29, 42, 157] all use relationship information as part of the entity resolution process. This means that the problem domains where advanced entity resolution techniques such as these are appropriate, also provide additional features that can be used as part of the training data generation process. In addition, many of these problem domains are very challenging, and the additional features may be required to achieve good results.

In this chapter, we extend our training data generation technique from Chapter 5 to incorporate relationship information. We use the relationship information to validate the matches produced by our approach, and we also extend the concept of ambiguity to groups of records, allowing us to add matches to the training set which may be individually ambiguous, but where the group as a whole0 has only one candidate group for matching.

To demonstrate the extension of our technique from Chapter 5, we present a case study using nineteenth century historical registry data from the Isle of Skye in Scotland, United Kingdom.

As we discussed in Chapter 3, population reconstruction techniques link registry data such as birth, death and marriage certificates, as well as census data [157]. Where available, other official or unofficial records can also be incorporated including legal documents such as land title, loan documents and wills [42], court or other criminal records [113], parish registers [174], and even private correspondance or other writings [61]. By linking such records, it is possible to build up details of major life events for individuals and family trees. Increasingly, governments and other organisations around the world are becoming interested in creating such linked data sets (both historical and contemporary) and have sponsored projects or setup units to create them[1]. The output of these projects provides a valuable research resource in fields such as history [113], social sciences [175] and medicine [63, 75].

The data sets used in this case study form part of the Digitising Scotland project[2]. While the population reconstruction aspect of the project only considers census records and birth, death and marriage certificates, the full scope of the project is broader than this and includes codifying occupation and cause of death details as well as incorporating address data.

The remainder of the chapter is structured as follows. In Section 7.1 we describe the background to the problem, the data sets involved, and the challenging nature of the problem. In Section 7.2 we show why current approaches to generating training data are unlikely to be successful for this problem, and describe the adaptations we made to our approach from Chapter 5. In Section 7.3 we describe our experimental setup, our results, and the results of the baseline approaches. In Section 7.4, we discuss our experimental results, how they performed for each matching case, and other observations we made while evaluating the performance of our technique. Finally, in Section 7.5 we provide a brief summary of our work in this chapter.

7.1 Overview

The Isle of Skye is a large island off the West coast of the Scottish mainland in the United Kingdom. It is approximately 80 kilometers long and 40 kilometers wide.[3] The island nature of the Isle of Skye makes it relatively isolated, and this was particularly true in the nineteenth century. This geographic isolation and relatively stable population has meant that the Isle of Skye has been used for research case studies in diverse areas including infant mortality [59, 149], linguistics [167], ecology [14] and others.

[1] https://www.ipdln.org/data-linkage-centres
[2] https://digitisingscotland.ac.uk/
[3] https://www.google.com/maps/place/Skye/@57.2341925,-7.0039461,8z

Data Set	Number of Records	First Record	Last Record	Unlinked Records
Births	17,614	1861	1901	1627
Deaths	12,285	1861	1901	2267
Marriages	2,668	1861	1901	857[a]
Census 1861	19,605	1861	1861	6917
Census 1871	18,102	1871	1871	2258
Census 1881	17,684	1881	1881	1911
Census 1891	16,476	1891	1891	1788
Census 1901	14,609	1901	1901	3155

Table 7.1: The different components of the Isle of Skye data set.

[a]This figure is the number of bride and groom records that are not linked, so a single marriage certificate could be counted twice if neither is linked.

This geographic isolation also makes the Isle of Skye particularly useful as a test case for population reconstruction. A major challenge for population reconstruction is individuals moving, either to another area, which makes them much harder to reliably link, or outside the scope of the data set altogether [31]. In the case of the Isle of Skye during the nineteenth century, many individuals were born, lived their entire lives, and died on the island, allowing a complete timeline of important life events to be constructed for individuals [69].

7.1.1 Isle of Skye Data Sets

The data set for this case study is split into eight separate files, and basic statistics about each are shown in Table 7.1. There is one file each for birth, death and marriage records, as well as five census files, one for each of the years 1861, 1871, 1881, 1891 and 1901.

In addition, there is a quasi-ground truth data set containing records that have been manually linked by historians [148]. This ground truth data set contains 54,545 individual records, representing life events that the historians have determined are about the same individual. Each record contains record ids from the eight other data sets. Every birth, death and census record appears once in the ground truth data, and every marriage record appears twice in the ground truth data, once for the bride and once for the groom.

In all cases, the data sets are dirty. There are many missing or abbreviated values, even in seemingly critical attributes like *First Name* and *Surname*. They also contain apparent errors, although it is not clear whether this is due to errors in the original records or in the transcription process.

7.1.2 Ground Truth Data Set

The ground truth data set used in this case study was created by historians [148]. They describe the process as ongoing, and detail many of the challenges that we discuss in the next section including the small domain sizes of attributes such as *First Name* and *Surname*. The approach used by the historians to create the ground truth data relies on similar ideas to those of Fu et al. [53] including age and generational differences between individuals being fixed over time. The approach also makes use of an idea similar to the concept of ambiguity that we presented mathematically in Chapter 5. They discuss how finding a single unambiguous match (for example due to a rare or unique name) not only assists in matching the individual, but also any related entities, an idea we make use of when generating training data in this chapter. The authors describe the process as being ongoing, and while successful in terms of the quality of the results they achieve, their approach requires a very high degree of manual intervention and evaluation.

The same authors extended the initial data set creation in a further study that attempted to investigate population movement off the island [60]. By searching for individuals born on the Isle of Skye in the censuses and death records of Scotland, England and Wales, they were able to identify a number of people who had left the island. By doing this, they were able to reduce the number of individuals who were unmatched in the Isle of Skye data sets and increase the confidence in other results.

Finally, it is important to note that the construction of the ground truth data has favoured precision over recall, so it is likely that the total number of individuals on the Isle of Skye was lower than 54,545 for the period 1861 to 1901. As shown in Table 7.1, many of the records in each data set have not been linked to any other record. While it is possible for example that a child born on the Isle of Skye, moved away before they could be recorded in a census, or an individual moved to the isle just before dying, the high proportions of such records, suggests that the links in the ground truth data set are only a portion of the true links, something that the data set creators note in the data definitions document.

7.1.3 Challenges

There are several challenges that make obtaining training data for these data sets extremely difficult. The general challenges of population reconstruction described by Christen et al. [31] apply here, but of particular note is the poor data quality, and the nature of the problem domain.

A major challenge is simply poor data quality. There are many missing attribute values in

all the data sets, although the census records are generally of a higher quality than the birth, death and marriage certificates. For example, 419 birth records are missing the *First Name* of the baby. While this may be correct, for example if the baby died during or immediately after birth, or a name had not been chosen at the time of registration, missing information in key attributes causes problems with blocking and similarity calculations. In addition, there are attributes which would be valuable for both generating training data and the subsequent entity resolution process, but which have such a high proportion of missing values that they are effectively rendered useless. An example is the attribute *Mother's Occupation* in the births data set, which has less than 10% completeness.

Another challenge is that the domains of many attributes are very small, with a very skewed distribution. In the birth records, the ten most common values of *First Name* account for approximately half the records, and the situation is the same for the *Surname* attribute. In addition, the most common value of *First Name*, 'John' and the most common value of *Surname*, 'Macdonald', each make up nearly 10% of their respective attribute values. The situation is similar for occupations. For the attribute *Father's Occupation* over 30% have a variation of 'farmer' ('fmr', 'crofter', 'cr', etc.) as the attribute value and another 20% have a variation of 'fisherman' ('fish', 'seaman', 'sea', etc.) as the attribute value. In addition, due to the different ways of recording what is essentially the same occupation, traditional string comparison methods are unlikely to work well on these attributes. The domain size of the *Address* attribute is small since there are only a small number of villages on the Isle of Skye, so most records have similar address values. To further complicate matters, even within a single household, living children were sometimes given the same first name [56], meaning that even if a birth certificate could be linked to a household in a census, it may still be impossible to link the baby to the correct household member with certainty.

The presence of attributes such as *First Name*, *Surname*, *Occupation* and *Age*, on each record type (birth records, census records, etc.) initially suggests that traditional approaches to generating training data should be viable for this problem. However, the limited size and the skewness of the domain in many attributes, combined with a high proportion of missing values, mean that none of the approaches presented in Chapter 5 are effective. Even active learning (described in Chapter 3) may not be viable given that the ground truth data set has taken multiple domain experts more than a decade to create and still likely contains missing and incorrect matches. However, by extending our training data generation approach from Chapter 5 to include relationship information, we can produce a set of high quality training data for this problem.

7.2 Relational Ambiguity Approach

The approach we use to generate training data for this case study is similar to the one we presented in Chapter 5, and it exploits one-to-one and one-to-many matching constraints and the concept of ambiguity in order to generate training data for this problem. However, rather than comparing candidate record pairs and looking for examples where there is only a single record pair with similarity above a certain threshold (i.e. the match is unambiguous), we use relationship information to validate the training examples. We then expand our training data by matching the related records of the validated training examples. As a final step, we compare groups of records, and where a particular group has only a single valid candidate then the records within the groups can be matched. Essentially, even though the individual records may have many candidates for matching, collectively they are sufficiently unique to resolve this ambiguity.

7.2.1 Preliminaries

The first part of our work was to confirm that traditional techniques for generating training data would not be effective on this problem. To this end, we performed blocking and similarity calculations as for traditional entity resolution (and a necessary step towards using the bootstrapping approaches described in Chapter 5). Our first observation with respect to the blocking is that there are not many reliable attributes to use. The primary entity on every record type has *First Name*, *Surname*, *Gender*. The attribute *Year of Birth* can be derived for each record, noting that for census, marriage and death records it may be incorrect by one year. Census, marriage and death records also have *Occupation* and *Address* attributes, however *Occupation* is problematic for many different groups of people. Married women and children are often given their husband's or father's occupation on the census (e.g. 'Farmers Wife', 'Farmers Daughter') and typically have no occupation listed on death certificates. There is also no standard in the way particular occupations are recorded, even after data cleaning. This means that blocking based on *Occupation* is likely to miss a significant number of true matches. The *Address* attribute is typically more complete, however the value of this can change as people move, meaning it is also risky to use for blocking purposes. As a result, we primarily rely on the attributes *First Name*, *Surname* and *Gender* to perform blocking. We test three blocking schemes using sorted neighbourhood based indexing and traditional blocking and the details are described in Table 7.2.

The similarity calculation step suffers from the same problems as blocking since there are

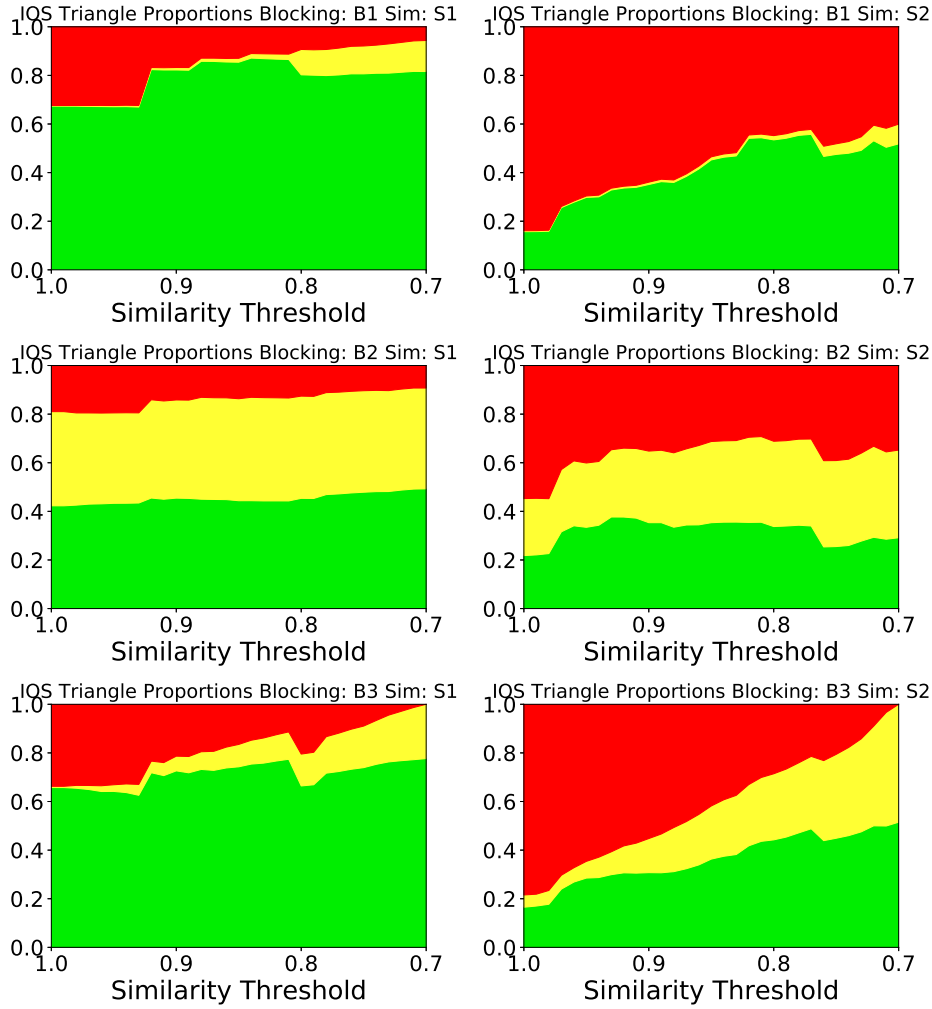

Figure 7.1: Triangle proportions for the IOS data set using different combinations of blocking attributes and similarity functions detailed in Table 7.2. The plots show the normalised proportions of *Consistent Triangles* (green), *Incomplete Triangles* (yellow) and *Inconsistent Triangles* (red), produced using the unsupervised evaluation approach described in Chapter 6.

Blocking Approaches		
B1	SN	Key = *Gender* + *Surname* + *First Name* + *Year of Birth* - Window = 200
B2	SN	Key = *Gender* + *First Name* + *Surname* - Window = 200
B3	TB	*Gender* + *First Name* ∪ *Gender* + *Surname*

Similarity Calculations	
S1	⟨*First Name*, `Jaro`, 0.4⟩, ⟨*Surname*, `Jaro`, 0.4⟩, ⟨*Year of Birth*, `AgeDiff.`, 0.2⟩
S2	⟨*First Name*, `Jaro`, 0.25⟩, ⟨*Surname*, `Jaro`, 0.25⟩, ⟨*Year of Birth*, `AgeDiff.`, 0.2⟩ ⟨*Birth Parish*, `qgram`, 0.15⟩, ⟨*Occupation*, `qgram`, 0.15⟩

Table 7.2: The blocking approaches and similarity calculations tested on the case study data sets. We used a combination of Sorted Neighbourhood indexing (SN) and traditional blocking (TB). For traditional blocking, we blocked separately on each attribute combination and took all resulting candidate pairs. Similarity calculations methods are described by triples of ⟨*Attribute*, similarity-function, weighting⟩ values.

few reliable attributes available. The common attributes also vary depending on the certificate types in question, so the similarity calculation between the baby on a birth certificate and a census record is different to the similarity calculation between the baby on a birth certificate and a death certificate. In order to address this problem, all our similarity calculations use the following two principles:

1. Where an attribute is not present for a particular record type, or one or both of the records does not have a value in the attribute, then remove it from the similarity calculation for that record pair only.

2. Normalise the similarity calculation based on the attributes that were used.

What this means in practice is that where both records in a candidate pair have values for an attribute we use it in the similarity calculation, however the similarity score is not reduced for missing attributes or attribute values.

The attributes, functions and weightings we use in the similarity calculations are described in Table 7.2. The similarity functions are described in Chapter 2, with the exception of `AgeDifference`, which is the function we use to calculate the similarity in the *Age* attribute (or *Year of Birth* attribute), given it may be incorrect by one year. It is calculated as:

$$S_{AgeDiff}(a_1, a_2) = \begin{cases} 1 & |a_1 - a_2| \leq 1 \\ (6 - |a_1 - a_2|) \times 0.2 & 1 < |a_1 - a_2| < 6 \\ 0 & 6 \leq |a_1 - a_2| \end{cases} \tag{7.1}$$

where a_1 and a_2 are the age values being compared. An age difference of 0 or 1 gives similarity of 1.0. As the age difference increases, similarity decreases linearly until it is 0.0 when the age difference is 6 or more years.

We test six different combinations of blocking and similarity functions made up by combining three blocking or indexing approaches and two similarity calculation approaches. We evaluate them using our unsupervised evaluation approach described in Chapter 6 to determine which ones are the most likely to be useful. The results are shown in Figure 7.1.

Our main observation about the results is that none of them are particularly convincing. The best results appear to be the first or third blocking approach combined with the first similarity calculation, where a high proportion of consistent triangles are achieved. However, both these options show a high proportion of inconsistency with similarity close to 1.0, which is driven partly by differences in the *Year of Birth* attribute and partly by missing values (as explained below). The differences in the *Year of Birth* attribute means that long chains of records are being matched together which cannot represent real entities. Within the chains, individual matches only differ by one year in the *Year of Birth* attribute, but across the chain the difference in year of birth is greater than one, which cannot reflect the real-world situation (since year of birth is incorrect by at most one year). It is likely that some of these are caused by data errors, where the *Age* attribute was incorrect and thus the derived *Year of Birth* attribute is incorrect by more than one year. However, the high proportion of these cases, even for a similarity threshold of 1.0, suggests that there are likely to be false matches included, which is a problem when generating training data.

It is also worth discussing the very high proportion of inconsistent triangles in all approaches that use similarity calculation 'S2', since at first these appear quite unusual. It is due to our decision to exclude attributes from the similarity calculation where one of the records has no value in the attribute - something that occurs in the attributes *First Name*, *Surname* and *Year of Birth*, but is much more common in the *Occupation* attribute, hence the results for 'S2' appearing much worse than the results for 'S1'.

7.2.2 Using Relationships to Generate Training Data

The poor results from the unsupervised evaluation of the blocking and indexing step, and the similarity calculation step are not surprising, given the challenges discussed in the previous section. As such, after conducting the initial experiments to confirm that trying to generate training data using the traditional bootstrapping approaches described in Chapter 5 is unlikely

to be effective, we now propose a refinement of the technique we presented in Chapter 5 that can deal with this situation. The refined technique still uses one-to-one matching constraints and ambiguity (defined in Chapter 5) in order generate matches, however it also uses relationship information to both extend and validate the matches that we generate.

Since each record type has different attributes and levels of completeness, we treat each match type separately, census-to-census, birth-to-census, etc. with the aim of creating different sets of matches for each match type. We first compare the census data sets, and then proceed with other match types.

The refinement of the training data generation process presented in Chapter 5 works as follows for the census-to-census matches:

1. The first step was to generate an initial set of unambiguous matches. We used traditional blocking based on the value of *First Name* and separately based on the value of *Address*. We compared each census data set with the subsequent one (i.e. the 1861 census was compared to the 1871 census, 1871 to 1881, and so forth) and where there was only a single candidate for matching detected, we treated the pair of records as a potential match. To compare two records, we used a strict similarity requirement on each of the following attributes: *First Name*, *Surname*, *Year of Birth*, *Gender* and *Birth Parish*.

2. The second step of our approach was to validate the potential matches we identified in the first step. For each potential match, we considered the households of the two records and looked for potential matches between other individuals in the households. If there were at least four potential matches between household members, or more than half the members of each household had a potential match in the other household, we treated the original potential match as validated and added it to our training set. We also added any additional potential matches among other household members to the training set (provided they were also unambiguous). If there were insufficient potential matches between the two households to validate the original potential match, we discarded it.

3. The final step in the training data generation was to extend the concept of ambiguity to groups. The idea was essentially identical to the approach we described in Chapter 5 however instead of just considering individual records, we looked for households that collectively had only a single candidate for matching in the following census. We used the same criteria to select candidates households as we did for the group validation, i.e. at least three records in the household needed to have a potential match with the other

household, or more than 50% of the records in the household needed to have a potential match. Where a household had only a single candidate household in the following census by these criteria, all the potential pairwise matches were added to the training set.

The approach we used to generate training data for the other record types followed the same three steps, i.e. first - identify potential matches, second - validate potential matches using relationship information, third - identify unambiguous group matches. However, because the groups on the birth, death and marriage certificates are typically smaller than the households in the census data sets, we had to tailor our matching requirements and validation requirements to the match type.

Birth-to-census matches: since we do not have a value for *Year of Birth* or *Birth Parish* for parents on a birth certificate, we required both parents to have a potential match in the census household during the validation step and the group matching step. We also added a constraint that each parent must be at least 20 years older than the child. In other words, if a baby was born in 1865, when trying to match his or her parents to records in the census, we only considered people born prior to 1845. While this certainly excluded a small number of true matches, the great majority of parents were over the age of 20 at the time of their child's birth and this constraint eliminated a lot of false positives.

Death-to-census matches: for death certificates, the situation was more difficult than for birth certificates, since there is no guarantee that anyone else on the death certificate (either of the deceased's parents or the deceased's spouse) was alive during the previous census. As such, to perform validation we only required that a single related individual have a potential match. Similarly to the birth certificates, we do not have values for the *Year of Birth* or *Birth Parish* of any related entities. Instead, to improve the validation process, we make use of the *Address* attribute, since it is the only relatively complete and reliable attribute that can be extended to the relatives of the deceased.

Marriage-to-census matches: similar to the situation with death certificates, there is no guarantee that anyone besides the bride and groom was alive to be recorded in the census, either before or after the marriage. It was also common to change household at the time of, or shortly after, a marriage, meaning that using the parents' details for validation of matches with the census data is more difficult. As a result, to validate potential matches in step two, we only considered the bride and groom records, and selected marriage records as training data where there was a match for both the bride and groom to census records in the same household.

Finally, we note that for all match types, we limited matches to the either the following census or the previous census as appropriate. Birth-census matches and marriage-census matches used the following census, and death-census matches used the previous census.

7.3 Evaluation

We evaluated our approach by calculating the precision, recall and the f-measure (each described in Chapter 2) for each of the different match types. As with our experimental evaluation in Chapter 5, the primary considerations when producing training and bootstrapping data was to generate enough high quality matches that the resulting entity resolution process can learn a classification model. To this end, we prioritise precision over recall in the evaluation. Where recall is potentially too low to be useful, we discuss the reasons and the potential consequences in the next section.

The census-to-census results are shown in Table 7.3 and the births-to-census, deaths-to-census and marriages-to-census results are shown in Table 7.4. The *Potential Candidates* row corresponds to our approach from Chapter 5. The *Validated Matches* row is the results after some of the potential matches have been removed because the related records of the potential match are not likely to themselves be matches. For the census data sets we also tried to match the related records, and the additional matches this generated are shown in the *Related Matches* row. Because birth records and death records only have one entity that appears in the ground truth data set (the new-born or the deceased), there cannot be additional related matches generated for these data sets. For the marriages data set, the validation process that we applied essentially means that either both the bride and groom are matched, or neither is matched, so related matches are also impossible in this case. The *Group Matches* row indicates additional matches that were found based on groups of records that collectively had low ambiguity. Finally, the *Total Matches* row records the overall results of our relational training data generation process.

In addition to our approach from Chapter 5, we also tested the baseline approaches from that chapter, including the simple threshold classifier (STC), the greedy threshold classifier (GTC) and the single option set approach described by Kalashnikov and Mehrotra [86] (SOS). We tested these for different values of ψ_{min} on each of the two best performing blocking and similarity calculation techniques shown in Figure 7.1. The results for blocking technique *B1* and similarity calculation technique *S1* are shown in Figure 7.2, and the results for blocking technique *B3* and similarity calculation technique *S1* are shown in Figure 7.3.

		1861-1871			1871-1881			1881-1891			1891-1901		
Total True Matches		10,764			9,647			10,249			8,484		
Potential Candidates	Matches	2,056			2,232			2,276			1,932		
	TM | FM	1,748	308		1,859	373		2,080	196		1,688	244	
	Pr | Re | F-m	0.8500	0.162	0.273	0.833	0.193	0.313	0.914	0.203	0.332	0.874	0.199	0.324
Validated Matches	Matches	1,065			1,225			1,290			998		
	TM | FM	1,057	8		1,212	13		1,289	1		994	4	
	Pr | Re | F-m	0.992	0.098	0.179	0.989	0.126	0.223	0.999	0.126	0.223	0.996	0.117	0.210
Related Matches	Matches	674			634			622			637		
	TM | FM	657	17		606	28		611	11		628	9	
	Pr | Re | F-m	0.975	0.061	0.115	0.956	0.063	0.118	0.982	0.060	0.112	0.986	0.074	0.138
Group Matches	Matches	1,743			1,825			1,576			1,274		
	TM | FM	1,683	60		1,735	90		1,526	50		1,220	54	
	Pr | Re | F-m	0.966	0.156	0.269	0.951	0.180	0.302	0.968	0.149	0.258	0.958	0.144	0.250
Total Matches	Matches	3,482			3,684			3,488			2,909		
	TM | FM	3,397	85		3,553	131		3,426	62		2,842	67	
	Pr | Re | F-m	0.976	0.316	0.477	0.964	0.368	0.533	0.982	0.334	0.499	0.977	0.335	0.499

Table 7.3: The number of matches produced at each stage of the training data generation for the census data sets. At each stage we show the number of identified matches, the split between True Matches and False matches and the precision, recall and the f-measure values.

		Births - Census			Deaths - Census			Marriages - Census		
Total True Matches		13,559			7,546			4,435		
Potential Candidates	Matches	1,831			1,393			818		
	TM | FM	1,773	58		1,119	274		664	154	
	Pr | Re | F-m	0.968	0.131	0.230	0.803	0.148	0.250	0.812	0.150	0.253
Validated Matches	Matches	1,285			574			265		
	TM | FM	1,282	3		566	8		259	6	
	Pr | Re | F-m	0.998	0.095	0.173	0.986	0.075	0.139	0.977	0.058	0.110
Group Matches	Matches	2,116			246			178		
	TM | FM	2,038	78		232	14		168	10	
	Pr | Re | F-m	0.963	0.150	0.260	0.943	0.031	0.060	0.944	0.038	0.073
Total Matches	Matches	3,401			820			443		
	TM | FM	3,320	81		798	22		427	16	
	Pr | Re | F-m	0.976	0.245	0.392	0.973	0.106	0.191	0.964	0.096	0.175

Table 7.4: The number of matches produced at each stage of the training data generation for the births, deaths and marriages data sets. At each stage we show the number of identified matches, the split between True Matches and False matches and the precision, recall and the f-measure values.

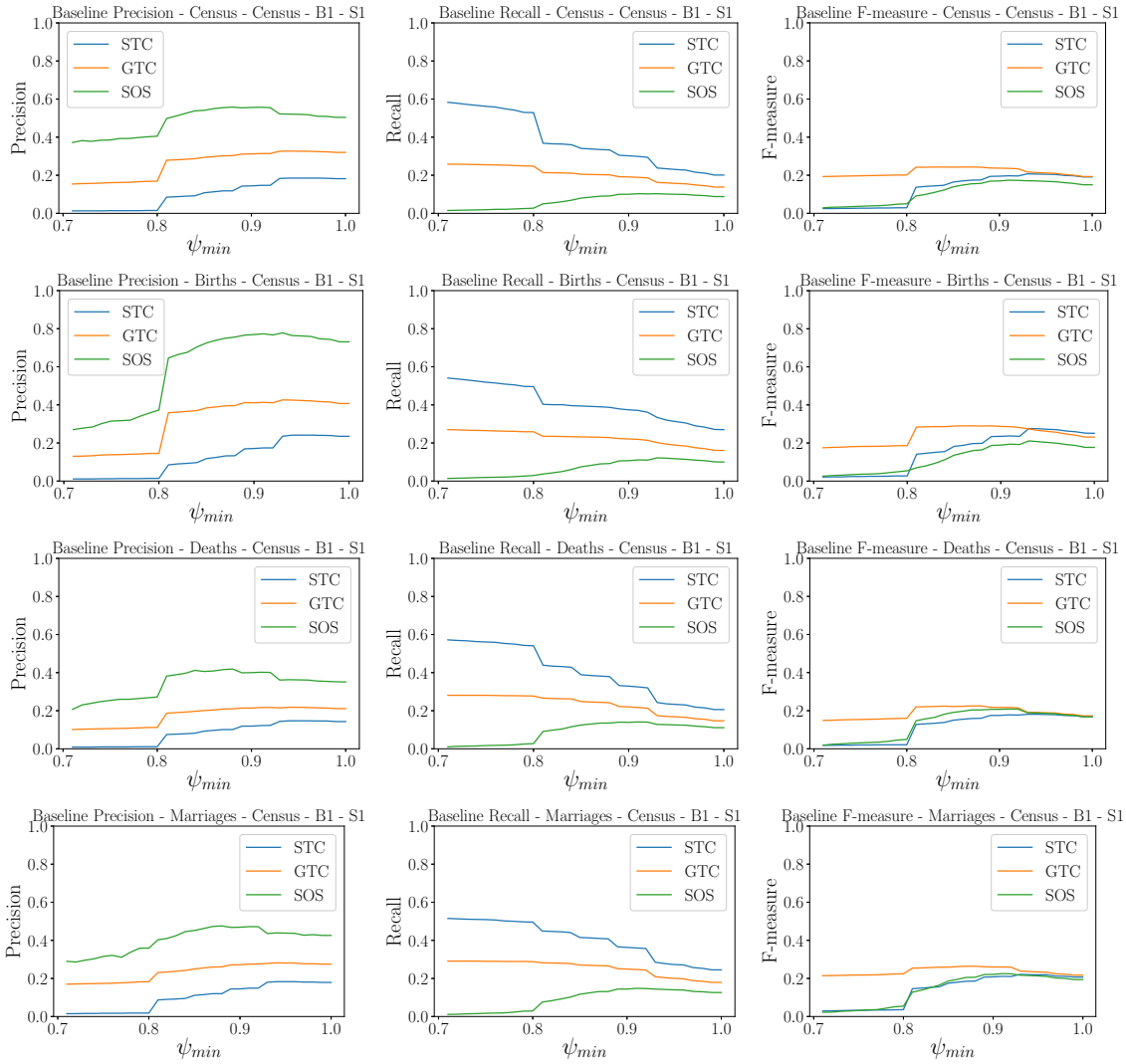

Figure 7.2: Baseline precision, recall and the f-measure results using blocking method B1 and similarity calculation method S1.

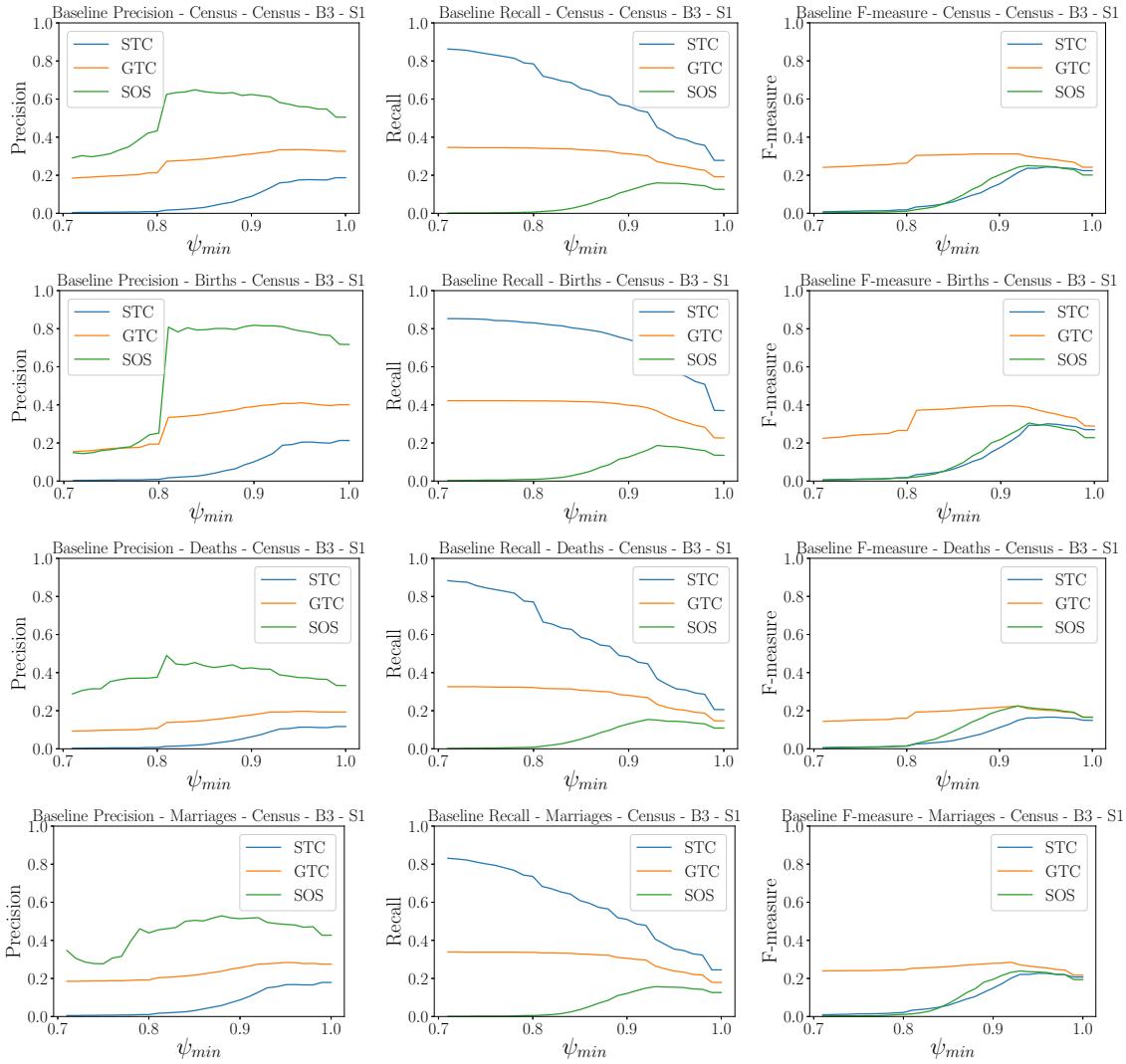

Figure 7.3: Baseline precision, recall and the f-measure results using blocking method B3 and similarity calculation method S1.

7.4 Discussion

Given that for training data generation, a very high precision is more important than a high recall, the overall results are promising, suggesting that for this type of problem, expanding the concept of ambiguity to include related records may be very effective. However, the results do vary significantly across each of the different match types, and we analyse this in more detail now.

The census data sets were the easiest to generate training data for, and there are a number of reasons why this was the case. The groups in the census data sets are often much larger than those on the other record types, which allows many more related entities to be assessed as part of both the validation process and the group ambiguity linkage process. The fact that the related entities also contributed to the match data (since all the census records are in the ground truth) meant that a single individual match could result in matching the rest of the household, thus improving the recall. Finally, because the census records have a consistent set of attributes, which are more likely to be complete than for some of the other record types, both the identification of unambiguous records and the validation process was easier.

The births data set was easier to generate training data for than we initially expected. A significant factor was the *year of birth* attribute which is correct for the birth record itself, and appears to be more accurate for young people in the census data (we discuss this in more detail below). In addition, although there are generally only two related entities for a birth record (the mother and the father), they are often both alive and part of the same household in the following census. These factors meant that a lot of children could be identified unambiguously in the next census period. Although, there is no age data for the parents, requiring a 20 year age gap between the child and parents in the validation process, greatly reduced the number of false validations that occurred.

The deaths data set was much harder to generate training data for, although the matches we created are still of high quality (the overall precision is still above 97%). Even though a death certificate contains the details of up to four related individuals (the mother, father, spouse and informant), in practice this turned out to be significantly less useful than the parent details on a birth certificate. One reason is that for older people (and the majority of the death records are either old people or very young people), the parents are generally not living themselves, and the spouse is often deceased as well, meaning there is no related entity to use for validation or group linkage (since they do not appear in the census records if they are deceased). In addition,

even where the parents were alive during the previous census, unless they were living in the same household as the deceased, they are not part of the same census household and so cannot be used for validation purposes. This means that the majority of the death-to-census matches in the training data we generated are for young people (where at least the parents and child formed part of the same household in the previous census period). As such, the training data is not likely to be representative of the overall deaths-to-census matches and additional work is required for this case.

Similarly to the death data set, the marriage data set was also difficult to generate large quantities of training data for. While each marriage record contains the details of six individuals (the bride, groom, and both sets of parents), because only the bride and groom are guaranteed to be alive at the time of marriage, the other individuals are less useful for validation and group matching purposes. Similarly, because it was common for one or both of the bride and groom to change household at the time of or shortly after the marriage, we could either look for the bride and groom together in subsequent census periods, or look for each of them with their parents in previous census periods. Because the bride and groom are both guaranteed to be alive at the time of the marriage, we opted to look for the couple together in subsequent census periods. However, the recall of the approach is not particularly high, and it might be necessary to try alternative validation and matching scenarios in order to increase the number of matches in the training set.

For the results of our non-relational training data generation technique described in Chapter 5, only the births-to-census matches achieve a precision greater than 95%, meaning that it would be risky to use them as a training set without the additional validation step we performed. The results of the baseline approaches from Chapter 5 are all very poor, with precision normally below 60% and only the single option set approach of Kalashnikov and Mehrotra [86] achieving a precision greater than 80%, and that only for the births-to-census matches with blocking technique *B3*. Based on the results from Figure 7.1 as well as our own impression of the difficulty of the problem, this is not surprising, but it does highlight how traditional approaches may be inadequate for certain data sets or problem domains.

7.4.1 Observations

Overall, the improved version of the training data generation approach described in Chatper 5 was effective at dealing with the difficulties of the problem. Using relationship information to validate potential matches identified by the similarity calculation step proved effective at

removing the majority of the false matches. In addition, extending the concept of ambiguity to groups allowed for more matches to be added to the training data. While the three stage approach - select potential matches, validate potential matches, select group matches - was most effective when used to identify census-to-census matches, it was still much more effective than any of the baseline techniques when applied on other match types (birth-to-census, etc.). However, this did require customisation of the matching and validation conditions for each match type. While it may be impossible to have a completely generalised technique for problems of this difficulty and complexity, it does suggest that it's application may only be justified in situations where other approaches are inadequate.

The *Year of Birth* attribute is completely wrong for many records, particularly for older people. While we expected this to be the case to some degree, after analysing some of the matches we missed in the ground truth, it differs by over ten years in many cases, suggesting that an individual might know they were 'old' but not really more precisely than that. For individuals under the age of 25, this appears to be less of a problem, but even then, the value may not be completely accurate. Adapting our approach so that the implications of a difference in age were treated differently depending on how old a person was may have helped link more of the census groups. It may also have assisted in linking the deaths records, although given the scarcity of information on many of the death records, there is a risk this would have resulted in too many incorrect matches in the training set. Another option for the census records would be to use the ordering of ages in a household rather absolute ages, although this causes problems where individuals leave the group (and in some cases join the group, such as lodgers or servants).

Some attributes such as *Address* and *Birth Parish* were more useful than we expected, however only if they could be validated with additional information (such as by looking at related individuals). Other attributes such as *Occupation* and *Relationship to Head of Household* could have been incorporated to a greater extent, if more processing was done to standardise the values. However doing this with any degree of accuracy is a challenging process [91] or would require a large amount of manual effort.

Finally, we note that although incorporating relationships into the training data generation process improved both the quality (precision) and quantity (recall) there are still some limitations of our technique. Aside from the problems with the death records discussed above, we were not able to match a large proportion of the groups, either from census-to-census, or from any of the births, deaths or marriages data sets to the census. This means that techniques such as the relational clustering approach of Bhattacharya and Getoor [12], which need the bootstrapping

process to create a path between each of the two records in a true match, will not perform well with this training data set. How to overcome this limitation is something we intend to investigate in the future.

7.5 Conclusion

Many advanced entity resolution techniques require training data or a bootstrapping process in order to work. In practice, however, this can be a significant limitation, because for many real-world entity resolution problems, training data does not exist and it can be very expensive to create. In Chapter 5, we presented a technique that uses ambiguity to generate training and bootstrapping data for entity resolution problems with one-to-one and one-to-many matching constraints. We evaluated the technique on the data sets described in Section 2.5, and showed that it was more successful than the baseline techniques at generating training data when such matching constraints exist. However, many advanced entity resolution techniques incorporate relationships between records in the entity resolution process. When generating training data for these techniques, our ambiguity based approach can be extended to also include relationships in the training data generation process.

In this chapter, we have undertaken a case study on population registry data from the Isle of Skye in Scotland, United Kingdom. We showed how we can expand our ambiguity based approach from Chapter 5 to include relationships, and that these can be used to both validate the potential matches in the training data, and also expand the size of the training data set. Related records can be themselves be matched, and in addition, groups of related records can be unambiguous, even where the individual records have multiple candidates for matching. We evaluated our extended approach on a data set and problem domain which are extremely challenging. We showed that even though all the baseline approaches are inadequate, our approach can produce a high quality set of training data for each of the different match types.

Chapter 8

Conclusion

Entity resolution, also called record linkage, deduplication and many other names, is the common data pre-processing problem of determining which records in one or more data sets correspond to the same real-world entities. Entity resolution has a variety of applications in domains such as government agencies, research institutions and commercial organisations.

Traditional entity resolution techniques were formulated in the late 1960s [47], and since that time, each aspect of the entity resolution process has been an active area of academic research. This has resulted in many entity resolution techniques becoming very sophisticated in order to solve specific types of problems, and to overcome limitations of traditional entity resolution approaches. Collective entity resolution techniques [12, 39, 86, 165] were developed to incorporate relationships into the entity resolution process as well as capture dependencies in matching decisions. Group linkage techniques [34, 53, 54, 126] deal with entity resolution problems where records are grouped together in some fashion and considering the groups as well as the individual records is much more effective than traditional entity resolution approaches. Population reconstruction techniques [29, 42, 157] deal with complex roles and relationships between records and often have to operate with very limited information. Temporal entity resolution techniques [21, 28, 106] incorporate time stamp data into the entity resolution process and capture the likelihood of change within different attribute values.

While techniques such as these have been shown to be very effective in academic experiments, there are many difficulties that need to be overcome in order to use them on real-world entity resolution problems. In this thesis we have aimed to address three particular limitations of advanced entity resolution techniques. While these difficulties also apply to traditional entity resolution, they are much more problematic for advanced entity resolution techniques.

- **Scalability**: The scalability for even traditional entity resolution techniques is poor, and advanced entity resolution techniques are typically even worse. The scalability of collective

entity resolution techniques range from O($nlog_2n$) for the relational clustering technique of Bhattacharya and Getoor [12] when used with an ideal blocking algorithm, through to worse than NP-Hard for Markov logic networks [150]. Some temporal entity resolution and group linkage techniques are only marginally slower than traditional entity resolution [28, 54], however techniques that enforce constraints or consider entity evolution [21, 53, 126] typically have very poor scalability. In such cases, traditional techniques for improving the scalability of entity resolution may be insufficient to make the problem tractible when applied to very large data sets.

- **Lack of Training Data**: Many real-world entity resolution problems do not have training data. In the absence of such training data, collective entity resolution techniques [12, 86] need some kind of bootstrapping phase in order to generate the starting relationships. Many group linkage techniques are supervised [52, 53], meaning that they require an initial set of training data in order to work correctly. Similarly for temporal entity resolution techniques, training data is also required in order to calculate parameters such as agreement decay and disagreement decay [28, 106] so a lack of training data in these instances is a significant problem.

- **Parameter Tuning and Evaluation**: Traditional entity resolution has a number of parameters to tune, including choice of attributes, similarity functions, weightings, similarity thresholds, blocking or indexing approach, blocking or indexing attributes, etc. Advanced entity resolution techniques typically have the same parameters and often many additional ones. While experience and domain expertise can assist in choosing appropriate techniques and setting parameters, it is often still necessary to run the entity resolution process multiple times in order to fine-tune the approach for good performance. Given the poor scalability of most advanced entity resolution techniques, and the normal lack of training or evaluation data (see above), this can be an extremely time-consuming process, particularly if a manual evaluation is required for each iteration.

Each of these limitations is much more likely to be present for real-world entity resolution problems than in an academic environment. In an academic environment, problems with evaluation, model tuning, obtaining training data and scalability can potentially all be overcome (or at least minimised) by choosing an appropriate data set or data sets on which to evaluate the techniques, something that is unlikely to be possible in real-world applications.

8.1 Recap of Contributions

In this thesis, we make three main contributions, each aimed at partially addressing one of the limitations of advanced entity resolution techniques that we described in Chapter 1, and summarised in the previous section.

- Our first contribution is described in Chapter 4 and aims to create an improved blocking process for advanced entity resolution techniques. In order to achieve this, we proposed a blocking technique that produces blocks within a size range. This technique will limit the effects of the poor scalibility of many advanced entity resolution techniques, since the processing time for blocks of a fixed size is constant. Our proposed approach also produces disjoint blocks which allows graph based advanced entity resolution techniques to operate on individual blocks rather than having to work with the entire data set(s) at once.

- Our second contribution was an approach to generate training and bootstrapping data which is detailed in Chapter 5. Since many advanced entity resolution techniques either require training or boostrapping data, or work better when it is available, our proposed approach is suitable for such techniques. We showed how we can exploit matching constraints that are present in many entity resolution problems in order to improve the size of the training data set without sacrificing precision. We also showed how our concept of ambiguity can be applied to active learning in order generate additional training data when a limited number of manual classifications are available.

- Our third contribution was an unsupervised approach to evaluating entity resolution which is described in Chapter 6. Our technique makes use of transitive closure to detect inconsistencies in entity resolution results. This approach can be applied between the similarity calculation and classification steps in the entity resolution process in order to detect parameter settings that will result in a high degree of inconsistency, allowing them to be discarded prior to the classification step and without the need for a manual evaluation. The approach can also be applied as part of the evaluation step in order to detect groups of records that require further investigation or assessment, and which may be indicative of wider problems in the entity resolution process.

In addition to our three main contributions, we extended the training data generation approach presented in Chapter 5 to incorporate relationship information. We demonstrated that

even on a very difficult problem, where other approaches to generating training data were not effective, by exploiting one-to-one matching constraints, and validiting potential matches by considering related records, we were able to generate a high quality set of training data for each matching scenario. We evaluated this approach as a case study, using historical registry data from the Isle of Skye in Scotland, United Kingdom. This case study is detailed in Chapter 7.

8.2 Future Work

While our work has made significant progress in overcoming some of the practical difficulties that limit the widespread use of advanced entity resolution techniques in real-world situations, there is still room for further improvement. In addition, many of our techniques can be modified for application in other domains or on related problems. We briefly summarise some directions we intend to extend our work in the future.

8.2.1 Blocking

Our current approach to blocking relies on domain expertise to select the blocking functions and blocking attributes. While this is common for most blocking techniques, as we discussed in Chapter 6, it can lead to poor entity resolution results. In addition, choosing blocking functions and blocking attributes requires knowledge of both the problem domain and the entity resolution techniques being applied and there may be no individual with expertise in both areas. Kejriwal and Miranker [87] and Ramadan and Christen [144] presented approaches for automatic selection of blocking keys for entity resolution and a similar approach would have value for our work as well.

We also intend to test our work on other problem areas. In particular, real-time entity resolution scenarios [12, 27, 145, 173] only allow a fixed number of comparisons in a certain time-frame. Since our approach controls the maximum size of each block, we can guarantee this requirement is met. However, real-time entity resolution problems also typically feature dynamic data sets, which presents two additional problems that need to be overcome before our blocking approach can be applied. Firstly, what to do with blocking key values that were not present in the original data set, and secondly how to update blocks while still ensuring that size constraints are satisfied. We intend to investigate whether a balanced tree data structure would allow us to maintain both the size and disjoint properties while being fast enough to update and query in real-time situations.

8.2.2 Training Data Generation

There are a number of directions that we hope to extend our work on training data generation, and the use of ambiguity in the future. We intend to investigate whether ambiguity can be incorporated into the entity resolution process in addition to attribute and relational similarity measures. This could be particularly effective for classifiers that take an iterative approach to entity resolution, where they resolve easier cases first based on some ordering of increasing difficulty. Ambiguous record pairs could be deferred until later in the process in the hope that resolving other records pairs might reduce the ambiguity and allow the correct match to be more easily determined.

In addition, as was demonstrated in the Isle of Skye case study (see Chapter 7), ambiguity is useful at the group level as well as the individual record level. In the future we hope to formalise the notion of group-based ambiguity and incorporate it into a group based entity resolution approach for situations where one-to-one and one-to-many matching constraints apply.

Finally, as noted in Chapter 3, there are many similarities between one-to-one matching restrictions in entity resolution and the so called *stable marriage problem* [55]. We hope to investigate whether any of the algorithms used to solve the stable marriage problem and its variants, can be adapted to work in the entity resolution domain.

8.2.3 Unsupervised Evaluation

There are also directions we intend to extend the unsupervised evaluation approach from Chapter 6. In our current approach, the similarity threshold ψ_{ij} is treated as a single value. However, for *inconsistent* triangles, a breakdown of the similarity values by attribute could potentially help determine why the similarity calculation method is leading to inconsistencies. For example this might make it easier to detect the problem that occured with the UKCD experiments where the *Occupation* attribute was resulting in a high proportion of inconsistent triangles for similarity values greater than 0.75. Similarly for the blocking approach, knowing the block combinations that were leading to *incomplete* triangles would make it easier to adjust the blocking techniques accordingly.

Just as it would be possible to separate ψ_{ij} into different components based on the attributes used, so it would also be possible to have different values of ψ_{min} in different parts of the graph. Along similar lines, the entity resolution problem can be reframed as a community detection problem in graphs [51], and while many of the algorithms for this are computationally

expensive, there are potentially approaches from this area that could be adapted to our problem.

While our approach was effective at detecting a high degree of inconsistency, as we discussed in Section 6.4, consistent results are not guaranteed to be useful. In extreme cases this is relatively easy to detect, however, we found that for less obvious cases two additional pieces of information were very effective in spotting problems, namely the total number of matches and the size of the largest cluster. While they can be evaluated separately as we did in our experimental evaluation, we intend to investigate whether they can be incorporated into the measures, potentially through the normalisation process. This would maintain the relatively simple output of our approach while still allowing us to identify additional, less obvious, problems.

8.3 Conclusion

Entity resolution is an important data pre-processing task. In real-world applications it is often necessary when integrating, preparing or maintaining data sets prior to analysis or as part of ongoing business. It has been an active field of academic research since the late 1960s and numerous techniques have been developed for each part of the entity resolution process or to apply to particular types of entity resolution problems.

However, despite these techniques achieving good results in controlled experiments in academic environments, there is a significant gap between the work done in academia and real-world entity resolution applications. In particular, advanced entity resolution techniques do not see as wide-spread practical application as would be suggested by the quality of results they achieve in academic research.

We identified three limitations of advanced entity resolution techniques that can all be controlled or overcome in a research environment but which are much more problematic in many real-world applications, name poor scalability of advanced entity resolution techniques, a lack of training and bootstrapping data required by many advanced entity resolution techniques, and the high degree of parameter tuning and model refinement required by advanced entity resolution techniques.

In this thesis, we have presented three techniques to partially overcome these limitations. We developed a blocking technique to strictly control the size of blocks and thus limit the impact of poor scalability. We developed a technique to generate training and bootstrapping data in situations where one-to-one or one-to-many matching constraints apply. Finally we developed an unsupervised evaluation technique which allows parameter settings and techniques that pro-

duce entity resolution results with a high degree of inconsistency to be quickly identified and discarded. It is our hope that due to these techniques and similar ones developed in the future, real-world application of advanced entity resolution techniques will be more feasible for future practitioners.

Bibliography

1. Adamic, L. A. and Adar, E., 2003. Friends and neighbors on the Web. *Social Networks*, 25, 3 (2003), 211–230. Elsevier. (cited on pages 17 and 27)

2. Aizawa, A. and Oyama, K., 2005. A fast Linkage Detection Scheme for Multi-Source Information Integration. In *Web Information Retrieval and Integration (WIRI)*, 30–39. IEEE. (cited on page 23)

3. Althoff, T.; Dong, X. L.; Murphy, K.; Alai, S.; Dang, V.; and Zhang, W., 2015. TimeMachine: Timeline Generation for Knowledge-Base Entities. In *Proceedings of the 21st International Conference on Knowledge Discovery and Data Mining (KDD)*, 19–28. ACM. (cited on page 46)

4. Antonie, L.; Inwood, K.; and Ross, J. A., 2015. Dancing with Dirty Data: Problems in the Extraction of Life-Course Evidence from Historical Censuses. In *Population Reconstruction*, 217–241. Springer. (cited on page 86)

5. Arasu, A.; Götz, M.; and Kaushik, R., 2010. On Active Learning of Record Matching Packages. In *Proceedings of the 36th International Conference on Management of Data (ICMD)*, 783–794. ACM. (cited on page 53)

6. Arasu, A.; Ré, C.; and Suciu, D., 2009. Large-Scale Deduplication with Constraints Using Dedupalog. In *Proceedings of the 25th International Conference on Data Engineering (ICDE)*, 952–963. IEEE. (cited on page 51)

7. Australian Broadcasting Commission, 2011. Centrelink high-rollers ordered to pay back millions. Https://www.abc.net.au/news/2011-04-17/centrelink-high-rollers-ordered-to-pay-back/2613392 Accessed: 2018-11-22. (cited on pages 1 and 32)

8. Bagon, S. and Galun, M., 2011. Large Scale Correlation Clustering Optimization. *arXiv preprint arXiv:1112.2903*, (2011). (cited on page 55)

9. Bansal, N.; Blum, A.; and Chawla, S., 2004. Correlation Clustering. *Machine Learning*, 56, 1-3 (2004), 89–113. Springer. (cited on page 55)

10. Bass, J.; Silcot, S.; and Smith, L., 2015. Founders and Survivors Linkage Strategy. In *Population Reconstruction*. Springer. (cited on pages 1 and 33)

11. Bellare, K.; Iyengar, S.; Parameswaran, A. G.; and Rastogi, V., 2012. Active Sampling for Entity Matching. In *Proceedings of the 18th International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM. (cited on page 53)

12. Bhattacharya, I. and Getoor, L., 2007. Collective Entity Resolution in Relational Data. *Transactions on Knowledge Discovery from Data (TKDD)*, 1, 1 (2007), 5–es. ACM. (cited on pages 1, 2, 4, 5, 6, 7, 16, 26, 38, 39, 40, 41, 44, 56, 60, 61, 79, 81, 82, 83, 84, 85, 95, 96, 104, 129, 147, 149, 150, and 152)

13. Bothwell, E., 2016. World University Rankings 2016-2017: results announced. Https://www.timeshighereducation.com/news/world-university-rankings-2016-2017-results-announced Accessed: 2018-11-22. (cited on page 33)

14. Bourn, N. A., 1995. *The ecology, conservation and population genetics of three species of Zygaenid moths, Zygaena lonicerae, Zygaena purpuralis and Zygaena filipendulae in North West Scotland*. Ph.D. thesis, University of Aberdeen. (cited on page 130)

15. Brizan, D. G. and Tansel, A. U., 2006. A. Survey of Entity Resolution and Record Linkage Methodologies. *Communications of the International Information Management Association (IIMA)*, 6, 3 (2006), 5. (cited on pages 1 and 38)

16. Brook, E.; Rosman, D.; and Holman, C., 2008. Public good through data linkage: measuring research outputs from the Western Australian Data Linkage System. *Australian and New Zealand Journal of Public Health*, 32, 1 (2008), 19–23. Wiley Online Library. (cited on page 1)

17. Burdick, D.; Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W.-C., 2016. A Declarative Framework for Linking Entities. *Transactions on Database Systems (TODS)*, 41, 3 (2016), 17. ACM. (cited on page 52)

18. Chaudhuri, S.; Ganti, V.; and Motwani, R., 2005. Robust Identification of Fuzzy Duplicates. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, 865–876. IEEE. (cited on page 28)

19. Chen, J.; Niedermeier, R.; and Skowron, P., 2018. Stable Marriage with Multi-Modal Preferences. In *Proceedings of the 19th Conference on Economics and Computation*, 269–286. ACM. (cited on page 52)

20. Chen, Z.; Kalashnikov, D. V.; and Mehrotra, S., 2007. Adaptive Graphical Approach to Entity Resolution. In *Proceedings of the 7th Conference on Digital Libraries*, 204–213. ACM. (cited on page 60)

21. Chiang, Y.-H.; Doan, A.; and Naughton, J. F., 2014. Tracking Entities in the Dynamic World: A Fast Algorithm for Matching Temporal Records. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 7, 6 (2014), 469–480. (cited on pages 1, 2, 4, 5, 6, 46, 149, and 150)

22. Christen, P., 2012. A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *Transactions on Knowledge and Data Engineering (TKDE)*, 24, 9 (2012), 1537–1555. IEEE. (cited on pages 43, 48, and 65)

23. Christen, P., 2012. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer. (cited on pages 3, 4, 15, 20, 22, 23, 26, 28, 29, 30, 38, 43, 49, 55, 65, 66, 71, 102, 104, and 109)

24. Christen, P., 2014. Preparation of a real temporal voter data set for record linkage and duplicate detection research. Technical report, Australian National University. (cited on pages 34 and 85)

25. Christen, P., 2016. Application of Advanced Record Linkage Techniques for Complex Population Reconstruction. *arXiv preprint arXiv:1612.04286*, (2016). (cited on pages 33 and 47)

26. Christen, P.; Churches, T.; and Hegland, M., 2004. Febrl – A Parallel Open Source Data Linkage System. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, vol. 3056, 638–647. Springer. (cited on pages 31 and 108)

27. Christen, P. and Gayler, R., 2008. Towards Scalable Real-Time Entity Resolution using a Similarity-Aware Inverted Index Approach. In *Proceedings of the 7th Australasian Data Mining Conference (AusDM)*, vol. 87, 51–60. Australian Computer Society. (cited on pages 38, 61, 79, and 152)

28. Christen, P. and Gayler, R. W., 2013. Adaptive Temporal Entity Resolution on Dynamic Databases. In *Advances in Knowledge Discovery and Data Mining*, 558–569. Springer. (cited on pages 1, 5, 6, 7, 46, 81, 149, and 150)

29. Christen, P.; Gayler, R. W.; Tran, K.-N.; Fisher, J.; and Vatsalan, D., 2016. Automatic Discovery of Abnormal Values in Large Textual Databases. *Journal of Data and Information Quality (JDIQ)*, 7, 1-2 (2016), 7. ACM. (cited on pages 5, 129, and 149)

30. Christen, P. and Goiser, K., 2007. Quality and Complexity Measures for Data Linkage and Deduplication. In *Quality Measures in Data Mining*, 127–151. Springer. (cited on page 55)

31. Christen, P.; Vatsalan, D.; and Fu, Z., 2015. Advanced Record Linkage Methods and Privacy Aspects for Population Reconstruction – A Survey and Case Studies. In *Population Reconstruction*, 87–110. Springer. (cited on pages 131 and 132)

32. Christen, P.; Vatsalan, D.; and Wang, Q., 2015. Efficient Entity Resolution with Adaptive and Interactive Training Data Selection. In *Proceedings of the 15th International Conference on Data Mining (ICDM)*, 727–732. IEEE. (cited on page 17)

33. Christen, P.; Vidanage, A.; Ranbaduge, T.; and Schnell, R., 2018. Pattern-Mining Based Cryptanalysis of Bloom Filters for Privacy-Preserving Record Linkage. In *Proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 530–542. Springer. (cited on page 76)

34. Christen, V.; Gross, A.; Fisher, J.; Wang, Q.; Christen, P.; and Rahm, E., 2017. Temporal group linkage and evolution analysis for census data. In *Proceedings of the 20th International Conference on Extending Database Technology (EDBT)*, 620–631. (cited on pages 1, 5, 45, 129, and 149)

35. Cochinwala, M.; Kurien, V.; Lalk, G.; and Shasha, D., 2001. Efficient data reconciliation. *Information Sciences*, 137, 1–4 (2001), 1–15. Elsevier. (cited on pages 28 and 104)

36. Dal Bianco, G.; Galante, R.; Gonçalves, M.; Canuto, S.; and Heuser, C., 2015. A Practical and Effective Sampling Selection Strategy for Large Scale Deduplication. *Transactions on Knowledge and Data Engineering (TKDE)*, 27, 9 (2015), 2305–2319. IEEE. (cited on page 53)

37. Das Sarma, A.; Jain, A.; Machanavajjhala, A.; and Bohannon, P., 2012. An Automatic Blocking Mechanism for Large-scale De-duplication Tasks. In *Proceedings of the 21st Conference on Information and Knowledge Management (CIKM)*, 1055–1064. ACM. (cited on page 48)

38. Dinler, D. and Tural, M. K., 2016. A Survey of Constrained Clustering. In *Unsupervised Learning Algorithms*, 207–235. Springer. (cited on page 50)

39. Dong, X.; Halevy, A.; and Madhavan, J., 2005. Reference Reconciliation in Complex Information Spaces. In *Proceedings of the 31st International Conference on Management of Data (ICMD)*, 85–96. ACM. (cited on pages 1, 5, 16, 41, 60, 129, and 149)

40. Du, J. and Ling, C., 2010. Active Learning with Human-Like Noisy Oracle. In *Proceedings of the 10th International Conference on Data Mining (ICDM)*, 797–802. IEEE. (cited on page 54)

41. Efremova, J.; Ranjbar-Sahraei, B.; Oliehoek, F. A.; Calders, T.; and Tuyls, K., 2014. A Baseline Method for Genealogical Entity Resolution. In *Proceedings Workshop on Population Reconstruction*. (cited on pages 7 and 47)

42. Efremova, J.; Ranjbar-Sahraei, B.; Rahmani, H.; Oliehoek, F. A.; Calders, T.; Tuyls, K.; and Weiss, G., 2015. Multi-Source Entity Resolution for Genealogical Data. In *Population Reconstruction*, 129–154. Springer. (cited on pages 5, 47, 129, 130, and 149)

43. Elmagarmid, A. K.; Ipeirotis, P. G.; and Verykios, V. S., 2007. Duplicate Record Detection: A Survey. *Transactions on Knowledge and Data Engineering (TKDE)*, 19, 1 (2007), 1–16. IEEE. (cited on pages 48, 64, 71, and 78)

44. Elmasri, R. and Navathe, S. B. N., 2011. *Database Systems: Models, Languages, Design, and Application Programming*. Pearson. (cited on page 18)

45. Fan, X.; Wang, J.; Pu, X.; Zhou, L.; and Lv, B., 2011. On Graph-Based Name Disambiguation. *Journal of Data and Information Quality (JDIQ)*, 2, 2 (2011), 10. ACM. (cited on page 60)

46. Felche, K., 2013. Harnessing the Power of Data in Government through Analytics. Keynote presentation at the 11th Australasian Data Mining Conference (AusDM). (cited on page 31)

47. Fellegi, I. P. and Sunter, A. B., 1969. A Theory for Record Linkage. *Journal of the American Statistical Association (JASA)*, 64, 328 (1969), 1183–1210. Taylor & Francis. (cited on pages 1, 3, 6, 15, 16, 21, 22, 24, 28, 37, 39, 59, 60, 71, 114, and 149)

48. Fisher, J.; Christen, P.; and Wang, Q., 2016. Active Learning Based Entity Resolution Using Markov Logic. In *Proceedings of the 20th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 338–349. Springer. (cited on page 11)

49. Fisher, J.; Christen, P.; Wang, Q.; and Rahm, E., 2015. A Clustering-Based Framework to Control Block Sizes for Entity Resolution. In *Proceedings of the 21st International Conference on Knowledge Discovery and Data Mining (KDD)*, 279–288. ACM. (cited on page 11)

50. Fisher, J. and Wang, Q., 2015. Unsupervised Measuring of Entity Resolution Consistency. In *Proceedings of the 15th International Conference on Data Mining Workshop (ICDMW)*, 218–221. IEEE. (cited on page 11)

51. Fortunato, S., 2010. Community detection in graphs. *Physics Reports*, 486, 3-5 (2010), 75–174. Elsevier. (cited on pages 55, 127, and 153)

52. Fu, Z.; Christen, P.; and Boot, M., 2011. A Supervised Learning and Group Linking Method for Historical Census Household Linkage. In *Proceedings of the 9th Australasian Data Mining Conference (AusDM)*, vol. 125. Australian Computer Society. (cited on pages 7, 81, 82, 104, and 150)

53. Fu, Z.; Christen, P.; and Zhou, J., 2014. A Graph Matching Method for Historical Census Household Linkage. In *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Springer. (cited on pages 1, 2, 5, 7, 28, 34, 45, 129, 132, 149, and 150)

54. Fu, Z.; Zhou, J.; Christen, P.; and Boot, M., 2012. Multiple Instance Learning for Group Record Linkage. In *Proceedings of the 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. Springer. (cited on pages 1, 5, 28, 44, 81, 82, 104, 125, 129, 149, and 150)

55. Gale, D. and Shapley, L. S., 1962. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69, 1 (1962), 9–15. Taylor & Francis. (cited on pages 52, 106, and 153)

56. Galley, C.; Garrett, E.; Davies, R.; and Reid, A., 2011. Living same-name siblings and British historical demography. *Local Population Studies*, 86, 1 (2011), 15–36. Local Population Studies Society. (cited on page 133)

57. Ganganath, N.; Cheng, C.-T.; and Tse, C. K., 2014. Data Clustering with Cluster Size Constraints Using a Modified *k*-means Algorithm. In *Proceedings of the 6th International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE. (cited on page 50)

58. Garcia-Molina, H., 2004. Entity Resolution: Overview and Challenges. In *Proceedings of the 23rd International Conference on Conceptual Modeling*, 1–2. Springer. (cited on page 38)

59. Garrett, E. and Davies, R., 2003. Birth spacing and infant mortality on the Isle of Skye, Scotland, in the 1880s; a comparison with the town of Ipswich, England. *Local Population Studies*, 71 (2003), 53–74. Local Population Studies Society. (cited on page 130)

60. Garrett, E. and Reid, A., 2015. Introducing 'Movers' into Community Reconstructions: Linking Civil Registers of Vital Events to Local and National Census Data: A Scottish Experiment. In *Population Reconstruction*, 263–283. Springer. (cited on pages 34 and 132)

61. Georgala, K.; van der Burgh, B.; Meeng, M.; and Knobbe, A., 2015. Record Linkage in Medieval and Early Modern Text. In *Population Reconstruction*, 173–195. Springer. (cited on page 130)

62. Getoor, L. and Machanavajjhala, A., 2012. Entity resolution: Tutorial. In *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 1527–1527. (cited on pages 1 and 38)

63. Glasson, E.; De Klerk, N.; Bass, A.; Rosman, D.; Palmer, L.; and Holman, C., 2008. Cohort Profile: The Western Australian Family Connections Genealogical Project. *International Journal of Epidemiology*, 37, 1 (2008), 30–35. Oxford University Press. (cited on page 130)

64. Gottschalk, S. and Demidova, E., 2018. EventKG: A Multilingual Event-Centric Temporal Knowledge . In *Proceedings of the 15th International European Semantic Web Conference*, 272–287. Springer. (cited on page 46)

65. Gruenheid, A.; Nushi, B.; Kraska, T.; Gatterbauer, W.; and Kossmann, D., 2015. Fault-Tolerant Entity Resolution with the Crowd. *arXiv preprint arXiv:1512.00537*, (2015). (cited on page 17)

66. Gu, L. and Baxter, R., 2006. Decision Models for Record Linkage. In *Data Mining*, 146–160. Springer. (cited on page 21)

67. Gupta, S., 2017. A Survey on Balanced Data Clustering Algorithms. *International Journal for Women Researchers in Engineering, Science & Management*, (2017), 2611–2614. (cited on page 50)

68. Han, J.; Kamber, M.; and Pei, J., 2012. *Data Mining: Concepts and Techniques*. Waltham, MA: Morgan Kaufmann, 3 edn. (cited on page 28)

69. Han, Q., 2015. Visualising Complex Linked Data. Australian National University https://cs.anu.edu.au/courses/csprojects/15S2/Reports/Quanwei_Han_ Report.pdf. (cited on page 131)

70. Hand, D. and Christen, P., 2018. A note on using the F-measure for evaluating record linkage algorithms. *Statistics and Computing*, 28, 3 (2018), 539–547. (cited on pages 31 and 55)

71. Hassanzadeh, O. and Miller, R., 2009. Creating probabilistic databases from duplicated data. *The International Journal on Very Large Data Bases*, 18, 5 (2009), 1141–1166. Springer. (cited on page 5)

72. Hernandez, M. A. and Stolfo, S. J., 1995. The Merge/Purge Problem for Large Databases. In *Proceedings of the 21st International Conference on Management of Data (ICMD)*, 127–138. ACM. (cited on page 15)

73. Hernandez, M. A. and Stolfo, S. J., 1998. Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Mining and Knowledge Discovery*, 2, 1 (1998), 9–37. Springer. (cited on pages 15, 24, 48, 60, 71, 101, and 108)

74. Hirsch, J., 2005. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102, 46 (2005), 16569–16572. National Academy of Sciences. (cited on page 33)

75. Holman, C. D. J.; Bass, J. A.; Rosman, D. L.; Smith, M. B.; Semmens, J. B.; Glasson, E. J.; Brook, E. L.; Trutwein, B.; Rouse, I. L.; Watson, C. R.; et al., 2008. A decade of data linkage in Western Australia: strategic design, applications and benefits of the WA data

linkage system. *Australian Health Review*, 32, 4 (2008), 766–777. Commonwealth Scientific and Industrial Research Organisation. (cited on page 130)

76. Hu, Y.; Wang, Q.; Vatsalan, D.; and Christen, P., 2016. Regression classifier for Improved Temporal Record Linkage. In *Proceedings of the 14th Australasian Data Mining Conference (AusDM)*. Australian Computing Society. (cited on page 46)

77. Hu, Y.; Wang, Q.; Vatsalan, D.; and Christen, P., 2017. Improving Temporal Record Linkage Using Regression Classification. In *Proceedings of the 21st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 561–573. Springer. (cited on page 46)

78. Hussain, B.; Hassanzadeh, O.; Chiang, F.; Lee, H. C.; and Miller, R. J., 2013. An Evaluation of Clustering Algorithms for Duplicate Detection. Technical report, University of Toronto, Department of Computer Science. (cited on page 5)

79. Huynh, T. N. and Mooney, R. J., 2008. Discriminative Structure and Parameter Learning for Markov Logic Networks. In *Proceedings of the 25th International Conference on Machine learning (ICML)*. ACM. (cited on page 43)

80. Irving, R. W.; Kavitha, T.; Mehlhorn, K.; Michail, D.; and Paluch, K., 2004. Rank-Maximal Matchings. In *Proceedings of the 15th Annual Symposium on Discrete Algorithms*, 68–75. ACM SIAM. (cited on page 52)

81. Irving, R. W.; Manlove, D. F.; and Scott, S., 2003. Strong Stability in the Hospitals/Residents Problem. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, vol. 2607, 439–450. Springer. (cited on page 52)

82. Iwama, K. and Miyazaki, S., 2008. A Survey of the Stable Marriage Problem and Its Variants. In *Proceedings of the 2008 International Conference on Informatics Education and Research for Knowledge-Circulating Society*, 131–136. IEEE. (cited on page 52)

83. Jaro, M. A., 1989. Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84 (1989), 414–420. Taylor & Francis. (cited on page 26)

84. Jitta, A. and Klami, A., 2018. On controlling the size of clusters in probabilistic clustering. In *Proceedings of the 32nd Conference on Artificial Intelligence*, 3350–3357. AAAI. (cited on page 51)

85. Jonas, J., 2006. Threat and Fraud Intelligence, Las Vegas Style. *Security & Privacy*, 4, 6 (2006), 28–34. IEEE. (cited on pages 1 and 32)

86. Kalashnikov, D. and Mehrotra, S., 2006. Domain-Independent Data Cleaning via Analysis of Entity-Relationship Graph. *Transactions on Database Systems (TODS)*, 31, 2 (2006), 716–767. ACM. (cited on pages 1, 5, 16, 17, 40, 41, 44, 47, 57, 60, 81, 82, 83, 84, 85, 95, 96, 104, 125, 129, 140, 146, 149, and 150)

87. Kejriwal, M. and Miranker, D. P., 2013. An Unsupervised Algorithm for Learning Blocking Schemes. In *Proceedings of the 13th International Conference on Data Mining (ICDM)*, 340–349. IEEE. (cited on pages 77, 79, and 152)

88. Kim, H. and Lee, D., 2007. Parallel Linkage. In *Proceedings of the 16th Conference on Information and Knowledge Management (CIKM)*, 283–292. ACM. (cited on pages 38, 62, and 77)

89. Kim, S.; Nowozin, S.; Kohli, P.; and Yoo, C. D., 2011. Higher-Order Correlation Clustering for Image Segmentation. In *Proceedings of the 25th Conference on Advances in Neural Information Processing Systems (NIPS)*, 1530–1538. Curran Associates. (cited on page 55)

90. Király, Z., 2013. Linear Time Local Approximation Algorithm for Maximum Stable Marriage. *Algorithms*, 6, 3 (2013), 471–484. Multidisciplinary Digital Publishing Institute. (cited on page 52)

91. Kirby, G.; Carson, J.; Dunlop, F.; Dibben, C.; Dearle, A.; Williamson, L.; Garrett, E.; and Reid, A., 2015. Automatic Methods for Coding Historical Occupation Descriptions to Standard Classifications. In *Population Reconstruction*, 43–60. Springer. (cited on pages 33 and 147)

92. Kirby, G.; de Kerckhove, C.; Shumailov, I.; Carson, J.; Dearle, A.; Dibben, C.; and Williamson, L., 2014. Comparing Relational and Graph Databases for Pedigree Datasets. *Proceedings Workshop Population Reconstruction*, (2014). (cited on page 46)

93. Kirby, G.; Derkani, M. H.; Dearle, A.; Carson, J.; Dunlop, F.; Dibben, C.; and Williamson, L., 2015. Automatic extraction of multiple underlying causes from textual death records. In *Proceedings of the Farr Institute 1st International Conference: Data Intensive Health Research and Care*. Farr Institute. (cited on page 33)

94. Kirsten, T.; Kolb, L.; Hartung, M.; Gross, A.; Köpcke, H.; and Rahm, E., 2010. Data Partitioning for Parallel Entity Matching. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 3, 2 (2010). (cited on pages 38 and 49)

95. Klami, A. and Jitta, A., 2016. Probabilistic Size-constrained Microclustering. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press. (cited on page 50)

96. Kok, S. and Domingos, P., 2005. Learning the Structure of Markov Logic Networks. In *Proceedings of the 22nd International Conference on Machine learning (ICML)*, 441–448. ACM. (cited on page 43)

97. Kok, S.; Singla, P.; Richardson, M.; Domingos, P.; Sumner, M.; Poon, H.; and Lowd, D., 2005. The Alchemy system for statistical relational AI. (2005). University of Washington, Seattle. (cited on page 44)

98. Kolb, L. and Rahm, E., 2013. Parallel Entity Resolution with Dedoop. *Datenbank-Spektrum*, 13, 1 (2013), 23–32. Springer. (cited on page 38)

99. Köpcke, H. and Rahm, E., 2010. Frameworks for entity matching: A comparison. *Data and Knowledge Engineering*, 69, 2 (2010), 197–210. Elsevier. (cited on pages 38 and 48)

100. Köpcke, H.; Thor, A.; and Rahm, E., 2010. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 3, 1-2 (2010), 484–493. (cited on page 55)

101. Kothari, C. R., 2004. *Research methodology: Methods and techniques*. New Age International. (cited on page 8)

102. Kouki, P.; Pujara, J.; Marcum, C.; Koehly, L.; and Getoor, L., 2017. Collective Entity Resolution in Familial Networks. In *Proceedings of the 17th International Conference on Data Mining (ICDM)*, 227–236. IEEE. (cited on page 47)

103. Kouki, P.; Pujara, J.; Marcum, C.; Koehly, L.; and Getoor, L., 2018. Collective entity resolution in multi-relational familial networks. *Knowledge and Information Systems*, (2018), 1–35. Springer. (cited on pages 47, 82, and 104)

104. Lait, A. and Randell, B., 1993. An Assessment of Name Matching Algorithms. Technical report, Department of Computer Science, University of Newcastle upon Tyne. (cited on pages 24 and 108)

105. Levenshtein, V. I., 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, vol. 10, 707–710. (cited on pages 17 and 25)

106. Li, P.; Dong, X.; Maurino, A.; and Srivastava, D., 2011. Linking Temporal Records. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 4, 11 (2011), 956–967. (cited on pages 1, 4, 5, 6, 7, 46, 81, 104, 149, and 150)

107. Liang, H.; Wang, Y.; Christen, P.; and Gayler, R., 2014. Noise-Tolerant Approximate Blocking for Dynamic Real-Time Entity Resolution. In *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 449–460. Springer. (cited on pages 23, 48, and 108)

108. MacKay, D. J., 1992. Information-Based Objective Functions for Active Data Selection. *Neural computation*, 4, 4 (1992), 590–604. MIT Press. (cited on page 53)

109. Maidasani, H.; Namata, G.; Huang, B.; and Getoor, L., 2012. Entity resolution evaluation measures. (2012). (cited on page 55)

110. Malinen, M. and Fränti, P., 2014. Balanced *K*-Means for Clustering. In *Proceedings of the 9th Joint International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, 32–41. Springer. (cited on page 50)

111. Malmi, E.; Gionis, A.; and Solin, A., 2018. Computationally Inferred Genealogical Networks Uncover Long-Term Trends in Assortative Mating. *arXiv preprint arXiv:1802.06055*, (2018). (cited on pages 47, 82, and 104)

112. McCallum, A.; Nigam, K.; and Ungar, L. H., 2000. Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining (KDD)*, 169–178. ACM. (cited on pages 23 and 48)

113. McCalman, J.; Smith, L.; Silcot, S.; and Kippen, R., 2015. Building a Life Course Dataset from Australian Convict Records: Founders & Survivors: Australian Life Courses in His-

torical Context, 1803–1920. In *Population Reconstruction*, 285–298. Springer. (cited on page 130)

114. McConville, R.; Liu, W.; and Hong, J., 2017. Vertex Deduplication Based on String Similarity and Community Membership. In *Proceedings of the 6th International Workshop on Complex Networks and their Applications*, 178–189. Springer. (cited on page 45)

115. McGrath, P., 2016. Veda Advantage accused of providing incorrect credit information, refusing to fix errors. Https://www.abc.net.au/news/2016-05-25/veda-advantage-provided-incorrect-credit-reports/7444392 Accessed: 2018-11-18. (cited on page 32)

116. Menestrina, D.; Whang, S.; and Garcia-Molina, H., 2010. Evaluating Entity Resolution Results. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 3, 1–2 (2010), 208–219. (cited on page 55)

117. Munera, D.; Diaz, D.; Abreu, S.; Rossi, F.; Saraswat, V. A.; and Codognet, P., 2015. Solving Hard Stable Matching Problems via Local Search and Cooperative Parallelization. In *Proceedings of the 29th Conference on Artificial Intelligence*, 1212–1218. AAAI. (cited on page 52)

118. Nanayakkara, C.; Christen, P.; and Ranbaduge, T., 2018. Temporal graph-based clustering for historical record linkage. *arXiv preprint arXiv:1807.02262*, (2018). (cited on page 45)

119. Naumann, F. and Herschel, M., 2010. *An Introduction to Duplicate Detection*, vol. 3 of *Synthesis Lectures on Data Management*. Morgan and Claypool. (cited on pages 15, 17, 25, 26, and 38)

120. Newcombe, H. and Kennedy, J., 1962. Making Maximum Use of the Discriminating Power of Identifying Information. *Communications of the ACM*, 5, 11 (1962), 563–566. ACM. (cited on page 37)

121. Newcombe, H.; Kennedy, J.; Axford, S.; and James, A., 1959. Automatic Linkage of Vital Records. *Science*, 130, 3381 (1959), 954–959. American Association for the Advancement of Science. (cited on page 37)

122. Niu, F.; Ré, C.; Doan, A.; and Shavlik, J., 2011. Tuffy: Scaling up Statistical Inference in

Markov Logic Networks using an RDBMS. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 4, 6 (2011), 373–384. (cited on page 44)

123. Nuray-Turan, R.; Kalashnikov, D. V.; and Mehrotra, S., 2013. Adaptive Connection Strength Models for Relationship-Based Entity Resolution. *Journal of Data and Information Quality (JDIQ)*, 4, 2 (2013), 8. ACM. (cited on pages 40 and 60)

124. Odell, M. and Russell, R., 1918. The Soundex Coding System. *US Patents*, 1261167 (1918). (cited on pages 24, 60, and 108)

125. O'Hare, K.; Jurek, A.; and de Campos, C., 2018. A New Technique of Selecting an Optimal Blocking Method for Better Record Linkage. *Information Systems*, (2018), 151–166. Elsevier. (cited on page 55)

126. On, B.-W.; Koudas, N.; Lee, D.; and Srivastava, D., 2007. Group Linkage. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE)*, 496–505. IEEE. (cited on pages 1, 5, 44, 129, 149, and 150)

127. Pandove, D.; Goel, S.; and Rani, R., 2018. Correlation clustering methodologies and their fundamental results. *Expert Systems*, 35, 1 (2018), e12229. Wiley Online Library. (cited on page 55)

128. Papadakis, G.; Ioannou, E.; Niederée, C.; Palpanas, T.; and Nejdl, W., 2011. Eliminating the Redundancy in Blocking-based Entity Resolution Methods. In *Proceedings of the 11th International Conference on Digital Libraries*, 85–94. ACM. (cited on page 49)

129. Papadakis, G.; Ioannou, E.; Niederée, C.; Palpanas, T.; and Nejdl, W., 2012. Beyond 100 Million Entities: Large-scale Blocking-based Resolution for Heterogeneous Data. In *Proceedings of the 5th International Conference on Web Search and Data Mining (WSDM)*, 53–62. ACM. (cited on page 49)

130. Papadakis, G.; Ioannou, E.; Palpanas, T.; Niederee, C.; and Nejdl, W., 2013. A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces. *Transactions on Knowledge and Data Engineering (TKDE)*, 25, 12 (2013), 2665–2682. IEEE. (cited on page 49)

131. Papadakis, G.; Koutrika, G.; Palpanas, T.; and Nejdl, W., 2014. Meta-Blocking: Taking

Entity Resolution to the Next Level. *Transactions on Knowledge and Data Engineering (TKDE)*, 26, 8 (2014), 1946–1960. IEEE. (cited on page 49)

132. Papadakis, G. and Palpanas, T., 2016. Blocking for Large-Scale Entity Resolution: Challenges, Algorithms, and Practical Examples. In *Proceedings of the 32nd International Conference on Data Engineering (ICDE)*, 1436–1439. IEEE. (cited on page 48)

133. Papadakis, G.; Papastefanatos, G.; Palpanas, T.; and Koubarakis, M., 2016. Scaling Entity Resolution to Large, Heterogeneous Data with Enhanced Meta-blocking. In *Proceedings of the 19th International Conference on Extending Database Technology (EDBT)*, 221–232. (cited on page 49)

134. Papadakis, G.; Svirsky, J.; Gal, A.; and Palpanas, T., 2016. Comparative Analysis of Approximate Blocking Techniques for Entity Resolution. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 9, 9 (2016), 684–695. (cited on pages 29 and 48)

135. Philips, L., 2000. The Double-Metaphone Search Algorithm. *C/C++ User's Journal*, 18, 6 (2000). CMP Media. (cited on pages 24, 60, and 108)

136. Poon, H. and Domingos, P., 2006. Sound and Efficient Inference with Probabilistic and Deterministic Dependencies. In *Proceedings of the 21st Conference on Artificial Intelligence*, vol. 6, 458–463. AAAI. (cited on page 42)

137. Poon, H. and Domingos, P., 2007. Joint Inference in Information Extraction. In *Proceedings of the 22nd Conference on Artificial Intelligence*, vol. 7, 913–918. AAAI. (cited on page 42)

138. Poon, H. and Domingos, P., 2008. Joint Unsupervised Coreference Resolution with Markov Logic. In *Proceedings of the 13th Conference on Empirical Methods in Natural Language Processing*, 650–659. Association for Computational Linguistics. (cited on page 42)

139. Prasad, K.; Faruquie, T.; Joshi, S.; Chaturvedi, S.; Subramaniam, L.; and Mohania, M., 2009. Data Cleansing Techniques for Large Enterprise Datasets. In *Proceedings of the 1st Service Research and Innovation Institute (SRII) Global Conference*, 135–144. IEEE. (cited on page 17)

140. Qian, K.; Popa, L.; and Sen, P., 2017. Active Learning for Large-Scale Entity Resolution. In *Proceedings of the 26th Conference on Information and Knowledge Management (CIKM)*, 1379–1388. ACM. (cited on page 54)

141. Rahm, E., 2014. Discovering Product Counterfeits in Online Shops: A Big Data Integration Challenge. *Journal of Data and Information Quality (JDIQ)*, 5, 1-2 (2014), 3. ACM. (cited on page 32)

142. Rahmani, H.; Ranjbar-Sahraei, B.; Weiss, G.; and Tuyls, K., 2016. Entity resolution in disjoint graphs: An application on genealogical data. *Intelligent Data Analysis*, 20, 2 (2016), 455–475. IOS Press. (cited on page 47)

143. Ramadan, B. and Christen, P., 2014. Forest-Based Dynamic Sorted Neighborhood Indexing for Real-Time Entity Resolution. In *Proceedings of the 23rd Conference on Information and Knowledge Management (CIKM)*, 1787–1790. ACM. (cited on page 49)

144. Ramadan, B. and Christen, P., 2015. Unsupervised Blocking Key Selection for Real-Time Entity Resolution. In *Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 574–585. Springer. (cited on pages 77, 79, and 152)

145. Ramadan, B.; Christen, P.; Liang, H.; and Gayler, R. W., 2015. Dynamic Sorted Neighborhood Indexing for Real-Time Entity Resolution. *Journal of Data and Information Quality (JDIQ)*, 6, 4 (2015), 15. ACM. (cited on pages 38, 48, 49, 61, 76, 79, and 152)

146. Rastogi, V.; Dalvi, N.; and Garofalakis, M., 2011. Large-Scale Collective Entity Matching. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 4 (2011), 208–218. (cited on pages 1, 43, and 44)

147. Rebollo-Monedero, D.; Solé, M.; Nin, J.; and Forné, J., 2013. A modification of the *k*-means method for quasi-unsupervised learning. *Knowledge-Based Systems (KBS)*, 37, 0 (2013), 176 – 185. Elsevier. (cited on page 50)

148. Reid, A.; Davies, R.; and Garrett, E., 2002. Nineteenth-Century Scottish Demography From Linked Censuses and Civil Registers: A 'Sets of Related Individuals' Approach. *History and Computing*, 14, 1-2 (2002), 61–86. Edinburgh University Press. (cited on pages 17, 34, 131, and 132)

149. Reid, A. and Garrett, E., 2012. Doctors and the causes of neonatal death in Scotland in the second half of the nineteenth century. *Annales de démographie historique*, 123, 1 (2012), 149–179. Editions Belin. (cited on page 130)

150. Richardson, M. and Domingos, P., 2006. Markov logic networks. *Machine Learning*, 62, 1-2 (2006), 107–136. Springer. (cited on pages 6, 16, 42, 47, 51, 59, 60, 104, and 150)

151. Saeedi, A.; Peukert, E.; and Rahm, E., 2017. Comparative Evaluation of Distributed Clustering Schemes for Multi-source Entity Resolution. In *Proceedings of the 21st European Conference on Advances in Databases and Information Systems*, 278–293. Springer. (cited on page 5)

152. Sarawagi, S., 2008. Information Extraction. *Foundations and Trends in Databases*, 1, 3 (2008), 261–377. Now Publishers. (cited on page 17)

153. Sarawagi, S. and Bhamidipaty, A., 2002. Interactive Deduplication Using Active Learning. In *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining (KDD)*, 269–278. ACM. (cited on page 53)

154. Scheffer, T.; Decomain, C.; and Wrobel, S., 2001. Active Hidden Markov Models for Information Extraction. In *Proceedings of the 4th International Symposium on Intelligent Data Analysis*, 309–318. Springer. (cited on pages 54, 92, and 93)

155. Schnell, R., 2016. Privacy-preserving record linkage. In *Methodological Developments in Data Linkage*, 201–225. Wiley Online Library. (cited on page 38)

156. Schraagen, M., 2014. Historical record linkage using event sequence consistency. In *Proceedings Workshop on Population Reconstruction*. (cited on pages 5 and 86)

157. Schraagen, M. and Kosters, W., 2014. Record Linkage Using Graph Consistency. In *Proceedings of the 10th International Conference on Machine Learning and Data Mining in Pattern Recognition*, 471–483. Springer. (cited on pages 5, 47, 129, 130, and 149)

158. Sehgal, V.; Getoor, L.; and Viechnicki, P. D., 2006. Entity Resolution in Geospatial Data Integration. In *Proceedings of the 14th International Symposium on Advances in Geographic Information Systems*, 83–90. ACM. (cited on pages 28 and 104)

159. Settles, B., 2010. Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin, Madison. (cited on pages 7, 13, and 22)

160. Settles, B., 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6, 1 (2012), 1–114. Morgan & Claypool. (cited on pages 53 and 92)

161. Settles, B. and Craven, M., 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the 13th Conference on Empirical Methods in Natural Language Processing*, 1070–1079. Association for Computational Linguistics. (cited on pages 53, 54, and 92)

162. Shen, Z. and Wang, Q., 2014. Entity Resolution with Weighted Constraints. In *Proceedings of the 18th East European Conference on Advances in Databases and Information Systems*, 308–322. Springer. (cited on page 52)

163. Singla, P. and Domingos, P., 2005. Discriminative Training of Markov Logic Networks. In *Proceedings of the 20th Conference on Artificial Intelligence*, vol. 5, 868–873. AAAI. (cited on page 43)

164. Singla, P. and Domingos, P., 2005. Object Identification with Attribute-Mediated Dependences. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 297–308. Springer. (cited on page 55)

165. Singla, P. and Domingos, P., 2006. Entity Resolution with Markov Logic. In *Proceedings of the 6th International Conference on Data Mining (ICDM)*, 572–582. IEEE. (cited on pages 1, 2, 4, 5, 17, 34, 42, 43, 44, 81, 82, 125, 129, and 149)

166. Smalheiser, N. R. and Torvik, V. I., 2009. Author name disambiguation. *Annual Review of Information Science and Technology*, 43, 1 (2009), 1–43. Wiley Online Library. (cited on pages 17, 33, and 109)

167. Smith-Christmas, C. and Smakman, D., 2009. Gaelic on the Isle of Skye: older speakers' identity in a language-shift situation. *International Journal of the Sociology of Language*, 2009, 200 (2009), 27–47. Walter de Gruyter. (cited on page 130)

168. Sokolova, M. and Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45, 4 (2009), 427–437. Elsevier. (cited on pages 29 and 30)

169. Spärck Jones, K., 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28, 1 (1972), 11–21. MCB University Press. (cited on page 27)

170. Steorts, R. C.; Ventura, S. L.; Sadinle, M.; and Fienberg, S. E., 2014. A Comparison of Blocking Methods for Record Linkage. In *Proceedings of the 6th International Conference on Privacy in Statistical Databases*, 253–268. Springer. (cited on page 48)

171. Suri, S. and Vassilvitskii, S., 2011. Counting Triangles and the Curse of the Last Reducer. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, 607–614. ACM. (cited on page 123)

172. Talburt, J., 2011. *Entity Resolution and Information Quality*. Morgan Kaufmann. (cited on page 38)

173. Tauer, G.; Date, K.; Nagi, R.; and Sudit, M., 2019. An incremental graph-partitioning algorithm for entity resolution. *Information Fusion*, 46 (2019), 171–183. Elsevier. (cited on pages 38, 61, 79, and 152)

174. Thorvaldsen, G.; Andersen, T.; and Sommerseth, H. L., 2015. Record Linkage in the Historical Population Register for Norway. In *Population Reconstruction*, 155–171. Springer. (cited on page 130)

175. Torres, C. and Dillon, L. Y., 2015. Using the Canadian Censuses of 1852 and 1881 for Automatic Data Linkage: A Case Study of Intergenerational Social Mobility. In *Population Reconstruction*, 243–261. Springer. (cited on page 130)

176. Vatsalan, D. and Christen, P., 2013. Sorted Nearest Neighborhood Clustering for Efficient Private Blocking. In *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 341–352. Springer. (cited on page 61)

177. Vatsalan, D.; Christen, P.; and Verykios, V. S., 2013. A taxonomy of privacy-preserving record linkage techniques. *Information Systems*, 38, 6 (2013), 946–969. Elsevier. (cited on pages 38, 61, 62, and 77)

178. Vatsalan, D.; Sehili, Z.; Christen, P.; and Rahm, E., 2017. Privacy-Preserving Record Linkage for Big Data: Current Approaches and Research Challenges. In *Handbook of Big Data Technologies*, 851–895. Springer. (cited on page 38)

179. Verroios, V. and Garcia-Molina, H., 2017. Top-K Entity Resolution with Adaptive Locality-Sensitive Hashing. Technical report, Stanford InfoLab. (cited on page 49)

180. Vesdapunt, N.; Bellare, K.; and Dalvi, N., 2014. Crowdsourcing Algorithms for Entity Resolution. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 7, 12 (2014), 1071–1082. (cited on page 54)

181. Vieira, P.; Salgado, A. C.; and Lóscio, B. F., 2016. A Dynamic Indexing for Incremental Entity Resolution over Query Results. *International Journal of Computational Linguistics Research*, 7, 3 (2016), 92–103. Digital Information Research Foundation. (cited on page 49)

182. Wang, H.; Li, J.; and Gao, H., 2015. Efficient entity resolution based on subgraph cohesion. *Knowledge and Information Systems*, (2015), 1–30. Springer. (cited on pages 5 and 55)

183. Wang, J.; Kraska, T.; Franklin, M. J.; and Feng, J., 2012. CrowdER: Crowdsourcing Entity Resolution. *Proceedings of the Very Large Data Bases Endowment (VLDB)*, 5, 11 (2012), 1483–1494. (cited on page 54)

184. Wang, Q.; Vatsalan, D.; and Christen, P., 2015. Efficient Interactive Training Selection for Large-Scale Entity Resolution. In *Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 562–573. Springer. (cited on pages 43, 53, and 54)

185. Weikum, G.; Ntarmos, N.; Spaniol, M.; Triantafillou, P.; Benczúr, A. A.; Kirkpatrick, S.; Rigaux, P.; and Williamson, M., 2011. Longitudinal Analytics on Web Archive Data: It's About Time! In *Proceedings of the 5th Conference on Innovative Data Systems Research*, 199–202. www.cidrdb.org. (cited on page 46)

186. Whang, S. E.; Marmaros, D.; and Garcia-Molina, H., 2013. Pay-As-You-Go Entity Resolution. *Transactions on Knowledge and Data Engineering (TKDE)*, 25, 5 (2013), 1111–1124. IEEE. (cited on page 28)

187. Whang, S. E.; Menestrina, D.; Koutrika, G.; Theobald, M.; and Garcia-Molina, H., 2009. Entity Resolution with Iterative Blocking. In *Proceedings of the 35th International Conference on Management of Data (ICMD)*, 219–232. ACM. (cited on page 48)

188. Winkler, W., 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. In *Proceedings of the Section on Survey Research Methods*, 354–359. American Statistical Association. (cited on pages 17 and 26)

189. Winkler, W. E., 2006. Overview of Record Linkage and Current Research Directions.

Technical Report RR2006/02, US Bureau of the Census, Washington, DC. (cited on page 38)

190. Xu, R. and Wunsch II, D., 2005. Survey of Clustering Algorithms. *Transactions on Neural Networks*, 16, 3 (May 2005), 645–678. IEEE. (cited on page 66)

191. Xu, Z.; Akella, R.; and Zhang, Y., 2007. Incorporating Diversity and Density in Active Learning for Relevance Feedback. In *Proceedings of the 29th European Conference on Information Retrieval*, 246–257. Springer. (cited on pages 53 and 92)

192. Yang, Y.; Ma, Z.; Nie, F.; Chang, X.; and Hauptmann, A. G., 2015. Multi-Class Active Learning by Uncertainty Sampling with Diversity Maximization. *International Journal of Computer Vision*, 113, 2 (2015), 113–127. Springer. (cited on page 7)

193. Zhu, J.; Wang, H.; Tsou, B. K.; and Ma, M. Y., 2010. Active Learning With Sampling by Uncertainty and Density for Data Annotations. *Transactions on Audio, Speech & Language Processing*, 18, 6 (2010), 1323–1331. IEEE. (cited on page 7)

194. Zhu, S.; Wang, D.; and Li, T., 2010. Data clustering with size constraints. *Knowledge-Based Systems*, 23, 8 (2010), 883–889. Elsevier. (cited on page 50)