# Embedding Techniques to Solve Large-scale Entity Resolution

Samudra Herath

March 27, 2022

*Thesis submitted for the degree of*
*Doctor of Philosophy*
*in*
*Mathematics and Computer Science*
*at The University of Adelaide*
*Faculty of Engineering, Computer and Mathematical Sciences*
*School of Mathematical Sciences*


THE UNIVERSITY
*of* ADELAIDE

# Contents

# List of Tables

# List of Figures

# Signed Statement

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint award of this degree.

The author acknowledges that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Signed: . ..... Date: .....27/03/2022.....

# Acknowledgements

The work carried through this thesis is not possible without the guidance and help of the people who contribute their valuable time and resources to complete this work successfully.

First of all, I would like to give my heartiest gratitude to my supervisors Prof. Matthew Roughan and A/Prof. Gary Glonek, for their incredible encouragement and support throughout my PhD. This thesis would not have existed without their collaboration, advice, and inspiration.

My special thanks go to Matt, my principal supervisor, for being patient with me in listening and sharing his knowledge throughout the time. I am fortunate and privileged to spend all these years talking and learning from someone remarkably insightful, knowledgeable and generous.

Also, I would like to thank Gary, my secondary supervisor, for his continued support over the past four years by providing necessary domain knowledge whenever needed. Gary is my go-to person, who ultimately knows the answers to every question and who always help confused students like me figure out research.

I would also like to acknowledge Prof. Erhard Rahm for his valuable feedback during the initial stages of this research. My thanks also go to Prof. Peter Christen for inspiring me with his incredible work in data linkage. Thank you to Dr Asanga for sharing many valuable perspectives (and being the devil in detail) and Dr Selasi for being keen to discuss my research.

Finally, I express my sincere thanks to my family and friends for supporting me, caring so deeply for me, and believing in me in all these years. And thank you so much, Pasan, for being patient, supportive and constant for me every day. I do not know how I would have done it without you.

The work in this thesis would not have been possible without the tremendous support of all of you.

# Abstract

Entity resolution (ER) identifies and links records that belong to the same real-world entities, where an entity refer to any real-world object. It is a primary task in data integration. Accurate and efficient ER substantially impacts various commercial, security, and scientific applications. Often, there are no unique identifiers for entities in datasets/databases that would make the ER task easy. Therefore record matching depends on entity identifying attributes and approximate matching techniques. The issues of efficiently handling large-scale data remain an open research problem with the increasing volumes and velocities in modern data collections. Fast, scalable, real-time and approximate entity matching techniques that provide high-quality results are highly demanding. This thesis proposes solutions to address the challenges of lack of test datasets and the demand for fast indexing algorithms in large-scale ER.

The shortage of large-scale, real-world datasets with ground truth is a primary concern in developing and testing new ER algorithms. Usually, for many datasets, there is no information on the ground truth or 'gold standard' data that specifies if two records correspond to the same entity or not. Moreover, obtaining test data for ER algorithms that use personal identifying keys (e.g., names, addresses) is difficult due to privacy and confidentiality issues. Towards this challenge, we proposed a numerical simulation model that produces realistic large-scale data to test new methods when suitable public datasets are unavailable. One of the important findings of this work is the approximation of vectors that represent entity identification keys and their relationships, e.g., dissimilarities and errors.

Indexing techniques reduce the search space and execution time in the ER process. Based on the ideas of the approximate vectors of entity identification keys, we proposed a fast indexing technique (Em-K indexing) suitable for real-time, approximate entity matching in large-scale ER. Our Em-K indexing method provides a quick and accurate block of candidate matches for a querying record by searching an existing reference database. All our solutions are metric-based. We transform metric or non-metric spaces to a lower-dimensional Euclidean space, known as configuration space, using multidimensional scaling (MDS). This thesis discusses how to modify MDS algorithms to solve various ER problems efficiently. We proposed highly efficient and scalable approximation methods that extend the MDS algorithm for large-scale datasets.

We empirically demonstrate the improvements of our proposed approaches on several datasets with various parameter settings. The outcomes show that our methods can generate large-scale testing data, perform fast real-time and approximate entity matching, and effectively scale up the mapping capacity of MDS.

# Chapter 1

# Introduction

## 1.1 Research Problem

Data is increasingly becoming voluminous, varied and valuable with advances in science and technology and the rapid digital transformation of global industries. A key issue is how to store, process, analyse, interpret, and consume an overwhelming amount of data to make better decisions. Data integration plays a vital role in data analysis and mining projects by combining data from different sources into meaningful information. *Entity resolution* (ER), a core step in data integration, detects records across multiple datasets that correspond to the same real-world entity [4, 5]. It has also been known as the object identity problem, record linkage, the merge/purge problem, deduplication, duplicate record detection, and data matching.

ER has been widely used in commercial and government applications and also in academic and statistical research. Therefore, ER appears as a problem across a range of applications, e.g., medical or epidemiological research, crime detection and national security, tax and other fraud detection, education research, bibliographic database construction, e-commerce applications, and customer relationship management [1].

An *entity* can be a person, place, product, organisation or any real-world object with a unique identity that distinguishes it from all other entities of the same type. ER applies to various data types, from structured relational databases to unstructured entities extracted from free text [1, 6]. Many of these data contain records of individual entities, e.g., patients, customers and travellers. In this thesis, we consider structured data described by a well-defined schema that resides in relational databases or CSV files.

The problem of combining two or more separately recorded measures that belong to a particular individual dates back to the early 1950s [7]. Despite extensive research studies over several decades, ER remains a challenging problem in practice, especially for applications in large-scale data.

This thesis is aimed at improving state of art in ER for large-scale datasets. It mainly

addresses the problems with the scarcity of test data and the demand for fast indexing algorithms. One of the major problems that impede the development of large-scale ER algorithms is the shortage of testing data.

Therefore, our goal was to develop a numerical simulation model that generates large synthesis datasets for large-scale testing of ER techniques. Indexing or blocking techniques reduce the search space and execution time in ER. However, many existing techniques are not suitable for applications requiring online capabilities, especially those that involve big, fast, or streaming data. We aimed at developing fast indexing algorithms suitable for real-time and approximate entity matching in large-scale ER.

## 1.2 Motivation

This section discusses the challenges in large-scale ER that motivate the work presented in this thesis.

- Approximate matching

  Often, there are no unique identifiers for entities that would make record matching easy in ER databases/datasets. Therefore, entity record matching relies on the common entity attributes across databases that may serve as quasi-identifiers [8]. For instance, quasi-identifiers of a database that contains personal information include name, address, and date of birth.

  Exact matching between those attribute values is often not possible due to various data quality issues, e.g., unexpected errors (such as typos, phonetic errors), various customs (the use of abbreviations), and unavoidable changes over time (such as addresses and family names for women after marriage). As a result, many ER approaches apply approximate matching techniques when identifying different descriptions of a real-world entity.

  Approximate matching techniques use pairwise comparisons between record attributes to identify similar records. Pairwise comparisons in ER constitute an inherently quadratic task. Naive ER algorithms have a quadratic running time, which is computationally prohibitive for large-scale data collections.

- Lack of testing data

  Massive amounts of data collected by government and business organisations are about people and their surroundings. Hence, most application-specific data (e.g., medical, employment, and customer records) are stored together with personal information (e.g., name, address, gender, age). Linking entity records based on personal information occurs in various ER application domains, including health informatics, national security, fraud and crime detection, online shopping, social sciences [1].

Quasi-identifiers such as gender and postcode can be easy to obtain. However, privacy and confidentiality are great concerns when more sensitive quasi-identifiers, e.g., names and addresses, are used in data analysis. Hence, sharing this information between organisations or making them publicly available is often impossible. Even if we have access to these sorts of data, there are other challenges as well. For instance, the ground truth or gold stand that specifies the matching status of records in datasets is not available [1]. Therefore, suitable public datasets, especially those containing personal information for linking, are restricted in this area and usually small in size.

Artificially simulated test data is the alternate solution for testing many ER algorithms. Testing data provide a gold standard by labelling duplicate data in a dataset while resembling the required real-life properties as well as possible. However, many existing solutions that generate test data struggle to generate realistic data (e.g., real-world errors) and only generate small datasets [9]. This creates a fundamental problem of ER research: the lack of large-scale, real-world datasets to test new ER algorithms.

- Large volumes of data

  ER process becomes quadratically more complex and time-consuming with large-scale data volumes. The asymptotic time complexities of most traditional ER algorithms are $O(N^2)$ (given $N$ records) and require a significant amount of processing capability for each comparison. Hence, large-scale ER solutions are needed when scaling to hundreds of millions of entity references.

  Despite the long development history of ER in research, some algorithms are still in use today that are inefficient for large scale data. For instance, the Fellegi-Sunter model [4] proposed in 1969 implements pairwise logic, where each attribute of a record is potentially compared with every other to determine similar records. Therefore, the resources required for applying this model to large-scale data exceeds the capabilities of modern supercomputers [10]. For instance, consider an ER task that identifies all matching records in two datasets containing 100 million records each. The pairwise distances calculated among the two datasets are $10^8 \times 10^8$, which is prohibitive.

  Indexing or blocking plays a major role in scaling ER solutions to large-scale data. Indexing addresses the problems of inefficiencies that occur due to the excessive number of pairwise comparisons. However, many existing indexing techniques still have quadratic time complexity and are inefficient to deal with very large-scale data.

- High velocity data

  In addition to the volume, modern data brings many other challenges to ER research, e.g., semi-structured data, high-velocity streams and complex similarity metrics. Ve-

locity refers to the rate of data generated at high speed, and high-velocity streams require real-time ER solutions [2]. Real-time ER demands higher efficiency compared to traditional online solutions. Entities provided as queries may arrive in high-velocity streams to resolve against a known entity collection [6]. Some of these applications may also require incremental strategies to update linkage results when data updates arrive [11]. Hence, query-based or streaming data execution strategies needs to be implemented in real-time ER solutions.

- Query matching

  Many applications require real-time ER where query records must be matched to a database of known entities, with many organisations moving online [12, 13]. For instance, consider the real-world application on the real-time matching of personal information from consumer credit applications given in [1]. Similarly, other examples can be found in online financial or shopping transactions, telecommunication records, or data from the multitude of increasingly online sensors (internet of things).

  These applications process large-scale data that arrive on high-velocity streams, and ER solutions must provide prompt responses. Queries that require prompt responses against existing datasets have applications in various domains, e.g., online government services, credit applications, and law enforcement [1].

  Traditional ER solutions often process databases offline in batch mode, and no further action is required once a pair of matches are determined. Hence, these techniques are not suitable for applications that require real-time query matching, especially if they are processing large-scale or streaming data.

  Usually, the databases involved in query matching are large-scale; hence sequential querying for entity matching to provide prompt responses is difficult. Scalability is an issue for real-time settings in the ER process and requires suitable tools and techniques to improve efficiency. Hence, fast indexing methods that provide approximate and real-time entity query matching is needed.

In the following, we introduce novel approaches that address the above challenges for large-scale data. We have explored several statistical and machine learning tools and techniques to provide solutions.

## 1.3 Summary of Our Approach

Our approach to large-scale ER goes beyond those presented in the literature in the following ways to address the challenges presented in the previous section. The proposed solutions focus on large-scale ER that perform matching or linking based on personal information, e.g., names and addresses.

Many ER datasets represent entity records using a set of attributes, e.g., names, addresses, and gender. When linking or matching the entity records, the distances or dissimilarities between attribute values are the main property of interest. Similar attribute values have small distances, whereas different attribute values have large distances. Following this phenomenon, we were interested in looking at the relationship between entity records mathematically.

We were particularly interested in name matching in ER since *names* are the most important quasi-identifier in many datasets, e.g., health records, census data, police reports, tweets, bibliography collections, Linkedin and Facebook posts, and customer reviews [14]. Therefore names are heavily used in entity matching. Since many ER datasets consist of strings, e.g., names, addresses, and occupations, studies of name strings can be easily used to interpret relationships between other string attributes.

We assume names as objects reside in a complicated high-dimensional space and call this the *namespace*. Similar names are closer together in this space, while different names are further apart. We can easily explain the mathematical relationships between names if we move the namespace to a metric space. Hence, we reduce the complexity of the namespace by approximating the distance between names in a metric space; accordingly, representing names as vectors. The distances between vectors approximate the distances or dissimilarities measured in the namespace.

The approximate vector representation of names is the key fundamental concept of the ideas, tools and techniques proposed in this thesis. In particular, the key application outputs are,

- The ability to synthesis large-scale test data.
- The ability to apply efficient searching (e.g., Kd-trees) for approximate entity record matching.

The concept of name approximation in a metric space also leads to specific research problems to solve.

1. How can we construct vector representations and increase the accuracy of the name approximation?

    - We mainly used multidimensional scaling (MDS) algorithms (a standard embedding technique) to explore the name approximation in a metric space.

2. How can we develop tools that use these approximations more efficiently for large-scale data?

    - We used the concepts of landmarks, out-of-sample embedding and artificial neural networks to propose efficient and scalable ER solutions based on metric space for large-scale data.

We used a range of other statistical and machine learning techniques to construct our algorithms and tools (details are in Chapter 3). In the following section, we summarise our contributions in this thesis.

### 1.3.1 Contributions

This thesis provides a detailed study of large-scale ER. It proposes novel solutions for large-scale ER, addressing several gaps in ER research. The thesis mainly covers:

1. A simple, inexpensive, and fast synthetic data generation (Chapter 4):

   As discussed in Chapter 4, many existing simulation tools in ER lack support for generating large-scale data and have problems in complexity, scalability, and limitations of resampling. In our work, we propose a simple, inexpensive, and fast synthetic data generation tool. We avoid the detail-level simulation of entity names using a simple vector representation that delivers simplicity and efficiency. The key idea is to simulate vectors called *name-like vectors* that approximate the properties of names, particularly the distances between name strings. Hence, our model generates name-like vectors and their errors using name dissimilarities of an actual namespace. Using our model, we can produce realistic large-scale test data to develop new methods when suitable public data is unavailable. We are the first to propose such a numerical simulation model to generate data needed for large-scale ER.

2. Em-K indexing for query matching in ER (Chapter 3 and Chapter 5):

   As discussed in Chapter 5, ER solutions that support online capabilities are in demand. Towards the challenges of real-time and approximate query matching, we propose the *Em-K indexing*, a fast indexing method for query matching. The method transforms a set of records into vectors in a metric space, precisely a lower-dimensional Euclidean space. Then a k-nearest neighbour search is performed to create blocks of similar records for querying entities in the Euclidean space using a Kd-tree data structure. Our method can process a stream of queries against a large-scale data set within a fixed period. By doing so, we obtain as many of the matching records as possible where the processing time of a single query takes a fixed sub-second time. The technique is robust to noisy data that contains errors and allows fast approximate query matching.

   Several metric space-based ER solutions are used in the literature mainly to link a fixed set of entity records [15, 16]. While non of them address indexing or query matching, our method stands out as a solution that simultaneously addresses large-scale, real-time and approximate query matching.

3. Out-of-sample embedding techniques (Chapter 3 and Chapter 6):

   MDS configure a given high-dimensional data into a target dimension [17]. We mainly use least-squares multidimensional scaling (LSMDS) to embed non-metric or metric spaces into a configuration space (described in Chapter 3). For instance, we apply LSMDS when approximating dissimilarities between entity names in a Euclidean space. MDS techniques, including LSMDS, do not offer an explicit function that maps new data into the configuration space without extensively reconstructing the entire configuration from the augment dataset. This is known as the out-of-sample embedding (OSE) problem, which limits the capabilities of LSMDS.

   To address this shortcoming, we propose two OSE approaches; an optimisation method that minimises a cost function to map a new point and an artificial neural network that learns exiting embeddings to predict the mapping of a new point. We used them to improve the efficiency of the Em-K indexing. Both techniques process data very fast with a minor trade-off in the approximation and improve the scalability of the standard LSMDS. However, the results suggested that the neural network model is faster. While existing solutions only consider Euclidean data, our methods can handle both metric and non-metric data as inputs and scale to large-scale MDS problems.

### 1.3.2 Publications

1. Major contributions:

   (a) S. Herath, M. Roughan and G. Glonek, *Generating name-like vectors for testing large-scale entity resolution,* in IEEE Access, vol. 9, pp. 145288-145300, 2021, doi: 10.1109/ACCESS.2021.3122451. Corresponds to Chapter 4 of this thesis.

   (b) S. Herath, M. Roughan and G. Glonek, *Em-K indexing for approximate query matching in large-scale ER,* (Submitted to a data mining journal and the preprint available at: arXiv, 2111.04070). Corresponds to Chapter 5 of this thesis.

   (c) S. Herath, M. Roughan and G. Glonek, *High performance out-of-sample embedding techniques for multidimensional scaling,* (Submitted to a data mining journal and the preprint available at: arXiv, 2111.04067). Corresponds to Chapter 6 of this thesis.

2. Minor contributions:

   (a) S. Herath, M. Roughan and G. Glonek, *Landmarks-based blocking method For Large-scale Entity Resolution,* 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), 2020, pp. 773-774, doi: 10.1109/DSAA49011.2020.00110. Correspond to Chapter 5 of this thesis.

## 1.4 Thesis Outline

The rest of this thesis is organised as follows: Chapter 2 discusses ER preliminaries. It covers background material that contributes to understanding the basic concepts and techniques of ER; it emphasises large-scale ER. Chapter 3 provides the mathematical foundation that is necessary for describing our methodologies. It presents the main ER concepts relevant to our work and the statistical and machine learning tools and techniques used in our contributions with their formal definitions. Chapter 4 introduces the name approximation using vectors. It proposes a simple, inexpensive, and fast synthetic data simulation tool to generate vectors that approximate the properties of an actual namespace. In Chapter 5, we investigate the query matching problem in ER to propose a fast indexing method suitable for approximate and real-time query matching. As a solution, we propose the Em-K indexing technique for query matching to provide a quick and accurate block of candidate matches for a querying record. This chapter also introduces an OSE technique for LSMDS using an optimisation approach. Chapter 6 investigates the OSE problem to propose another solution based on an artificial neural network model that speeds up the calculations. It also compares the two OSE techniques. Finally, Chapter 7 concludes the thesis and provides directions for future work.

## 1.5 Notation and Terminology

Table 1.1 provides a list of general notations and terminology used throughout this thesis. We will introduce other notations specific to individual chapters at the beginning of those relevant chapters.

Table 1.1: Summary of the main notations frequently used in the thesis.

| | |
|---|---|
| $R$ | Set of objects in a metric or non-metric space |
| $n$ | Number of objects in the set $R$ |
| $X$ | Coordinates of $R$ objects in a configuration space |
| $\Delta$ | Dissimilarity matrix of $R$ objects |
| $D$ | Distance matrix of $X$ points in a configuration space |
| $\delta$ | Dissimilarity or distances between $R$ objects |
| $d$ | The Euclidean distances between $X$ coordinates |
| $K$ | Dimension of a multidimensional scaling configuration |
| $E$ | Collection of Entities |
| $Q$ | Stream of queries |
| $B_s$ | Set of blocks generated by an indexing method |
| $L_s$ | Set of landmarks selected from $R$ objects |
| $\sigma_{raw}$ | Stress (error) of a multidimensional scaling configuration |
| $\sigma$ | Normalised $\sigma_{raw}$ |
| $y$ | Out-of-sample object |
| $e_i$ | An entity in $E$ |
| $r_j$ | Entity record that represent $e_i$ |
| **ER** | Entity resolution |
| **MDS** | Multidimensional scaling |
| **LSMDS** | Least-squares multidimensional scaling |
| **MVN** | Multivariate normality |
| **DR** | Dimension reduction |
| **OSE** | Out-of-sample embedding |
| **LV** | Levenshtein distances |
| **k-NNs** | k nearest neighbours |
| **ANN** | Artificial neural network |
| **Name-like vectors** | Vector approximation of names proposed in Chapter 4 |
| **Em-K indexing** | The indexing method proposed in Chapter 5 |
| **ML** | Machine learning |
| **DL** | Deep learning |

# Chapter 2

# Background

In this chapter, we present the background materials that contribute to understanding basic concepts and techniques of ER. We provide an overview and application areas of ER in Section 2.1 and 2.2. Then we explain the challenges in ER tasks in Section 2.4 and different steps in the ER process in Section 2.5. Finally, Section 2.7 elaborates on the large-scale ER problem and a comprehensive overview of the state-of-the-art techniques for scalable ER.

## 2.1 Entity Resolution

Entity resolution (ER) is the process of identifying and linking a group of records from the same real-world entity in multiple databases/datasets [18, 1]. The benefits of linking medical and vital statistical records were acknowledged even before computers became broadly available. Linking records using computers started in the early 50s with Newcombe [7] proposing the automation of ER. Fellegi and Sunter [4] described a framework for probabilistic ER in their seminal paper in 1969, and many practical applications followed this framework.

Medical and statistical organisations started applying ER techniques extensively for both academic and industrial purposes [19, 20]. For instance, advanced probabilistic ER methods have been utilised in injury epidemiology and also in applications of linking medical, police, and traffic data [21].

### 2.1.1 Entity Resolution Definitions

ER has been widely used in academic and statistical research since research data often come from a variety of data sources with different formats. Depending on the field, ER is known by many names. The term record linkage is used in many statistical and health domain [19, 20, 22], whereas the database community refers to the merge/purge

problem or object reconciliation [23]. The term deduplication appears in both database and business contexts [24, 5]. Deduplication often refers to identifying matching entities within a single database or a dataset. ER is also known as record matching, data matching, duplicate detection, or entity resolution in computer science [1, 25]. We use the term *entity resolution* for the rest of this thesis.

Various researchers in different disciplines have adapted one or more above definitions. Early on, Newcombe *et al.* [7] used the term record linkage to indicate the combination of two or more separately recorded pieces that belong to a particular individual or family. Whereas, in recent definitions, ER refer to the task of identifying and matching records that refers to the same real-world entity across several databases or simply the task of identifying different entity profiles that describe the same real-world object [1].

## 2.2 Applications of Entity Resolution

ER solutions are used in many database applications when identifying duplicates, cleansing data, or improving data quality. The following is a list of some example application areas that use ER as a crucial component to improving their services:

- Medical/epidemiological research

  Medical and epidemiological research often combines demographics, hospital admissions, emergency medical services data with genealogies (for hereditary disease), hospital laboratory data. These data are often about patients where patient identifiers are unavailable. Hence, multiple quasi-identifiers (e.g., name, initials, date of birth, and gender) are used to identify records that refer to the same individuals. Combining data on a patient provides a detailed picture of that person's health, providing information for population-level studies.

  The interest lies in the linked data rather than the information (e.g., quasi-identifiers) used to link them in many medical applications. Linked data are crucial to understanding correlations and trends among different factors that affect disease spreading, medical treatments, drug developments, health services, and many more [26, 27]. Some of this information can be used as alternatives to setting up clinical trials or prospective cohort studies [22]. Linked data are also beneficial when conducting longitudinal studies to evaluate treatment outcomes and drug-disease associations [21]. Hence, a significant amount of research explores suitable ER techniques for linking data [22].

- Health services and surveillance

  Health services link records between multiple sources for various purposes. Many health services constantly monitor information about patients, treatments, clinical records, and emergency services to evaluate health care services, improve existing

data infrastructure, and develop health policies [28]. Hence, linking records in different data sources plays a vital role in understanding the relationships and patterns of the data.

Health surveillance is critical for preventing and controlling disease and injury [21]. Such systems require linking several data sources collected on an ongoing basis, such as administrative data, ambulance dispatch data, and clinical data.

Australia became a pioneer in health data matching with the establishment of the health services research linked database program in Western Australia in 1995 [29]. This program has developed and maintained a linkage system that connects data on health events across all individuals in Western Australia, providing data for various projects and research. Later on, New South Wales (NSW) and Australian Capital Territory (ACT) data linkage units and some other countries, including the UK, Canada, United States, have conducted similar programs [30]. The Centre for Health Record Linkage (CHeReL) was established in 2006 in New South Wales and Australian Capital Territory [31]. Similar to Western Australia, CHeReL makes linked administrative and routinely collected health data available to researchers and the government.

- Crime and fraud detection

  ER has applications amongst national security agencies, law enforcement, taxation, and criminal investigations to identify individuals who have committed fraud or other crimes [1, 25]. For instance, law enforcement and intelligence agencies often combine data in criminal justice databases, separate jurisdictions, watch lists, prison and police records, and travel documents. Tax returns, bank records, stock market and real-estate transaction data are usually combined to detect tax and other fraud [32]. ER solutions are applicable to improve the quality of the data and to enhance the performance of data mining and statistical tools used in fraud detection [32, 33].

  Identity deception is a major challenge in such situations where someone intentionally conceals their original identity by impersonating another individual's identity or using forged identity documents [33, 34]. Having multiple identities, either fraudulent or legitimate, can easily mislead intelligence and law enforcement investigators [35, 36]. Identity deception can cause massive financial losses, fatalities and property damages. The 9/11 commission report describes how terrorists fraudulently acquire travel documents using photo-substitute passports and empty baptismal certificates to hide their identities. Hence, early detection of identity deception is important to identify potential terrorists, prevent acts, and protect people from identity theft [37]. Identity resolution is a special type of ER specialised in identity management. Similar to ER, it determines whether a single identity is the same when described differently [36].

Applications of ER techniques are found in eCommerce applications and customer relationship management (CRM). They employ ER for tracking particular customers and managing their interests, conducting marketing activities, deduplicating lists of customers to reduce the cost of advertising, and collaborating with businesses [1].

National statistical agencies collect and link data relating to population, economy, health, education, culture, or politics and provide statistics to governments to improve decision making [38]. Linking census data provides a platform of data for different studies. The US Census Bureau is one of the pioneers in adapting matching techniques and conducting ER research [38, 39]. At present, many such bureaus, e.g., the Australian Bureau of Statistic (ABS), apply ER techniques to improve their services, surveys, and data quality.

ER has also been used to link academic citations of papers to papers themselves and to identify documents on the same topic or written by the same author [40]. A web of knowledge can be created to compile the bibliometric measures of paper influence (where the entities are publications) by using appropriate ER solutions [41]. While such databases allow many users to access research papers, they also provide other services, e.g., impact analysis, notifications of new publications, new citations and similar topic recommendations [1].

ER has been rigorously used to improve data quality, enrich existing data sources, facilitate data mining, and create single information systems known as data warehouses [1, 42]. In many of the integrating projects, ER techniques are significant in standardising and cleaning data to address data quality issues such as inconsistencies, duplicates and errors that could lead to misleading and incorrect information [43, 44]. ER is also critical when creating systems that facilitate information sharing across entity-centric, inter-organisational data stores. Different domains such as health care, law enforcement, public education, and the military collect and store information on a common set of entities. Building a single information system is an efficient way of exchanging information among those organisations [45].

## 2.3 Entity Resolution Software

ER solutions and related products are in demand in commercial settings, often as standalone applications or specialises in particular kinds of applications. These are mainly aimed at deduplication or linking business-related databases that are usually embedded in much larger databases or data warehousing software systems [46]. However, many commercial products on ER or data matching do not publish technical details of their solutions, and only white papers are available.

While there are many open-source data matching systems and various commercial ER software systems available, only a few studies found that has comparatively evaluated these systems. Köpcke *et al.* [47] have provided a survey that evaluates eleven data

matching systems. Campbell *et al.* [48] compared two public domain record linkage software known as *Link plus* and *Link king* and showed that the use of any of these software depends upon the available resources. Ferrante and Boyd [49] proposed a methodology to evaluate ER software packages and systems with a benchmark for linkage quality in different operational environments.

FEBRL (free extensible biomedical record linkage) is one of the most popular ER systems that includes a collection of functions applicable for data cleaning, deduplication and record linkage [50]. The well-known R record linkage package [51] and TAILOR [52], a toolbox for record linkage are described in [1]. Magellan is a recently developed new EM system by Konda *et al.* [53]. While most work on EM focus on developing data matching algorithms, Magellan offers tools and support to the entire EM pipeline (e.g., debugging, sampling), not just the matching and blocking steps. The Magellan repository of ER datasets is also available online[1].

A recent list of freely available software prototypes of ER systems is given in [46]. Some of them are Anonlink [2], AtyImo [54], FEMRL [55], and PRIMAT [56], which are available as open-source software tools. For more details, refer to the references therein.

## 2.4 Challenges of Entity Resolution

The main task of ER solutions is to identify and match all records that refer to the same entity within or across multiple databases. This task is challenging for several reasons, and we have discussed those that motivate our work in Chapter 1.

### 2.4.1 Lack of keys and data quality

The simplest ER case in a data set would be to have keys that uniquely identify the entities available for matching. Some example keys are social security number, driver's license number and tax file number for a person entity. In such a case, identifying records that belong to the same entity can be done using efficient search algorithms with a complexity of $O(N \log N)$ given the dataset size $N$. If the datasets are structured databases, database joins using efficient SQL queries would quickly solve the problem. There are some possible departures from this simplest, ideal scenario,

(a) Keys do not identify an entity uniquely, e.g., two separate entities sharing identical keys
(b) Keys have errors, e.g., a single entity may have different keys.
(c) Both (a) and (b)

[1] `https://sites.google.com/site/anhaidgroup/useful-stuff/data`
[2] `https://clkhash.readthedocs.io/en/stable/`

Therefore, we must be certain about the accuracy, completeness, robustness and consistency of these identifiers [1]. Otherwise, these departures could lead to wrongly matched records.

However, the matching process usually relies upon common attributes across the databases in the absence of unique identifiers, which is the general case in many ER problems. For instance, a database that contains details about people can include quasi-identifiers such as name, address, and date of birth. A key can be created for each record using one or a combination of such identifiers. However, the quality of that information can be low for various reasons such as incorrect, incomplete or outdated details.

Quasi-identifiers such as names are prone to errors, e.g., typographical, spelling, phonetic and optical character recognition errors. These errors can occur due to various reasons. Scanning handwritten texts, entering manually type data and recording through telephone conversations are some situations where the erroneous entries would be recorded. Missing and outdated data are also problematic when matching entity records based on these attributes [1].

Hence, detecting distinct entities across datasets or within datasets is challenging, and it influences the quality and accuracy of the matching results. We will discuss the data quality issues in ER datasets in detail in Subsection 2.5.1.

### 2.4.2 Scalability to large datasets

Due to the lack of unique identifiers and various data quality issues, record matching between multiple data sources requires pairwise comparisons between all records. The number of records to be matched increases quadratically with the size of the datasets. Hence, pairwise comparisons are computationally prohibitive for large-scale data. Comparisons between records require the usage of approximate comparison functions that can handle the data quality issues in records [38]. However, such functions are computationally expensive, where the complexity might depend on the length of the comparing attribute values [1]. Usually, indexing techniques are used to address the scalability challenges in ER, and we will discuss this in Subsection 2.5.2 and Section 2.7 in detail.

### 2.4.3 Privacy and confidentiality

Privacy and confidentially challenges have become a viable concern due to the increasing size of the data in the ER as well as computational and operational challenges [57]. Linked data may have the potential of revealing sensitive or private information since personal information involves in the matching process. Some databases contain highly sensitive information about individuals in fields such as healthcare, finance, and defence [58]. In such contexts, the matching process and the linked data should not reveal sensitive information such as personal data, detailed information about individuals, confidential business

data, or financial details. For an overview, see the book by Christen [46], a dedicated and comprehensive source on linking sensitive data.

The taxonomy provided by Vatsalan *et al.* [25] characterise privacy-preserving ER techniques along different dimensions. The recent survey published by Divanis *et al.* [59] has provided an overview of the large body of research literature in privacy-preserving ER. The authors discussed the existing privacy-preserving ER techniques (their advantages and limitations) by dividing them into four generations with an extensive survey of the latest methods in the last generation. Since privacy preserving ER is beyond the scope of this thesis, we would like to refer the interested readers to [25, 46, 59].

## 2.5 The Entity Resolution Process

ER is often known as a process consisting of multiple steps, where some steps directly address the ER challenges. They can also be a chain of the subtasks or subproblems that make up the ER task. In this section, we discuss the main steps of the ER task, comparing different techniques developed to address the subtasks in each step.

While there is no generally agreed-upon list of definite steps in the ER task, we define a high-level reference model describing the entity resolution process in four distinct steps. Figure 2.1 depicts this reference model of the traditional ER process referencing the processes and figures in [1, 60, 61].

Cleaned and standardised data is critical for improving the quality and accuracy of any data integration task. The first step, data preprocessing, brings different formats and structures of data into the same structure and format. The second step, indexing, reduces the number of comparisons in the matching process. The aim is to avoid comparing non-matching records by grouping similar records. However, if two similar records fall into different groups, they would not be identified as matches in the subsequent steps. The pairs determined by the indexing is known as *candidate pairs*. The third step, compares the candidate pairs in detail using a variety of similarity functions, where the comparisons can be used to determine either exact or approximate matches. In the classification step, the compared records are classified into matches, non-matches, or possible matches. The following sections describe each of these steps in detail.

### 2.5.1 Data Cleaning

Low quality, real-world data collected from different data sources is a challenge in data mining projects, information systems and database applications [43]. A variety of data quality issues such as incompleteness, incompatible formats, errors, and temporal inconsistency are found in real-world data [62]. Many independently managed databases can vary in structure, semantics, and underlying assumptions about the data. Hence, a preprocessing step is needed to prepare the data sources into consistent formats that fit

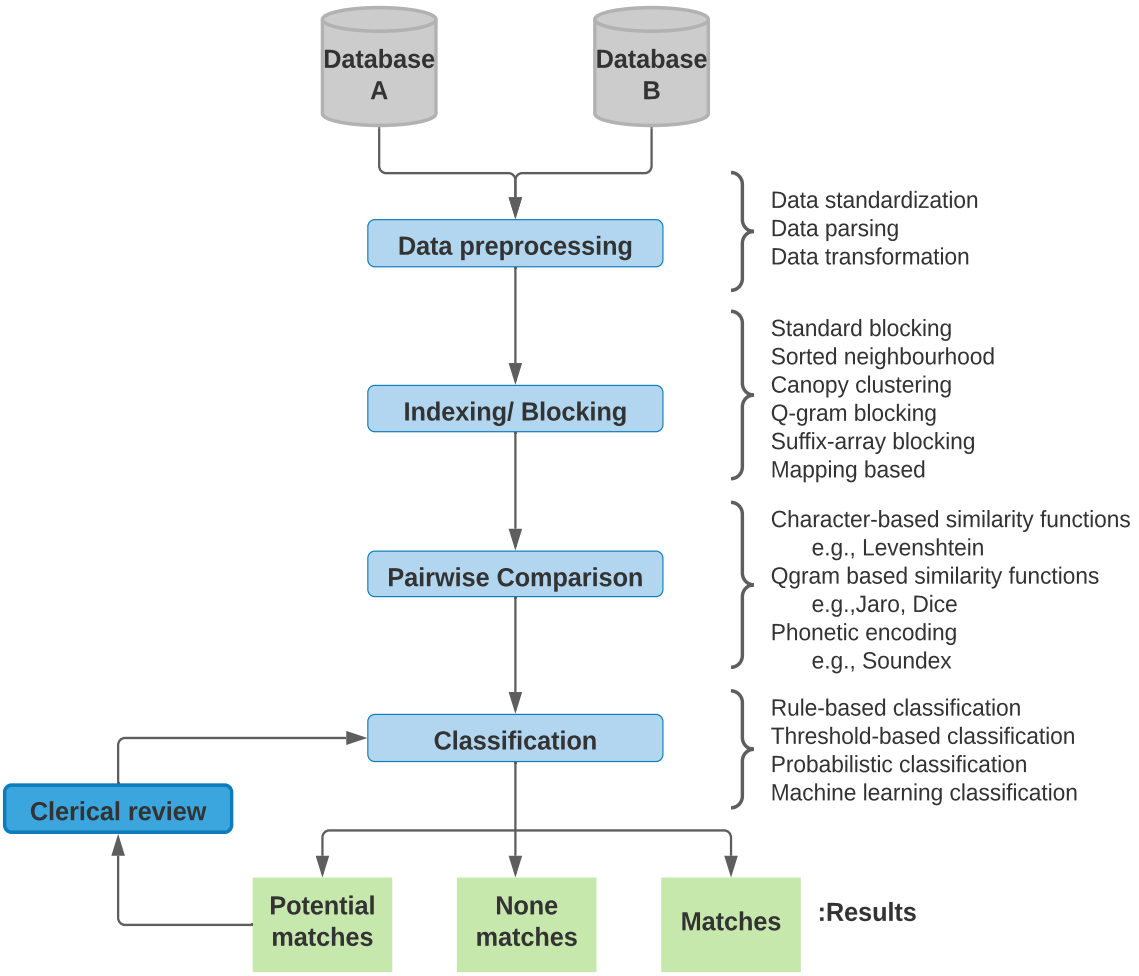



Figure 2.1: Illustration of the reference model for a traditional ER process and its four steps [1, 2]. Different techniques used in each step is summarised next to each primary step. Potential matches require clerical manual review to decide if a pair of records is a match or non-match.

downstream tasks.

Data quality dimensions assess the quality of the data, and popular dimensions in the ER context are accuracy, consistency, and completeness [1, 63]. Accuracy measures the extent that the data are correct, reliable, and free of error [1]. Consistency measures the degree that the several data are consistent and can be combined without a problem [44]. Completeness is the extent to which data are of sufficient depth, breadth, and scope for the task at hand without missing values [43, 63].

The significance of high-quality data and their effect on the final results of the ER solutions is discussed in previous research [5, 64]. Therefore, data preprocessing is necessary to improve the quality of the data by making them comparable and more usable. The following are a few standard data preprocessing techniques commonly used to clean and standardise data in ER.

- Standardisation: The standardisation process converts attribute values stored in different formats in input data sources to a uniform representation [65]. Typical steps involve removing excess punctuation, lower casing all letters, normalising values, removing abbreviates and unwanted characters, correcting misspellings, and using similar formats to store dates and addresses [66]. For instance, removing the abbreviation of all occurrences of the term "St" in an address column to "Street" would make the two addresses "20 Main Street" and "20 Main St." identical strings. The standardisation steps are usually applied by using hard-coded rules or lookup tables [1].

- Parsing: Parsing locates, identifies, and isolates individual data elements coming from the standardisation step to sub-components [5]. For instance, addresses are divisible into three attribute values: Street Name, City, and State. The comparisons of individual components allow easier comparisons between records than the lengthy, complex strings of data. Since parsing facilitates accuracy checks, the values of specific fields can be corrected using dictionaries and lookup tables [43].

- Transformation: The parsed attribute values are then passed to the data transformation step. General transformation steps include the conversion of a data element from one data type to another, renaming of a field, decoding encoded attribute values to original values, range checking, and dependency checking [5]. Range checking examines data fields to ensure they are within the expected range, e.g., a numeric or date range. Dependency checking compares the values of different data fields to make sure they are without conflict. For instance, a city name and a postcode must be consistent, or otherwise, it will become a data entry error that requires correction in one of the attribute values [60, 66].

### 2.5.2 Indexing

A typical matching process between two datasets would require comparing each record in one dataset with all the records in the other dataset. Usually, most comparisons are non-matches. For instance, consider two entity databases $A$ and $B$ with sizes (number of records) $N_A$ and $N_B$ that are to be linked. To match and classify pairs among the two databases in a naive ER approach, the number of comparisons required is the product of the size of the two databases ($N_A \times N_B$). Hence, the computational effort increases

quadratically when the size of the datasets grow and often leads to a performance bottleneck in matching systems [67]. Therefore, ER processes employ indexing techniques to determine which entities are worth comparing by grouping similar records together. It is a crucial part of ER for efficiency and scalability [68, 69].

A recent survey [68] has defined two categories of existing indexing techniques: blocking-based and filtering-based. Blocking based techniques aim to find entity record pairs that are likely to match [70] whereas filtering techniques aim to discard pairs that are guaranteed to be non-matches. Blocking groups likely match pairs into common blocks using a blocking schema based on one or a combination of attribute values (e.g., surname and first name combination). Filtering computes a set of potential candidates performing similarity joins on a given attribute collection using a specific similarity measure and a corresponding threshold [68]. While blocking and filtering shares the same goal, they perform under different settings and assumptions. We cover more details on indexing techniques in Section 2.7 and also in Chapter 5.

### 2.5.3 Comparisons

When the number of candidate pairs is reduced sufficiently, we can compare the records in detail within each group or block. While the indexing step uses one or a combination of a few attribute values, the comparison step utilises several attribute values (e.g., first name, surname, date of birth, and address) [25]. These comparisons are usually performed using a range of similarity functions. The pairwise comparison produces a similarity vector that consists of one or more numerical values that indicate the similarity of two records.

The agreement between two attribute values can be either an *exact* or an *approximate match*. A variety of exact and approximate similarity functions are available to determine whether a given two attribute values are the same or not.

- Exact string comparisons

  The outcome of an exact comparison is usually a binary value of 0 or 1 that indicates whether the attribute values are the same or not. Generally, an exact agreement is represented by a similarity of 1, and disagreement is represented by a similarity of 0. Exact comparisons work well in clean datasets that do not have any typographical or other errors, which is highly unlikely in the real world.

  There are two variations of exact string comparison functions. The first variation only considers the end or beginning of two strings for exact comparisons. These are usually known as truncate comparison functions. The second variation first encodes the two strings using a phonetic encoding function [1]. These functions encode strings with similar sounds (e.g., how a name is pronounced) to replace the original strings with the same code. Phonetic encoding functions are applicable for indexing as well. These are applied to encode blocking or sorting key values to

bring similar-sounding string values into the same block. The entity records with similar-sounding blocking keys will then be compared in detail within the blocks.

- Phonetic encoding: Phonetic encoding techniques provide string indexing by sound using a letter sequence of a string and pronunciation rules to convert (encode) them into index or key [71]. Two strings are considered close to one another if they have a similar sound when encoded. For instance, one of the oldest phonetic algorithms, Soundex [72], calculates the same encode "S530" for such words as "Smith", "Smithe", and "Smyth", as they sound identical. Phonetic encoding is more efficient as a blocking or indexing technique in ER. The variations of the initial Soundex [72] such as Phonex and Phonix [14], and Double-Metaphone [73] and NYSIIS [14, 71] are popular phonetic encoding techniques used in ER solutions.

Exact matching between records is impossible in many real-world ER scenarios due to data quality issues, such as typographical errors and variations. Hence many ER solutions adopt approximate comparison functions. Approximate functions developed for different data types, including strings, numerical, date and time, are discussed in [1, 65, 74]. However, approximate comparison functions are computationally expensive, where the complexity might depend on the length of the comparing attribute values [1]. Since many comparisons in ER are between strings of characters, we discuss some popular techniques for approximate string comparison in more detail.

- Approximate string comparisons

  Approximate comparison measures the similarity or closeness between two attribute values. Hence the similarity is usually expressed as a normalised numerical value between 0 and 1 [25]. A similarity of 1 indicates identical values, whereas 0 indicates two values that are entirely different. Unlike the exact comparison, the similarity between two attributes can also take any value between 0 and 1. Approximate comparison can be performed using a similarity function or a dissimilarity function, where *similarity = constant - dissimilarity*. If a dissimilarity function $d(s,t)$ has the following properties, it is also called a metric or distance function [75, 76]. A function that satisfies the first two conditions is usually considered a dissimilarity function.

  - Non-negativity $d(s,t) \geq 0$.
  - Identity $d(s,t) = 0$ only when $s = t$.
  - Symmetry $d(s,t) = d(t,s)$.
  - Triangle inequality $d(s,u) \leq d(s,t) + d(t,u)$.

However, as pointed out in [75], many dissimilarity functions do not have all these properties. As further described in [1], many dissimilarity functions used in string comparing applications or data matching do not fulfil the requirements of the distance function. For instance, the Jaro-Winkler distance does not hold the properties of symmetry or triangular inequality. Some dissimilarity functions like q-grams do not obey the identity property, while some dissimilarity functions are not necessarily symmetric, e.g., weighted versions of Levenshtein distance [75]. However, there is no strict requirement for dissimilarity measures to be distances in ER context.

A formal definition for approximate string matching is given in [74]. Edit distances formulate the principal approach of approximate string matching [77, 78].

- Edit distances: Edit distance measures the minimum cost of transforming one string ($S_1$) into another string ($S_2$) [63]. Several variations of the edit distance are Levenshtein (LV) distance, Damerau-Levenshtein distance, Bag distance, Smith-Waterman, and Longest Common Subsequence [14, 74, 79]. The popular LV distance measures the minimum number of character insertions, deletions, and replacements necessary to transform $S_1$ into $S_2$. For instance, converting "james" to "jane" requires two edit operations: insert 's' and replace 'm' with 'n', and $LV(S_1, S_2) = 2$.

  Edit distance utilises the strings as a whole without splitting them into segments or tokens and therefore has quadratic complexity. It does not perform well when segments of a string differ or in the presence of abbreviations, e.g., comparison between "james white" and "zara james white". However, some other string dissimilarity measures split strings into segments or tokens before comparing them, and the following overview some of them.
- Q-gram distances: The q-gram distance is derived by listing the q-gram occurrences of the two strings and counting the number of non-shared q-grams between them [75, 80]. The q-grams of a string obtains by sliding a window of $q$ characters over the string. For an example the bi-grams ($q = 2$) for the strings " james" and "jane" are ['ja','am','me','es'] and ['ja','an','ne'] respectively. The two strings only share the common bi-gram 'ja'. Several variations of the q-gram similarity function can be derived using a coefficient method such as the Jaccard coefficient, Dice coefficient, and the Overlap coefficient [14]. For instance, the Jaccard coefficient is obtained by counting the number of common q-grams and dividing it by the total number of unique q-grams in the two strings. The Jaccard distance is obtained by *1-Jaccard coefficient*, and for the above example with bigrams, it is $1 - 1/6 = 0.83$.
- Jaro and Winkler string comparison

  Jaro distance uses the string lengths, number of common characters, and transpositions (swapping of two adjacent characters) to calculate the distance be-

tween two strings [81]. The underlying property is that two common characters in two strings should be equal, and their positions should not differ by more than half of the length of the shorter string.

The Jaro-Winkler method is an improvement to the primary Jaro method, which includes an extra penalty for character mismatches in the first four characters [65]. Winkler has further stated that few errors occur at the beginning of a string, and the error rates by character position increase monotonically when the position moves to the right. Yancey [79] and Christen [14] have discussed and compared several variations of the Jaro-Winkler method.

Developing approximate comparison functions is a major research area in computer science, especially those that deal with typographical errors and variations [25]. Comparisons of such functions can be found in [14, 63, 65, 74, 82]. Besides ER, approximate string matching has numerous applications in bioinformatics, computational biology, text searching, text retrieval, and signal processing [83].

### 2.5.4 Classification

The outcome of the comparison step is a set of similarity vectors generated for each candidate record pair based on all of the compared attributes. These vectors are then used to classify record pairs as *matches, non-matches,* and *potential matches* based on the decision model used for the classification [84]. Potential matches are those pairs where the criteria are insufficient to make a conclusive match or non-match decision [4, 85]. These pairs often refer to a manual clerical review for which human oversight is needed [86]. There are several categories of classification techniques in the ER literature, and we will discuss some popular techniques among them: threshold-based, probabilistic, rule-based, and machine learning-based [25].

- Threshold-based classification: These techniques classify record pairs based on an overall similarity value that sums up all the values in a comparison vector. Assuming $n$ attributes, a single overall similarity of a comparison vector given by $S_v = \sum_{i=1}^{n} s_i$, where $s_i$ is a single value in the comparison vector, for each candidate pair. Then the total similarity $S_v$ is used to classify record pairs based on one or more thresholds. For two threshold case, upper-bound ($u$) and lower-bound ($l$) can classify pairs into matches ($S_v \geq u$), non-matches ($S_v \leq l$), and potential matches ($l < S_v < u$).

  However, this technique does not consider the discriminating power (how informative the attributes are) of the matching attribute and treat them all evenly [87]. For instance, while the maternal name might be more critical than the maternal municipality when identifying a child, threshold-based techniques treat both attributes

similarly. This problem is usually overcome by introducing weights for individual attributes according to their importance or discriminative power.

- Probabilistic classification: Fellegi and Sunter introduced the theoretical foundation of probabilistic classification in 1969 [4]. It evaluates the discriminatory power of each identifier and the likelihood of two records being a true match based on the agreement or disagreement on various identifiers. Each pair of records receives a matching weight where higher weight values indicate a greater likelihood for the underlying pair to be a true match [88]. Each attribute contributes to the match weight depending on its discriminative power. For instance, an agreement of surname would offer more support to a match than a gender agreement.

  Moreover, an attribute with a high frequency in a dataset is less informative than some attribute values that are not frequent. Thus the weights should be adjusted to be smaller for more frequent attribute values [1]. For example, two records with the surnames, "White" should receive less matching weight than two records that share the surname "Roughan". The likelihood of two random records having the surname value "White" is much higher than having the surname "Roughan", assuming "Roughan" is a rare surname compared to "White".

- Rule-based: Rule-based techniques apply a set of rules to classify candidate record pairs into matches and non-matches [89, 63]. We can apply a set of rules to the similarity values of the comparison vectors. Rules consist of individual tests on particular similarity values on the corresponding fields that are then combined with conjunctions (logical AND), disjunctions (logical OR), and negations (logical NOT) [1, 90]. The quality of the rule-based methods highly depends on the skills of domain experts and their ability to define a good set of rules [90]. Therefore, generating rules is a time-consuming process and often demands manual effort [25]. Alternatively, some studies have used machine learning techniques to generate rules by learning them from training data that consist of candidate record pairs and their actual matching status. However, assessing the accuracy of rules is impossible if the actual matching status (match or non-match) of candidate pairs is unavailable. Examples of rule-based classifiers that perform ER effectively by generating rules from the properties of entities are found in [90, 91, 92].

- Machine learning-based: Machine learning (ML) classifications fall into two categories: supervised and unsupervised techniques. Supervised ML approaches require training data with known class labels for matches and non-matches to train a decision model. Then the model can be used to classify the unlabelled pairs of records. Decision trees and support vector machines are two popular supervised learning classifiers applied in the ER process [52, 93, 94]. However, training data is often not available in ER applications and sometimes requires expensive and time-consuming

manual preparation [93]. This issue is usually overcome by using unsupervised learning classifiers since they do not require training data. Clustering is one such technique that groups similar record pairs based on their comparison vectors. The clustering should perform in such a way that each cluster consist of records belonging to one entity [52, 95].

Alternatively, active learning approaches only requires a small amount of training data at the beginning. Then the model is developed interactively by asking an experienced user for further training examples to improve the classification [96]. Modern ER applications apply the concepts of deep learning to develop efficient and effective supervised and active learning classifiers [61, 97].

Other than these four categories, there are collective, group, and graph-based classifiers. The collective classifier uses both attribute and relational information for determining the underlying domain entities [98]. Graph-based classifiers use graph data that represent relationships between entities and their attributes [99]. Group classifiers have been applied to the entities expressed as groups of relational records instead of the individual relational records. These techniques are known for achieving high linkage quality. However, they are not scalable for large-scale data due to quadratic or higher computational complexity [25].

### 2.5.5 Evaluation

Once the ER process is complete, we can evaluate the record pairs classified through the classification to measure the performance of the matching process. It is a difficult task since ground truth data about truly matching records are absent for many datasets. The ER solutions applied in each of the above steps are usually assessed for effectiveness and efficiency.

Classification techniques could produce pairs classified as possible matches that still need to review by a manual clerical process. This is a manual classification technique where each pair is visually assessed to make a manual decision. The manual clerical review of possible matches is a difficult process and impossible with large-scale data. Refer to [1, 100] for more details.

The quality of the matching results determines the effectiveness of the ER techniques. Ground truth data sets with known true-matching and non-matching record pairs are required to assess the quality of the matching data. A classified record pair can belong to any of the four categories below, assuming that the ground truth data with the true-match status is available.

- False Positives (FP): Candidate record pairs classified as matches while they are not actual matches.

- False Negatives (FN): Candidate record pairs classified as non-matches while they actually match.
- True Positives (TP): Correct classification of true matching pairs as matches.
- True Negatives (TN): Correct classification of non-matching pairs as non-matches.

Various quality measures can be defined using the above categories in matching results. Popular quality measures are *precision* and *recall* among the existing quality measures [67, 99] and we define them in the following. Assume $|TP|$ is the number of true positives, $|FP|$ is the number of false positives and $|FN|$ is the number of false negatives.

$$Precision = \frac{|TP|}{|TP| + |FP|}.$$

$$Recall = \frac{|TP|}{|TP| + |FN|}.$$

The efficiency and scalability of the ER techniques are also crucial to measure how the method accommodates large-scale data. The scalability and efficiency issues are usually addressed in the indexing step. The *reduction ratio* and the *pair completeness* are the two main complexity measures used to evaluate the efficiency and the accuracy of the indexing techniques [67] (defined in Chapter 5). A detailed discussion on evaluation measures is provided in each chapter according to their relevance in our work.

## 2.6 ER Tasks

For many of the given applications, we distinguish three cases of ER depending on the input and its characteristics [6].

1. *Dirty ER:* takes as input a single set of records that contains duplicates and produces a set of equivalence clusters of distinct entities. It is also known as deduplication in many database applications.
2. *Clean–Clean ER:* takes two duplicate-free but overlapping sets of records. The goal is to identify matching records that belong to a single entity.
3. *Multi-source ER:* takes more than two sets of overlapping records as inputs. Apply deduplication to the union of all collections or use a sequence of pairwise ER tasks given the individual data sets are duplicate free.

Christophides *et al.* [6] has presented a taxonomy of various ER settings and approaches based on their key characteristics. Most existing ER techniques are solutions for offline batch processing of static databases.

However, many organizations manage streaming data. Streaming data is commonly related to dynamic data sources (e.g., from web systems, social media or sensors), whose data processing is continuous [101]. Large-scale data may arrive in high-velocity streams or as queries against a known entity collection [6]. Such application requires online ER approaches that provide prompt responses. Therefore, static, offline ER solutions are not adequate to process as many entities as needed to resolve queries in real-time. We discuss real-time ER requirements in more detail below.

Real-time ER is a query-based data matching process where a stream of query records should resolve against records in one or more existing data set(s) in a sub-second time [102]. It is also called the query matching problem in ER. Relatively few ER studies consider real-time query matching [103, 104, 12].

Similarly, some applications require adding new entities incrementally to existing clean and evolving data repositories, such as a data warehouse [6]. This is known as incremental ER that adds entities without repeating the entire ER process to the previously matched data [11]. Incremental ER is usually combined with streaming data to process (i.e., match) entities as they appear. Some ER approaches that deal with streaming data, e.g., Kun *et al.* [105] do not consider incremental processing and discard queries after answering them [106].

The steps of ER discussed in Section 2.5 apply to static databases and static ER techniques. Depending on the application and the task of ER, these steps may need to be adjusted. Since indexing is one of the crucial steps that affect the efficiency of any ER technique, fast and approximate indexing techniques are essential for real-time ER. Real-time or query matching ER applications is one instance that shows the importance of large-scale ER. Next, we provide an overview of large-scale ER and existing solutions that address scaling issues and big data problems in data matching.

## 2.7 Large-scale Entity Resolution

Data integration becomes complex with big data characteristics, e.g., velocity, variety and volume [6]. Hence, the demand for scalable ER techniques has increased tremendously to facilitate the integration of large-scale data collections. In recent years, the rise of the Web and cloud-based applications has led to several extensions of techniques and algorithms for ER [77]. The data for these applications come from different domains with a semi-structured format that exists on a very large scale. These properties of data bring new challenges to ER and consequently require new techniques or algorithms as solutions.

Large-scale data are usually handled by applying indexing techniques to reduce the search space of ER. Even though indexing alleviates the problem, sequential processing

of an ER task may not be feasible with massive volumes of data. Many of the existing indexing solutions are designed for ER between two data sources. Even though pairwise matching is the foundation of many ER solutions, such binary mapping approaches do not scale to many data sources [107]. However, ER is an ideal problem to be solved in parallel on cloud infrastructures since it is a costly process [108]. Big data applications that employ parallel ER approaches to address the performance and scalability requirements are found in [109, 110].

### 2.7.1 Indexing for Large-scale ER

Indexing is a crucial step that scales ER solutions. However, traditional indexing techniques are usually inadequate to support large-scale ER. The underlying principle of these techniques is to group similar-seemingly records into blocks and use a pairwise comparator to explore the blocks exhaustively [111]. They often operate by defining a set of key fields from the data to determine which records will be in which blocks. Many of these methods are schema-aware indexing methods designed for structured data (databases), assuming that the input entity profiles adhere to a known schema. Hence these techniques are not applicable for heterogeneous and loosely structured data such as Web Data [68].

The main blocking techniques that operate on structured data are traditional blocking [4], sorted neighbourhood [23], q-gram [69], suffix array [112], StringMap [16], canopy clustering [113] and MFIblocks [114]. A survey presented by Christen [69] has discussed six primary indexing techniques that provide information on their scalability to large datasets and the performance for data with different characteristics. The experimental results have shown that the q-gram based methods are the slowest, while traditional blocking and the array-based sorted neighbourhood approaches are the fastest overall. Many of these methods depend on the selection of a suitable blocking key. ML based solutions are one approach to select the best blocking key [115].

ER has gained significant importance with the increasing heterogeneity in data. Developing indexing techniques for semi-structured, heterogeneous data (e.g., Web data collections) holds new challenges compared to databases. These are known as schema-agnostics indexing techniques that extract blocks from the attribute values without any assumption on schema knowledge [68]. The primary technique is token blocking which assumes that duplicates share at least one common token (attribute values) [116]. The method uses a transformation function to extract all tokens from all attribute values of every entity to create blocks for distinct tokens. For more details on schema-aware and schema-agnostics blocking methods, refer to [68].

### 2.7.2 Other Techniques for Large-scale ER

Parallel ER methods exploit the processing power of multiple cores to minimize the response time of ER to improve scalability. It is a technique that complements indexing

techniques to reduce the time needed to match records among databases. Many of the existing approaches use the MapReduce framework, which supports the parallel matching of entities [108, 110]. ER process contains separate, independent operations such as indexing, comparisons, etc., making it a perfect candidate for intra-step parallelism. A recent survey [77] provides an overview of several modern parallel ER approaches.

The MapReduce framework is often used to scale indexing methods to massive entity collections. Both schema-aware and schema-agnostic indexing methods can be adapted to MapReduce using one or more jobs for parallelising [6, 68]. Meta-blocking exploits the MapReduce framework to develop scalable algorithms for ER based on Web Data [117]. Similar efforts are applied using the tool Dedoop to structured data. Dedoop supports multi-pass blocking techniques such as the standard blocking and sorted neighbourhood, grouping entities according to multiple blocking keys [118]. Refer to the recent survey [68] for existing work in indexing and filtering techniques that use parallelisation techniques.

Data partitioning and load balancing are crucial aspects in parallel ER that significantly influence parallel programs [77]. Achieving a good load balance is difficult for applications when the distributions of data to processors depend on actual values that are skewed or irregular [119], which is often the case for ER applications [1]. The processors with a heavier load will dominate the runtime and decrease the response time if the workload is not assigned evenly. Hence some processors have to wait for other processors to start the next computation phase [77].

Holistic ER is a technique that matches entities from many sources instead of pairwise matching between one or two sources. It is a clustering-based technique that holistically determines clusters such that all matching entities from any source are combined to a single cluster [107]. These clusters facilitate the integration of data without individual links to each other data source [120]. Clustering the entities across all sources can be performed with much less effort than determining the quadratic number of pairwise matching [107]. For more details on holistic ER, refer to [109, 110].

## 2.8 Summary

In this chapter, we presented an overview of the ER process, its definitions, the importance of ER in various applications, and challenges. We also described the steps of the ER process, different ER forms and a brief overview of large-scale ER and current solutions. As the literature suggested, practical applications of ER have trade-offs in matching accuracy vs qualities and matching quality vs scalability.

A plethora of articles have been published on ER solutions over a decade of research. However, large-scale ER is still an emerging research area with the growing demand in integrating large data volumes in many data mining and analytic projects. While there are several ongoing research activities in large-scale ER, several factors make it difficult, e.g., poor data quality, lack of training data, scalability, real-time data processing. Therefore

matching speed, scalability to very large databases, and real-time matching capabilities of ER techniques that provide high-quality results are high in demand. While there is generous room for improvement in this area, we mainly focus on the following challenges in our thesis.

- Efficient test data generation: The lack of large test data collections and publicly available benchmark data sets is a primary concern when developing new algorithms and comparing existing ER algorithms and methods. While researchers use their own data sets for testing, only a few small test data sets are open-source. Hence simulation of large-scale data for testing ER algorithm remains open to research.
- Scalability and speed of computation: With the growth of large-scale data, ER solutions demand computational efforts and larger storage and memory resources. With many databases moving to online and real-time record matching systems from static databases, the existing indexing techniques become less efficient or not applicable. Hence, novel efficient indexing techniques that facilitate fast and scalable ER remains a challenging problem in practice.

# Chapter 3

# Mathematical Foundation

In this chapter, we present the main concepts and notions of entity resolution (ER) relevant to our work and the tools and techniques we used. This chapter also provides the mathematical foundation required for the solutions described in this thesis. Further, the relevant definitions of the individual tasks described in Chapter 4, 5, and 6 are provided. First, we summarise the terminology and the notation used in this chapter in Table 3.1.

Table 3.1: Summary of the main notations used in this chapter.

| | |
|---|---|
| $R$ | Set of objects in a metric or non-metric space |
| $n$ | Number of objects in set $R$ |
| $\delta$ | Dissimilarity or distance between two $R$ objects |
| $X$ | Coordinates of $R$ objects in configuration space |
| $d$ | The Euclidean distance between two $X$ coordinates |
| $\phi$ | Mapping from a one space to another |
| $\Delta$ | Dissimilarity matrix of $R$ objects |
| $D$ | Distance matrix in Euclidean space of $X$ points |
| $K$ | Dimension of a multidimensional scaling configuration |
| $\sigma_{raw}$ | Stress (error) of a multidimensional scaling configuration |
| $\sigma$ | Normalised $\sigma_{raw}$ |
| $y$ | Out-of-sample object |
| $E$ | Set of entities (entity collection) |
| $e_i$ | The $i^{th}$ entity in $E$ |
| $r_j$ | Entity record that represent $e_i$ |

## 3.1 Entity Resolution

An *entity* can be a person, place, product, organisation or any object with a unique identity that distinguishes it from all other entities of the same type. A collection of name-value pairs that describe a particular entity is known as an *entity profile*. A pair of similar entity profiles are called *duplicates*. A database representation of an entity profile is usually referred to as a *record*.

Let the letters $T, V$, and $I$ denote the infinite sets of *names, values* and *identifiers* respectively. The definitions of relevant concepts here follow those in [6, 68].

**Definition 1** *An entity profile (or simply, a record), denoted by $r_j = \langle id, A_{id} \rangle$, is a tuple consisting of a unique identifier $id \in I$, and a list of attributes $A_{id} = [(t_1, v_1), \cdots, (t_x, v_x)]$ of name-value pairs that describes a real-world object, where each attribute $a_k \in A_{id}$ is a tuple $\langle t_k, v_k \rangle$ given $t_k \in T; v_k \in V$ , $\forall k \in [1, x]$. A set of entity profiles is called entity collection, denoted by $E$.*

**Definition 2** *(Entity Resolution): Given two records $r_i, r_j$ is a match, if they refer to the same unique real-world object. We denote this as $r_i \equiv r_j$. The goal of ER is to link different records that describe the same entity within an entity collection or across entity collections.*

Following the above definitions, we distinguish between the following two tasks of ER. The definitions follow those in [6, 68]. Refer to Chapter 2, Section 2.6 for other forms of ER tasks.

**Problem statement 1** *(Dirty ER): Given an entity collection $E$ that contains entities $e_1, e_2, ..., e_n$, find all the records $r_j$ that represent each entity $e_i \in E$.*

**Problem statement 2** *(Clean-Clean ER): Given two duplicate free entity collection $E_1$ and $E_2$ that contains entities $e_1, e_2, ..., e_n$, find all the records $r_j$ that represent each entity $e_i \in E_1, E_2$.*

Matching records are determined by pairwise comparisons between records in entity collections. Hence, a similarity score is calculated by applying one or more similarity functions over the corresponding attributes of pair of records. We have discussed various similarity functions in Chapter 2, Subsection 2.5.3.

**Definition 3** *(Record comparison): Given a pair of records $(r_i, r_j)$ and a set of attributes $A_{id}$ that describe them, similarities $s_1, ..., s_x$ between attribute values are determined by applying a set of similarity functions $sim_k(r_i.v_k, r_j.v_k)$, with $1 \leq k \leq x$, to measure similarity of values in the attribute $a_k$. Here, $r_i.v_k$ and $r_j.v_k$ are the $k^{th}$ attribute values of the records $r_i$ and $r_j$. The total similarity between the records is given by the similarity score $S_v = \sum_{i=1}^{x} s_i$.*

Definition 3 provides a simplified definition for record comparison. However, the weighted summation of similarities is the common approach in ER. Hence, different attribute values will be given different weights according to their importance or discriminative power. The weighted sum is calculated by multiplying the similarity value and the given weighted value per attribute before summing it up to the final similarity score [1, 88].

A brute-force approach to finding all duplicates in $E$ requires a pairwise comparison of all entity records in $E$. In practice, this is infeasible when the number, $|E|$, of entity records in $E$ is large due to the inherent quadratic complexity of the comparison process.

To support large-scale ER, there is a need for a crucial first step known as *indexing* (or *blocking*). The idea is to group entity records in $E$ into blocks such that the comparison of records across different blocks is not necessary. Hence, reducing the number of pairwise comparisons.

Let $B_s$ denote the set of blocks generated by a indexing algorithm for a given $E$, and $b_i \in B_s$ be a block in $B_s$ for $i \in [1, m]$, where $m$ is the number of blocks in $B_s$. Let $|B_s|$ be the total number of record comparisons in $B_s$: $|B_s| = \sum_{i=1}^{m} |b_i|$. We formally define the indexing problem as follows.

**Problem statement 3** *(Indexing): Given an entity collection $E$, generate a set $B_s$ of entity records groupings such that similar records are grouped within the same block and the $|B_s|$ is minimal.*

## 3.2 The Embedding Problem

This section presents a brief introduction to the embedding problem and the MDS algorithms.

Embedding is a technique for capturing the critical structure of a high-dimensional space in a low dimensional space. It has applications in many fields, including mathematics and various branches of computer science. The embedding we use in this thesis is that of inferring coordinates from distances. Significant speed-ups can be obtained by embedding objects into another space that allows efficient distance functions such as the $L^p$ norm for applications with a computationally expensive distance function [121].

Many standard embedding methods map a metric space to a coordinate space, i.e., vector space. However, embedding a non-metric space into a coordinate space is also possible with modern dimensional reduction (DR) techniques. This thesis focuses on both metric and non-metric spaces since not all dissimilarity functions used in ER data comparisons are necessarily distance functions. Hence, the embedding can be started with dissimilarities (non-metric space) or distances (metric space). We define the embedding problem as follows.

Embedding of a metric or non-metric space $(R, \delta)$ into a coordinate space $(X, d)$ is a mapping $\phi : R \to X$. In this work, $(R, \delta)$ will always be a finite space (i.e., $R$ is a finite

set) and $(X, d)$ will always be a Euclidean space. This embedding process usually distorts the original distances between objects when embedding objects in a metric space. This distortion is usually measured using a loss function known as stress. More details are given in Subsection 3.2.1.

**Problem statement 4** *(The embedding problem): For a given metric or non-metric space, find a $\phi$ that minimises the distance distortion, stress, or a similar error metric between $(R, \delta)$ and $(X, d)$.*

The most commonly used technique for embedding a set of distances (or dissimilarities) into a coordinate space is multidimensional scaling (MDS) [122]. We use MDS as an embedding technique rather than a DR technique since our input data are mostly distances or dissimilarities.

We focus on MDS in this work because MDS can be used to analyse data defined only by a dissimilarity matrix. This does not even have to be strictly a distance metric, nor does the original space need to be Euclidean. Hence, MDS applies to many data and has applications across many disciplines, including clinical psychology, sociology, marketing, ecology, and biology [123].

### 3.2.1 Multidimensional Scaling

It appears that the use of MDS in ER applications or research has not been addressed before. However, metric space-based ER applications are found in the literature [15, 16]. We will discuss these in Chapter 5. One of the main reasons MDS has not been considered for ER research is that the algorithm is quadratic, hence prohibitive for large-scale data. Since ER has quadratic complexity, it is hard to obtain any speed up in the computations or scaled solutions for large-scale data by applying MDS. In this thesis, we show how MDS algorithms can be modified to provide efficient solutions to various ER problems, including scaling. We will discuss those in future chapters. In this section, we define MDS and discuss its several variations and applications.

#### Overview

MDS initially evolved as a model for certain psychological phenomena and, later, gained popularity as a general-purpose data-analytic tool [124, 125]. It has been known as the problem of finding a configuration whose distance fits best with the given dissimilarities [126], where *configuration* is $N$-points in a $K$-dimensional Euclidean space.

Typically, MDS takes proximity data or high-dimensional data and reduce the data into a more interpretable form, often though not always in two or three dimensions [127]. For instance, the outcome of MDS might be a map in two dimensions that conveys spatial relationships among input objects. This map needs to preserve the original proximities

such that similar items are close and dissimilar items are further apart [124]. Hence, the general MDS algorithm can be thought of as a non-linear optimisation problem for mapping from the original proximity information to the target space.

An MDS algorithm takes the input of $N$ points as a $N \times N$ matrix $\Delta$, a distance or affinity matrix. An entry in the $i^{th}$ row and $j^{th}$ column of $\Delta$ is the dissimilarity $\delta_{ij}$, between object $i$ and object $j$, where $\delta_{ii} = 0$, and $\delta_{ij} > 0$, $i \neq j$. Then it attempts to construct a configuration of $N$ data points $x_1, ..., x_N$ in $K$ dimensions, such that if $d_{ij}$ denotes the Euclidean distance between $x_i$ and $x_j$ , then $D$ is similar to $\Delta$ [128].

MDS maps high dimensional data (the original space) into low-dimensional metric space (the configuration space). The rationale for performing such a mapping is that distances in a configuration space approximate the distances between objects in the original space, where searching the configuration space become less expensive.

The configuration space is a vector space where the elements consist of real-valued coordinates (vectors). It is a special and a more restricted version of a metric space. The vector space holds additional properties that can be exploited in index structure designs [121]. A vector can be uniquely located in this space, and distance computations become efficient. Minkowski metrics based on $L^p$ norms, $\| x \|_p = (\sum |x_i|^p)^{1/p}$, with $p \geqslant 1$ are popular in vector spaces. We used the most common Minkowski metric Euclidean distance $d_E(p = 2)$ in our calculations. Moreover, due to the availability of operations such as vector addition and subtraction, new vectors can be constructed from previous vectors equally easily [121].

Different variations of MDS has been proposed in the literature [123, 125]. For an overview of MDS, see [17, 123]. Next, we discuss the two metric-MDS approaches used in this work.

### Classical Multidimensional Scaling

Classical multidimensional scaling (CMDS) is the simplest form of MDS, which uses an algebraic method to reconstruct a lower-dimensional map [129]. It is also known as a spectral method since the configuration is obtained using a set of eigenvalues and eigenvectors. CMDS is very similar to principal coordinates analysis (PCA), but it uses a distance matrix rather than a correlation matrix as the input [76]. For more details on CMDS refer to [76, 129].

For a given input dissimilarity matrix $\Delta$, the coordinate matrix $X$ in $K$ dimensions is obtained by applying the following steps of the algorithm Algorithm 1.

CMDS requires the computation of a full input matrix ($\Delta$) which is impossible with large-scale data. Several methods have been discussed in the literature to approximate the matrix $\Delta$, using only a subset of its data to reduce the space and time complexities [130].

The input data to CMDS can be any measurement of dissimilarities between objects. Interpreting the results of the CMDS algorithm is much easier if the input data are Euclidean distances. Otherwise, careful analysis of the output eigenvalues of the algorithm

**Algorithm 1** Classical multidimensional scaling algorithm

**Input:** Dataset $N$, $\Delta$

1: Compute the squared dissimilarity matrix $\Delta^2$.

2: Compute $J$ by applying double centring to $\Delta^2$: $J = -\frac{1}{2}H\Delta^2 H$.
$\#\ H = I - \frac{1}{N}ee^T$ and $e$ is a column vector of all 1's.

3: Calculate the eigendecomposition of $J = V \wedge^{1/2} V^T$.

4: Calculate the $K$ dimensional coordinate matrix $X$ by $X = \wedge_+^{1/2} V_+^T$.
$\#\ V_+$ is the first $K$ eigenvectors of $V$ and $\wedge_+$ is the first $K$ eigenvalues of $\wedge$.

5: return $X$.

is required when deciding the dimension of the output data. However, most of our input data is not Euclidean. For instance, string dissimilarities between names are not always distances, i.e., metrics. We will discuss the non-Euclidean nature of our data, e.g., name string dissimilarities, in detail in Chapter 4. However, least-squares multidimensional scaling (LSMDS) is appeared to be robust against non-Euclidean data compared to CMDS in our experiments, and we further elaborate on this in the next section.

### Least-squares Multidimensional Scaling

Another method for MDS is least-squares multidimensional scaling (LSMDS) [129], which minimises the raw stress of a configuration. LSMDS allows the introduction of weights and missing values, unlike the eigendecomposition approach in CMDS.

We observed that CMDS results were less accurate in approximating non-Euclidean or non-metric data during our experiments. Since many ER applications use non-metric dissimilarity measures, we will be using the LSMDS algorithm throughout the thesis.

The input to LSMDS is a dissimilarity matrix $\Delta = [\delta_{ij}]$, where $\delta_{ij}$ is the distance between the indices of two data points $i$ and $j$ in a high-dimensional space. For a given dissimilarity matrix $\Delta$, a configuration matrix $\mathbf{X} = [x_1, x_2, .., x_N]$, where $x_i \in \mathbb{R}^K$ is constructed by minimising the *raw stress*,

$$\sigma_{raw}(\mathbf{X}) = \sum_{i,j=1}^{n} \Big(d_{ij}(\mathbf{X}) - \delta_{ij}\Big)^2. \tag{3.1}$$

The Euclidean distance between the $i^{th}$ and $j^{th}$ points in the configuration space is given by $d_{ij}$. One can use squared distances or a normalised raw stress criterion as well. We prefer the normalised stress ($\sigma$) in our experiments since it is popular and theoretically justified [131].

The *normalised stress* $\sigma$ is obtained by

$$\sigma = \sqrt{\sigma_{raw}(\mathbf{X})/\delta_{ij}^2}. \tag{3.2}$$

The configuration matrix $X$ in $K$ dimensions is obtained for approximating $\Delta$ by applying the following steps of the algorithm Algorithm 2.

We implemented LSMDS by applying iterative gradient descent on the stress function ( See Algorithm 2). Some MDS applications have used other implementations for LSMDS based on the SMACOF (stress majorization of a complicated function) algorithm. For MDS, majorization was introduced by De Leeuw [132].

We compared the stochastic gradient-descent (SGD) algorithm and the SMACOF algorithm in terms of stress and computational time. More details are given in Chapter 4, Subsection 4.3.1. We found SMACOF is comparatively slow for our data. Hence, we use SGD-based LSMDS for the rest of the experiments.

**Algorithm 2** Least-squares multidimensional scaling algorithm.

**Input:** Dataset $N$, $\Delta$
1: Randomly initialise $Y^{[0]}$
2: $X \leftarrow Y^{[0]}$
3: $t \leftarrow 0$
4: $\epsilon \leftarrow$ small positive number
5: $MAX\_IT \leftarrow$ maximum iteration
6: Compute $\sigma(Y^{[0]})$ using Equation 3.2
7: Set $\sigma^{[0]} = \sigma(Y^{[0]})$
8: **while** $t \leq MAX\_IT$ **do**
9: $t = t + 1$
10: Update: $Y^{[t]} = Y^{[t-1]} - \eta\nabla\sigma^{[t-1]}$
# $\eta$ is the learning rate and $\nabla\sigma^{[t-1]}$ is the gradient of $\sigma$ in iteration $t-1$.
11: Compute $\sigma^{[t]} = \sigma(Y^{[t]})$
12: $X \leftarrow Y^{[t]}$
13: **if** $\sigma^{[t-1]} - \sigma^{[t]} < \epsilon$ **then**
14: break;
15: **end if**
16: **end while**
17: return $X$.

In general, MDS algorithms require extensive preprocessing and usually are computationally expensive, thus not appropriate for large scale applications. The two main drawbacks are,

- MDS requires $O(N^2)$ time, where $N$ is the number of items. Thus, it is impractical for large datasets.

- In an out-of-sample setting or a query-by-example setting, a new item has to be mapped to a point in the existing configuration. Given the MDS algorithm is $O(N^2)$, an incremental algorithm to search/add a new item in the database would be at least $O(N)$. Hence a new item search (or add) in an existing configuration would be similar to sequential scanning of a database.

### 3.2.2 Out-of-sample Embedding

Once a dataset has been transformed, it is common to need to incorporate new data. This is known as the out-of-sample embedding (OSE) problem.

Many DR or embedding techniques such as MDS do not offer an explicit OSE function. Hence, mapping new data into an existing configuration space without extensively reconstructing the entire configuration from the augment dataset is difficult. Running the MDS algorithm from the beginning using all the data (old data and new data) is not a suitable approach for large-scale multidimensional datasets.

In the context of metric-MDS, we found several out-of-sample extensions. The classic reference in the field is Trosset and Priebe [133] for CMDS. Bae *et al.* [131] proposed a similar approach that uses general MDS results to find embeddings approximating to the distances. The proposed method is an interpolation approach to the LSMDS that extends the SMACOF implementation. Following similar ideas of [131] and [133], we proposed an OSE method for LSMDS. We use the general LSMDS results rather than the inner products of the dissimilarity matrix for the least square approximation as in [131].

The method uses the SGD algorithm for numerical optimisation to minimise the following objective function. However, note that this optimisation is subtly different from that of the original LSMDS (3.1). Hence the OSE of a new object $y$ is obtained by minimising,

$$\hat{\sigma}(y) = \sum_{i=1}^{n} \big( \|x_i - \hat{y}\|_2 - \delta_{iy} \big)^2, \tag{3.3}$$

where $y$ is the new object and $\hat{y}$ is its position in the configuration space. The $\delta_{iy}$ represent the dissimilarities between point $i$ and the new object $y$. The Euclidean distance between the $i^{th}$ point and $y$ in the configuration space is given by $\|x_i - \hat{y}\|_2$. We seek to find a position of $y$ that minimises $\hat{\sigma}(y)$.

We will discuss the application of OSE methods in ER applications in detail in Chapter 5 and Chapter 6. We also used an artificial neural network (ANN) to solve the OSE problem as a regression problem. Our OSE solutions based on the optimisation approach and ANN are described and compared in detail in Chapter 6.

### Landmarks

One way to improve the speed of MDS mapping is to consider only a fraction of the input data initially. Hence, the first step is to apply MDS only on a subset of all input points. In the second step, the remaining data can be added to the existing configuration as out-of-sample points. The initially selected subset of input data is usually known as *landmarks*. Landmarks or *anchors* have been used with OSE extensions to scale MDS and other embedding techniques [134, 135].

Several landmark-based MDS algorithms extend CMDS. Silva *et al.* [134] proposed the Landmark MDS (L-MDS) algorithm to address the inefficiencies that occur when processing large data sets. The first step of L-MDS is to apply CMDS only to the landmark points, embedding them in a lower-dimensional space. Then each remaining point is embedded independently from each other using a fixed linear transformation given the distances to the landmarks [130].

Although L-MDS improves scalability, it is sensitive to noise or outliers than the original CMDS. In contrast, the Landmark MDS ensemble method [136] combines individual L-MDS solutions that operate on different input partitions. It has shown improved accuracy with similar scaling abilities to L-MDS for large datasets. Similar ideas of landmarks are discussed by Bae *et al.* [131] to extend the LSMDS using a subset of pre-configured MDS results to develop an interpolation technique.

There are several other scalable MDS algorithms worth mentioning here. The FastMap algorithm is an MDS approach that determines one coordinate at a time by examining a constant number of rows of the input distance matrix of the existing data points [137]. MetricMap [138] has developed as an improvement to FastMap. The algorithm first calculates the pairwise distances among a selected set of objects and uses these distances to establish the target space. Then it maps all remaining objects in the dataset to points in the existing target space [138]. A comparison of L-MDS, FastMap and MetricMap is given in [139].

The selection of the best landmarks remains an open research problem. Landmark selection may depend on the MDS algorithm, nature of the input data and application area. Some of the landmarks selection methods found in the literature are as follows:

- Random selection : Randomly selects landmark points from the input dataset. This works well in practice [130], especially with large-scale data, since the computations are quick and cheap.

- Farthest point sampling (FPS) is a standard method of selecting sources starting with an initial random point. Then it iteratively picks the farthest point from the already selected sources [140]. However, FPS accesses the entire distance matrix between all the available data points rather than a selected set of rows. Therefore, it is more computationally expensive than random selection.

- Hierarchical clustering: We consider the agglomerative method for landmark selection among the two main types of hierarchical clustering. It considers each point as a single-element cluster at the beginning. Then the clusters are combined to create new clusters (based on the distances from one to another) until all points are included in a final single cluster. The results create a dendrogram, and the cut to the dendrogram controls the number of clusters obtained. Then randomly chosen points are selected from each of those clusters as landmarks. Hence the number of landmarks depends on the number of chosen clusters. However, this method is computationally expensive due to the extensive number of distance calculations.

As noted, the choice of landmark points may seem like a clustering problem. However, clustering algorithms such as hierarchical clustering require subsequent calculations of distances between data samples and subclusters in each computationally expensive step. Hence we have used hierarchical clustering only for comparison purposes in our experiments. We recommend using random selection for quick approximations. However, FPS has the advantage of being controllable when reproducible results are desired.

We combine the ideas of landmarks and OSE to address the drawbacks in MDS (discussed in Subsection 3.2.1). Our applications use these techniques to propose an indexing technique for fast, real-time and approximate query matching in Chapter 5.

## 3.3 Similarity Search

Similarity searching refers to searching for database elements similar to a given query element [121, 141, 142, 143]. It is a fundamental task in computer science that appears in various fields such as information retrieval, data mining, pattern recognition, computational geometry, and databases [121, 142]. Exact searching is not as useful in many applications due to data quality issues. Proximity concepts (similarity, dissimilarity) are typically much more productive for searching [121]. We use similar ideas to 'similarity search' in vector spaces to develop ER solutions appropriate for large-scale entity matching. Here, we present the methods and techniques of similarity searching that are utilised to design our algorithms.

**Problem statement 5** ***(Similarity searching)****: Given a finite database $S \subseteq U$, which is a subset of the universe $U$ of objects that can be preprocessed, finds all similar elements in $S$ for a given new object from the universe (a query $q$), where $q \in U$.*

There are two general types of similarity queries [141, 144]; both using a distance function $d$ defined on $U$.

- **k-nearest neighbour (k-NN) queries**: The query retrieves the $k$ closest elements to $q$ in $S$. If the collection to be searched consists of fewer than $k$ objects, the query

returns the whole database. If the set of the $k$ nearest neighbours of $q$ are $S_q$, then formally, $S_q$ can be defined as follows:

$kNN(q) = S_q \subseteq S, |S_q| = k \wedge \forall x \in S_q, y \in S - S_q : d(q, x) \leq d(q, y).$

- **Range queries**: The query retrieves all objects within distance $r$ to $q$. When the search radius $r$, is zero, the range query $R(q, 0)$ is called an exact match. Formally, the query response is defined as :

  $R(q, r) = x \in S, d(x, q) \leq r.$

### 3.3.1 Nearest-Neighbour Search

Nearest neighbour retrieval is an essential operation in a wide variety of real systems such as classification [145] and analysis of biological sequences [146]. It is also a well-established technique in pattern recognition, memory-based reasoning, and vector quantisation [121, 147]. Efficient and scalable indexing methods that can facilitate accurate and efficient nearest neighbour (NN) search are required for large-scale databases [143].

An index is a data structure that reduces the number of distance evaluations needed at query time. An efficient indexing method would iteratively prune subsets of potential answers to a query by partitioning the data set [148]. We can apply several k-NN search methods for indexing arbitrary metric spaces; for more details, refer to the surveys [141, 142]. Distance-based indexing methods use distance computations to build the index. Once the index is created, these can often perform similarity queries with a significantly lower number of distance computations than a sequential scan of the entire dataset [141].

Tree-based indexing structures for metric spaces include R-tree and its variants [149], Kd-trees [150], VP-trees [147], and M-trees [121, 151]. The decision of which indexing structure to apply depends on several factors, including query type, data type, complexity and the application.

In addition to the popular tree-based indexing methods, one can also use a hashing based algorithm such as locality sensitive hashing (LSH) to identify approximate nearest neighbours. LSH maps high-dimensional data into lower dimensions using random hash functions. The idea is to map similar data points in the original space into the same hash bucket in the lower dimensional space [60, 152]. The applications of LSH as an indexing algorithm in ER is found in [153, 154]. However, applying LSH for nearest neighbour search is orthogonal to the methods proposed in this thesis since we apply MDS for embedding high dimensional data. Hence, we focus on tree-based indexing structures for k-NN search.

Among the tree-based indexing, we choose Kd-trees for the k-NN search. It is considered one of the best data structures for indexing multidimensional spaces and is designed for efficient k-NN search [15].

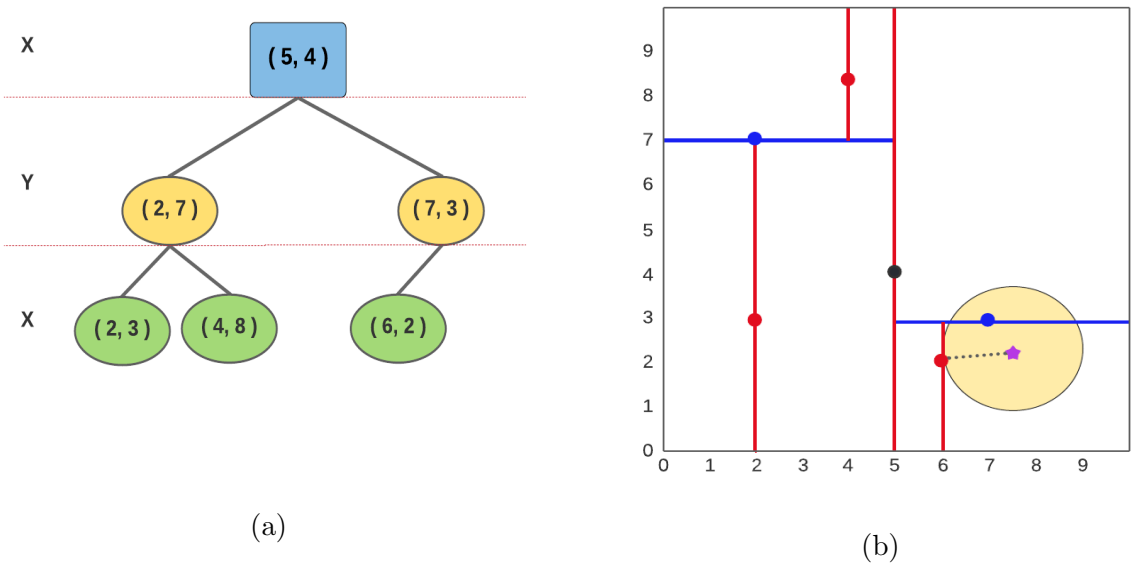
### 3.3.2 Kd-trees

A Kd-tree is a multidimensional binary search tree introduced by Bentley in 1975 [155], generalising the standard one-dimensional binary search tree. Later on, the optimised Kd-tree proposed by Friedman and Bentley [150] emerged as a valuable tool in Euclidean spaces of moderate dimensions.

Over the years, further improvements for high dimensional settings [156], distribution adaptation [157], and incremental searches [158] were proposed for Kd-trees.

The storage and the construction time of Kd-trees of $N$ nodes are $O(N)$ and $O(N \log N)$, respectively [150].

Multidimensional binary search trees, or Kd-trees, allow a wide variety of searches, including exact-match, partial-match, and range queries [155]. Additionally, Kd-trees can also be used as a sorting algorithm. Enhanced with many of the improvements described in [159] (allowing approximation errors, incremental distance calculations, and priority search), the Kd-tree is a data structure well suited to k-NN searching on most data sets.




(a) (b)

Figure 3.1: (a) The Kd-tree of the points (2,3) (2,7) (4,8) (5,4) (6,2), (7,3). The root node is (5,4) since it is the median along the x-axis. It splits the whole plane into left and right parts. Then the left and right subtrees split along hyperplanes based on the medians along the y-axis. The process continues until all the points are used to build the tree. (b) Shows how the Kd-tree splits up the $x, y$ plane geometrically. Here, the black node is the root node, nodes one level down are blue, and nodes two levels deep are red. The star indicates a query point. The shaded area shows the candidate hypersphere that may contain closer points to the query than the current guess.

Kd-trees organise $K$-dimensional vectors of numeric data. Each internal node of the

tree represents a branching decision in terms of a single attribute's value, called a split value [160]. These internal nodes generate a splitting hyperplane that divides the space into subspaces using this split value, usually the *median value* along the splitting dimension. We will use the median when constructing the Kd-tree for the data in our experiments.

Figure 3.1 shows an example Kd-tree and how it splits up the $X, Y$ plane geometrically. For the given dataset, all data with values less than or equal to the median along that dimension (points left to the hyperplane) are incorporated into that node's left subtree. All data with values larger than the median along that dimension (points right to the hyperplane) are incorporated into the right subtree. Each leaf is a collection of data items [15, 160].

The NN search algorithm aims to find a node in the tree closest to a given input vector. This search is efficient with an $O(\log N)$ complexity, where $N$ is the size of the Kd-tree, i.e., the number of nodes in the tree. For k-NN search, the complexity is $O(k \log N)$. The NN search uses tree properties to quickly eliminate large portions of the search space [15]. Figure 3.1-(b) illustrates the process of finding the nearest neighbour to a query point in the Kd-tree shown in Figure 3.1-(a).

The star indicates the query point, and the current guess is connected with the dashed line. The circle is the candidate hypersphere, which lies entirely on the left side of the splitting hyperplane running vertically through the tree's root. Any point in the right subtree cannot be nearest to the current estimate. However, the left subtree still needs to be investigated since the hypersphere intersects with a few hyperplanes on the left side of the tree's root.

The NN search starts by descending from the root node recursively to a leaf node and saving it as the nearest point. Then the algorithm unwinds the recursion of the tree, searching back up the tree to find whether each intermediate node is better than the current estimate. In each update of the nearest point estimate, the algorithm checks if the hypersphere around the current guess intersects a splitting hyperplane. If not, the algorithm prunes the search of the other side of the hyperplane and walk back to the next node. Otherwise, it walks down the other branch of the tree from the current node, searching the nearest nodes following the same recursive process as before.

Similarly, for the k-NN search, the algorithm can be extended to maintain a list of the $k$ best points using a priority queue. For more details on Kd-trees and k-NN search implementations, refer to [159].

Many tree-based indexing structures were initially proposed in the context of lower-dimensional applications. In general, the performance of most tree-based index structures degrades rapidly with more than 15 to 20 dimensions [161]. Similarly, Kd-trees generally work well for low dimensional problems and suffer as dimension increases. However, many optimisation methods are proposed to improve the efficiency of a KD tree for higher dimensions [159]. We will discuss the suitable dimensions and the performance of Kd-tree

in our application in Chapter 5.

We used the concepts of embedding entity records and similarity searching in configuration space (a vector space) to develop an indexing approach for fast query processing in large-scale ER tasks. We formulate our indexing method to generate blocks that group close-by objects using a Kd-tree and a k-NN search. We discuss the proposed indexing method in detail in Chapter 5.

## 3.4 Summary

In this chapter, we formalised the concepts of ER that are relevant to our work. We also presented the formal definitions of the individual tasks that correspond to the main contributions of our thesis. MDS is used as an embedding technique to propose our metric-based ER solutions. We will be using the definitions of LSMDS and OSE in the following chapters. We used LSMDS to generate vectors that approximate name strings and their properties in Chapter 4. The application of Kd-trees and k-NN search for developing a fast indexing technique for real-time and approximate query matching is proposed in Chapter 5.

# Chapter 4

# Generating Name-like Vectors for Testing Large-scale ER

The demand for scalable ER techniques has increased tremendously to facilitate the integration of large-scale data collections. However, the shortage of training and ground truth data impedes the development and testing of ER algorithms. Good public datasets, especially those containing personal information, are restricted in this area and usually small in size. In this chapter, we propose a simple, inexpensive, and fast synthetic data generation tool.

## 4.1 Introduction

The development of large-scale ER techniques requires large-scale testing data. Developing novel algorithms that process, integrate, or analyze data with personal identifying information in research is often difficult due to growing concerns over data privacy and identity theft. These concerns prevent the publication of datasets that contain personal information. Even if we are allowed to use such datasets for testing our algorithms, they cannot be published for replication studies. We discussed the lack of data problem in detail in Chapter 1 and Chapter 2.

There are several large-scale, real RDF (Resource Description Framework) Web datasets that could be used as ER benchmarks [162]. However, only a few real datasets are available for research that contain personal information; these are small in size [1, 163]. Even with available real datasets, the correct status of two records matched across two databases is unknown to many applications. The main reason is the lack of ground-truth or 'gold standard' data that specifies whether a given two records correspond to the same entity or not. Therefore, validating matched and linked results in the testing of newly developed ER algorithms is very difficult. Data simulation using similar characteristics of real datasets is the alternative approach in the absence of suitable datasets.

A good dataset suitable for ER tasks needs to ensure quality. The soundness of the Gold standard, error diversity and the number of exact duplicates in a dataset are important factors to define a good dataset in this context [9]. Similarly, according to Panse *et al.* [9], a suitable test dataset has to ensure quality (able to provide meaningful evaluation results), usability (able to customize), and reproducibility (able to reproduce previous evaluation results) as well.

Our motivation is to mimic the characteristics of large-scale data containing personal information that are useful in deduplication, fraud detection, cloud computing, and health informatics in our simulation model to generate large-scale datasets [1]. We also aim to ensure the desired properties of such data, e.g., quality, usability and reproducibility.

There are many advantages in using simulations as part of the development process, where algorithms will ultimately be tested on real datasets [164]. A key benefit of simulated data is that they contain ground truth data critical to evaluating ER algorithms. We can also control the properties such as size, linking rate, types of errors, and error rates in data simulation. Moreover, simulated datasets can be published with their source programs/tools, allowing others to adapt them in their application domains.

However, generating synthetic data is a nontrivial process since the data are expected to have similar characteristics to real-world data, such as distributions of values, errors, noise, and variations. Many researchers implement ad hoc methods/tools to simulate data. These are often complex, application-specific data generators with limited capabilities to generate large-scale data. Similarly, many of the existing personal data simulation tools do not support large-scale data generation and have problems in complexity, scalability, and limitations of resampling.

We propose a simple, inexpensive, and fast simulation model that captures and approximates the most relevant properties of one common identification key, specifically names. String comparison functions measure the distance or the dissimilarity between string attributes such as names in record matching. These values indicate how similar attribute values are for the underlying pairs of records. The main property of interest is the distance or the dissimilarity between attribute values when deciding whether two entity records correspond to the same entity. Therefore, our model simulates the level of detail required to construct the distance between identification keys. We use vectors of numbers to represent names in a dataset, aiming at low-dimensional vectors. The model outputs a set of vectors that can be used to evaluate the performance of big data solutions, especially those that involve name matching. We will refer to these vectors as *name-like vectors* for the rest of the chapter.

This chapter uses the concepts of name approximation in Euclidean space to propose a simulation tool that generates name-like vectors. One of our significant contributions in this thesis is the method of approximating vectors using dissimilarities between the actual names. In Chapter 5, we discuss how we utilise the vector approximation of names to develop metric-space base indexing techniques. The ideas and the data used in the name

approximation are used to propose a solution for large-scale MDS in Chapter 6. We will discuss more details in each relevant chapter.

Our method holds several attractive properties compared to traditional simulation tools. These simulation tools resample from real databases or datasets that contain information about entities. Thus, the size of the existing databases limits the size of the datasets a particular tool can generate effectively. A small list of names and addresses can generate a large synthetic dataset of entities. However, the number of combinations of names and addresses that can be used to generate unique entities is limited. In contrast, our solution relies on a vector representation that defines records, where the capacity of resampling from vector space is unlimited. We discussed some other attractive properties of vectors useful in our ER solutions in detail in Chapter 3.

Error modelling is an essential and complex component of test data simulation in ER since errors are inherent in real-world databases. The quality of entity identifying information (quasi-identifiers) in real data can be low for reasons such as incorrect, incomplete or outdated details that could lead attribute values to be erroneous. Generating those errors in data simulation requires decisions such as types of errors (e.g., spelling or phonetic errors) and the probabilities of introducing and positioning errors. However, the most common types of errors considered in many simulation tools are typographical errors. Our simulation model generates typographical errors of names using only one parameter: the *variance of errors* in initially considered name-like vectors in the underlying namespace. We introduce small typographical errors with small distances away from name-like vectors, and the same could be applied to generate large errors to simplify the error simulation process.

Record generation is often a complex and expensive process in traditional simulation tools requiring user-defined probabilities, tuning several parameters, and other modifications to entity records. In contrast, our model generates entity names with a fixed $O(K)$ time per record, given that the dimension of a vector representation is $K$ (which is small). We can generate one million name-like vectors that represent records in less than 2 minutes. The simplicity of our simulation model allows inexpensive and fast data generation.

Many modern ER applications require matching or linking datasets that would contain 10 million or 100 million names. Consider, as a motivating example, linking medical records spread across different health databases. A person can have many entries in those databases over several years. Thousands of record comparisons may be needed to link entries belonging to that person. For a country's population, this may mean millions or even billions of records. Our interest is in generating datasets of 10-100 million name-like vectors.

One example where such datasets are needed in testing is when developing matching or linking ER algorithms that use global information rather than simplistic pairwise matching. In that context, we need to understand how the global distribution of names

or other personal information and their errors impact the algorithm.

We developed our simulation model following data analysis of a namespace that contains entity names collected from the actual environment. The main property of interest is the dissimilarities or the distances between name strings. Hence, our model attempts to simulate name-like vectors such that these vectors preserve the distance between the names strings we measured in the real namespace. Towards this, we used a set of existing statistical methods and tools for the data analysis. Based on the results of the data analysis, we develop our numerical simulation model that generates name-like vectors.

Three surname datasets are used as inputs for the experimental data analysis. The results closely mirrored each other (details in Section 4.4).

The proposed numerical simulation model uses a normal distribution to generate name-like vectors in a low-dimensional Euclidean space (6 to 8 dimensions).

The rest of the chapter discusses the methods, experimental analysis, and the resulting contributions toward answering two key questions: (a) "Can we use vectors to approximate the namespace?", and (b) "How do we simulate name-like vectors including errors of a namespace?"

## 4.2 Related Work

In this section, we survey a few relevant works that align with the focus of this chapter.

There are frequently used datasets in the ER literature: a health data set containing midwife data records [165], the North Carolina Voter database (NCVR) [166] and CORA [1], a citation network. Besides, only a few 'real' datasets that contain personal information are available for research. However, each of these has a limited size, problems in resampling, and limited control over errors.

Arehart and Miller [163] developed a ground truth dataset that contains approximately 70,000 culturally diverse Roman names. The dataset includes personal names that were drawn from different sources and manually incorporated name variations.

Significant, recent work on generating realistic test datasets for duplicate detection is proposed by *Panse et al.* [9]. The authors offer a large-scale dataset of 120 million records extracted from a historical voter register from NCVR for evaluating duplicate detection algorithms. In addition to the test dataset, this work also provides an approach to generate and store test data based on NCVR as well. As far as we are aware, this is the first work on generating realistic test data for duplicate detection at this scale.

Several simulation tools have been developed in ER literature. These tools aim to simulate realistic records and common errors. Hernandez [89] presented the first work on data generation with duplicate records based on real-world error distributions. Bertolazzi *et al.* [167] addressed some limitations of Hernandez's method, including missing

[1] https://relational.fit.cvut.cz/dataset/CORA

values and various corruption functions.

Christen *et al.* [164] presented a tool known as Geco, that generates records with entity attributes such as names, addresses, and dates of birth, which has been widely used in ER research. However, it does not support large-scale data generation. Talburt *et al.* [168] presented a tool for generating records that represent people's residential occupancy histories. Several previous works on simulation studies were proposed by Tromp *et al.* [169] and Bachteler *et al.* [170]. These simulation tools are often application-specific, where the researchers proposed various ad hoc methods that generate datasets with specific characteristics.

Most of the above simulation tools contain two essential components. First, the data are generated by repeatedly sampling from an existing dataset or database. This limits the size and controllability of the data simulation. Second, these tools introduce errors using models of standard errors, e.g., typographical, phonetic, and OCR (optical character recognition) errors. The differences lie in the way they simulate records and errors using different structural models.

The frequency and look-up tables used in the record linkage system, FEBRL [171], have been regularly used by other simulation studies to generate records. Although this is a broadly used open-source approach, it is not scalable enough to explore ER algorithms that work with very large datasets.

Our work is motivated by the vector-based representation of words, which has a thriving history in the applications of information retrieval, computational semantics, and natural language processing. The underlying property is that words with similar contextual significance tend to have similar embeddings when mapped to a continuous vector space [172]. Word2vec [173], in particular, has been used extensively in natural language processing applications. However, it is less useful in proper-noun databases since *names* do not have semantic relationships [174].

Several methods that embed a set of strings in a metric space have been used to explore the problems in ER [16, 15]. These methods focus on matching similar records efficiently using the properties of the metric space. However, to the best of our knowledge, none of them addressed the data simulation using a metric space, particularly a vector space.

Several mapping algorithms, including StringMap [16], FastMap [137], SparseMap [15], and MetricMap [138], are proposed for embedding a set of objects in a metric space. While the goal is to find similar records by mapping them into a metric space, none of these addresses string (e.g., names) approximation or data simulation approaches. These mapping algorithms are used to propose different indexing techniques for ER, and we will discuss them in Chapter 5, Section 5.2.

We choose multidimensional scaling (MDS) because of its ability to work with the nonlinear and non-metric nature of our data and because it includes a distance preservation capability for large-scale data with a small amount of extra effort. We will discuss more details on other mapping algorithms in Chapter 5.

Table 4.1: Summary of the main notations used in this chapter.

| | |
|---|---|
| $N$ | Input data points to a MDS algorithm |
| $\Delta$ | Dissimilarity matrix for $N$ |
| $K$ | Dimension of a multidimensional scaling configuration |
| $N \times K$ | Configuration space (MDS output) |
| $X$ | Coordinates of $N$ points in configuration space |
| $D$ | Distance matrix for $X$ in Euclidean space |
| $\sigma$ | Stress of a configuration, defined |
| $\overline{x}$ | Mean of a multivariate dataset |
| $\Sigma$ | Covariance of a multivariate dataset (defined in (3.2)) |
| $\Sigma_s$ | Covariance of name-like vectors |
| $\Sigma_e$ | Covariance of errors of name-like vectors |
| $\Sigma_s^{1/2}$ | Inverse matrix of $\Sigma_s$ |
| $\Sigma_p$ | Pooled covariance estimate |
| $\gamma_1$ | Relative eigenvalue of the maximum covariance ratio analysis |

## 4.3 Methodology

Our methodology is organised into three main sections. In each section, we answer the questions defined in Section 4.1. In Table 4.1, we summarise the terminology and the notation used in this chapter.

### 4.3.1 Converting Names to Multidimensional Vectors

We first aim to find a simple representation for names that allow quick simulations. In name matching, the main property of interest is the similarities or dissimilarities of names. We assume that names reside in a complicated high-dimensional space (the namespace) where similar names are closer than dissimilar names. The first key question is: "Can we use lower-dimensional vectors to approximate the namespace?" In other words, we seek to approximate a high-dimensional space with a lower-dimensional Euclidean space.

We cast the problem as an embedding problem, which refers to constructing coordinates from distances (see Section 3.2). In domains that use computationally expensive distance functions, significant speed-ups can be obtained by embedding objects into a coordinate space and applying an efficient distance function, such as the $L^p$ norm [121].

MDS is a non-linear optimisation problem for finding the best mapping of $\phi$ [122]. Hence, we can project names into a lower-dimensional Euclidean space by approximating their dissimilarities using MDS. Then the pairwise comparisons between names become Euclidean distances between vectors.

As mentioned earlier (See Chapter 3, Subsection 3.2.1), CMDS and LSMDS are the

two popular metrics based MDS approaches. CMDS works well with Euclidean data inputs. The name dissimilarities we used in our work has a non-Euclidean nature, and we found that LSMDS works well with such data as inputs.

We applied LSMDS to determine a simple vector representation in a lower dimension for approximating characteristics of a namespace. This lower-dimensional Euclidean space is a configuration space containing $N$ data points in $K$ dimensional space. In this chapter, we call this an *approximated namespace* that contains *name-like vectors*.

To that end, we need to define an appropriate dimension $K$, in which distances are maintained at a suitable level of accuracy. We used two approaches to discover a reasonable lower dimension that fits our data. First, we analysed the rate of change of the normalised stress $\sigma$ against $K$-dimensions, where a lower $\sigma$ value indicates a better approximation. Second, we used a Shepard diagram [3] to determine the goodness-of-fit of LSMDS results. A Shepard diagram is a popular way of assessing the goodness of fit of data reduction or embedding techniques such as MDS and t-SNE (t-distributed stochastic neighbour embedding) [3]. It compares how far apart our data points are before and after the transformation using a scatter plot. We present the results in Section 4.5.

Figure 4.1 explains the basic workflow that creates the foundation of the proposed simulation model using an example of how textual similarities of strings relates to vector similarities. For a given set of names, $\Delta$ represents the dissimilarity matrix of Levenshtein distances. By applying LSMDS, we project the name strings into a lower-dimensional space such that $D \approx \Delta$. We choose the dimension $K$ of the lower-dimensional space that provides the best compromise for our data using the stress values.

In summary, our simulation model is designed based on data analysis of an actual namespace that studies the relationship between names in a Euclidean space. Then the model reconstructs these relationships, especially distances between the names to generate large-scale name-like vectors. Next, we discuss the name-like vector simulation in detail.

### 4.3.2 Simulating name-like vectors

If we can approximate a namespace in a $K$ dimensional Euclidean space, this leads us to our second question, "How do we simulate name-like vectors using the approximated namespace?" A simple approach is to determine the distribution of the approximated lower-dimensional vectors and simulate vectors with similar characteristics.

We start by checking the normality of the name-like vectors since it is generally a good starting point. The normal distribution makes simulation easy, requiring only two parameters: *mean* and *covariance*. In our context, these parameter values are vectors and matrices rather than single values due to the multivariate nature of the data. Hence, we explore the multivariate characteristic of the real namespace and the approximated name-like vectors in the Euclidean space.

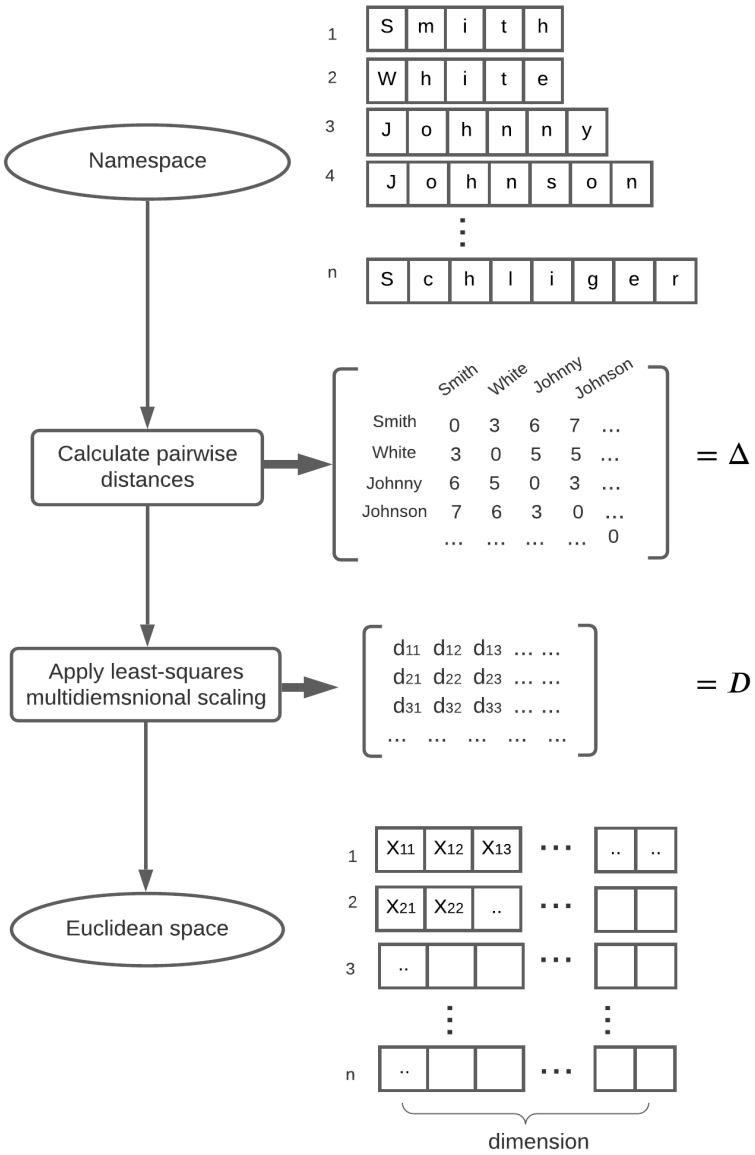



Figure 4.1: The basic workflow of our numerical simulation model. Names come from a complicated high dimensional space, and we can represent them in a lower-dimensional Euclidean space by applying LSMDS to name dissimilarities. For instance, $\Delta$ represents the dissimilarity matrix containing the Levenshtein distance between the given names. By getting $\Delta$ as the input, LSMDS project the names into a lower-dimensional space, approximating their initial Levenshtein distances by Euclidean distances, i.e., $\Delta \approx D$. Then, the names become vectors in $K$ dimensional space, where $K$ must be chosen.

### Multivariate Normal Characteristics

Many parametric multivariate statistical methods, such as linear discriminant analysis, require the data to hold the multivariate normality (MVN) assumption. These methods

produce more reliable results if the data are exactly or even approximately multivariate normal.

Many analytical and graphical methods test the goodness-of-fit of a dataset to the multivariate normal distribution. However, choosing which test in practice is difficult since different approaches may give different conclusions about the MVN of a dataset. Therefore, it is usually recommended to perform several tests while examining the results produced by graphical methods [175, 176].

We applied Mardia's skewness and kurtosis statistical tests to check the multivariate normality of our data. Additionally, we used graphical approaches such as histograms, density plots, and quantile-quantile plots (Q-Q plots) to visually assess the MVN characteristics of the data [176]. Moreover, we applied a multivariate outlier detection method based on the Mahalanobis distance. It demonstrates how far an individual data point is from the centroid of all points for the underlying variables. An observation is considered an outlier when the distance is great [177].

In addition to MVN, we applied univariate normality tests and plots to diagnose any deviation from normality. Investigating the univariate normality of the underlying variables builds a foundation for a complete understanding of MVN. In the univariate setting, normality tests such as Kolmogorov-Smirnov (K-S) and Shapiro-Wilk are extensively applied in practice [176]. Furthermore, we used histograms and Q-Q plots to assess univariate normality visually.

Based on the MVN analysis, we found that the approximated name-like vectors in the Euclidean space do not strictly follow a normal distribution. However, the departures from normality were relatively minor. Hence, the normal distribution is acceptable as an adequate approximation to simulate name-like vectors.

### Parameter Selection and Simulation

Name-like vector simulation is a two-step procedure. The first step is estimating the parameters (mean and variance) for the normal distribution that we would choose as our simulation model. Since our data are multivariate, we calculated the sample covariance matrix ($\Sigma$) [76]. All these parameters are estimated from an approximated namespace that contains the name-like vectors.

The second step is the simulation of name-like vectors using the estimated parameters of the normal distribution $N(\overline{x}, \Sigma)$, where $\overline{x} = 0$. In this way, we can generate many name-like vectors within seconds.

We evaluated the results of our simulator based on a comparison between different namespaces. Those are the *real namespace, the approximated namespace,* and *a simulated namespace*. Our interest is in the distributions of the distances in each namespace since it is a key property in the namespace approximation. The comparison explains how well a normal distribution can reproduce the distances between vectors mimicking the distances in an actual namespace.

The other characteristic of interest is the possible errors of an actual namespace.

### 4.3.3 Error Simulation in Name-like Vectors

A namespace can contain errors such as typographical, OCR, and phonetic errors. We wish to include some of these errors in our simulated namespace. Consequently, we come to the subquestion: "How do we model errors in a simulated namespace?" When simulating real name strings, the typical process would be to build a complicated error generating model. In contrast, we create errors of simulated name-like vectors by adding some noise to the simulated namespace.

Errors are variations of original names. Distances between a name and its variants are typically small compared to the distances between distinct names. Similarly, the typical distance between a name-like vector and that containing an error should be small. In our error model, we capture the variance of vectors that represent original names and their errors. Then, we use it to simulate errors of name-like vectors with a small Euclidean distance away from their initially simulated name-like vectors. Our prime interest is in edit distance errors such as insertions, substitutions, deletions, and transpositions since they are the most common errors in real datasets [178].

Similar to the name-like vector simulation, we started with the approximated namespace that contains name-like vectors. However, unlike the previous approximation, we selected an actual namespace containing names with edit distance errors. Hence, the original names and their errors are converted into name-like vectors in the Euclidean space. In this way, we can detect the distribution of errors with respect to their initial name-like vectors in the Euclidean space.

#### Maximal Ratio of Covariances

The *maximal ratio of the covariance* test is a useful way to report the differences between two variance-covariance structures [179]. The value of the ratio quantifies the two covariances. We compare the covariance of the two groups: approximated errors of name-like vectors and name-like vectors.

Let $\Sigma_s$ be the $N \times N$ covariance matrix of name-like vectors and $\Sigma_e$ be that of errors of name-like vectors and assume both matrices are invertible. Given that $\Sigma_s^{-1/2}$ is the inverse square root matrix, the ratio can be written as a product of one covariance matrix and the inverse of the other covariance matrix ($\Sigma_s^{-1/2}\Sigma_e\Sigma_s^{-1/2}$). Relative eigenanalysis or relative principal component analysis determines the direction along which the ratio of variances between two groups is a maximum [179]. This direction is the first eigenvector of $\Sigma_s^{-1/2}\Sigma_e\Sigma_s^{-1/2}$, also known as the first relative eigenvector of $\Sigma_e$ with respect to $\Sigma_s$. The eigenvalues of $\Sigma_s^{-1/2}\Sigma_e\Sigma_s^{-1/2}$ are called the relative eigenvalues of $\Sigma_e$ with respect to $\Sigma_s$. Thus, the first relative eigenvalue $\gamma_1$ is equal to the maximal ratio of variances. Small $\gamma_1$ values indicate that even in the worst-case scenario, the errors have small variances.

If we can find a small $\gamma_1$ for approximated name-like vectors and their errors, it implies that we can add Gaussian noise to simulate random errors in a simulated namespace. However, our experimental results supported this expectation. We calculated the Gaussian noise through a second normal distribution $N(0, \Sigma_e)$, different from the initially estimated normal distribution. We will discuss the results in the next section.

## 4.4 Experimental Results

We conducted a set of experiments on real datasets in order to evaluate the potential benefits of the proposed solution. All algorithms are implemented in R and executed on a desktop with Intel Core 5 Quad 2.3 GHz, 16 GB RAM, and MacOS Catalina.

### 4.4.1 The Data Sets

We used three datasets of surnames to explore namespaces and their properties, e.g., simple relationships such as the distributions of names and the dissimilarities between name strings. In general, all datasets exhibit the same characteristics. For instance, they follow a Zipf's distribution [180].

1. Ancestry data: The first dataset is from Ancestry.com [181], which contains the 250k most commonly occurring surnames. It is the largest unique surname repository we found for this study.

2. FEBRL data: The second dataset is derived from the frequency files included in the dataset generator program in FEBRL [171]. The frequency table contains 9000 unique surnames extracted from telephone directories in Australia.

3. US census data: The third data is from the US census [182] that contains all surnames occurring 100 or more times in the 2010 census. It includes 150,000 distinct surnames.

### 4.4.2 Experimental Setup

This section presents the results of the ideas introduced in the previous section to real data, particularly the results based on the first dataset. Since our data analysis observations and the simulation results were similar across the three datasets, we will not discuss them separately. In this chapter, we used 5000 randomly selected surnames for testing, which allowed us to perform resampling to evaluate the results for consistency.

The outcome of the proposed simulation model does not depend on the individual observations of the selected random sample. Hence the sample selection is flexible. Initially, we started with a distinct set of names, as we first attempted to simulate name-like

vectors without errors. However, later on, the simulation of errors for name-like vectors uses a dataset that contains real-world errors.

One can use a range of standard string distance measures to calculate the pairwise dissimilarities between surnames. We used the STRINGDIST R package [75] to calculate the dissimilarities between name strings. Initially, we considered five frequently used string dissimilarity measures [82, 14]: Levenshtein distances (LV), longest common subsequent (LCS), Jaccard dissimilarity, Q-grams, and Jaro-Winkler distances.

The first step of the data analysis is to choose a dimension $K$ that fits our data, where we would like to make $K$ as small as possible. Hence, the following stress analysis determines a $K$ that produces the smallest and the best approximation that suits the given data. Note that there is ambiguity in the definition of dissimilarity between names. Hence, a certain degree of error can be tolerated, especially if it is small compared to the differences between dissimilarity measures.

### 4.4.3 Stress Analysis

Figure 4.2 shows the stress-$\sigma$ against dimension-$K$. The stress $\sigma$ (defined by Equation 3.2) tends towards a small but nonzero asymptote for most dissimilarities, reflecting the non-Euclidean nature of the original data. The stress values of the LV and LCS distances are overlaid, and it is hard to visualize them separately. The JW distances illustrated in the inset of Figure 4.2 are different because JW tends to rate all dissimilar names as being the same distance apart, leading to large nonlinearity in LSMDS. Hence, we argue that the approximation of a namespace in the Euclidean space using LSMDS works for all but the JW dissimilarities.

Ideally, we seek an elbow in the *dimension vs stress* plot. In practice, however, such elbows are rarely visible. Since the $\sigma$ is non-zero, we find a reasonably small $K$ around 6-8 dimensions considering the best trade-off between them. We will test $K$=6 in what follows.

However, when we have non-zero $\sigma$, we should keep in mind that the distances among vectors are imperfect and distorted representations of the relationships given by our data. The LMDS transformation does not need to have zero $\sigma$ to be useful in our application since we are willing to tolerate a certain degree of distortion. There are different standards regarding the amount of $\sigma$ to tolerate [122]. Our rule is that any $\sigma$ value under 0.1 is excellent, and anything over 0.2 is unacceptable.

### 4.4.4 Shepard Diagram Analysis

An accurate dimension reduction or embedding technique will produce a straight line that goes through the origin in a Shepard diagram [3]. However, in practice, Shepard diagrams rarely look straight due to information loss during data reduction.

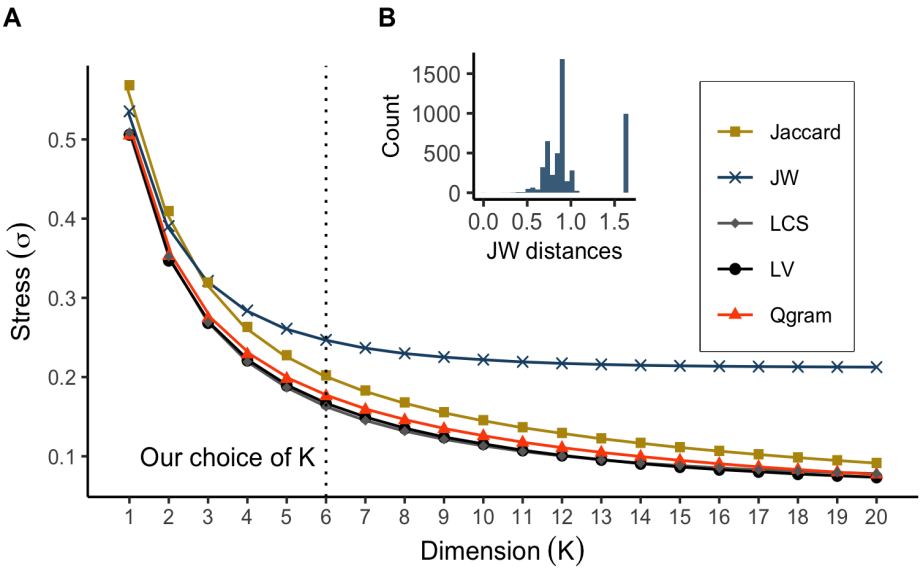


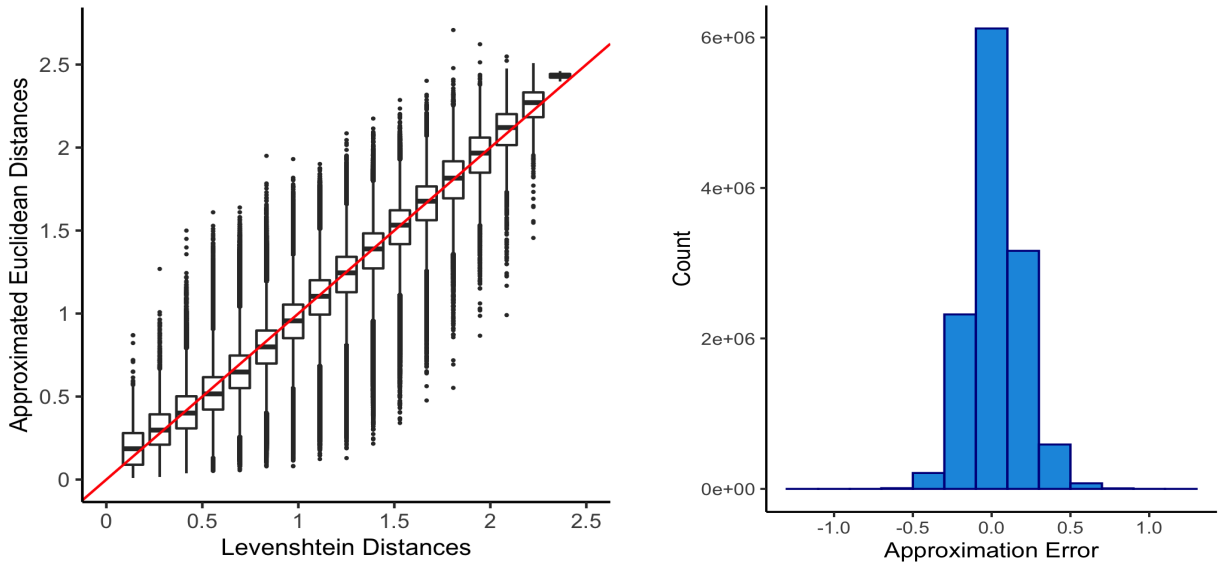

Figure 4.2: A) The trade-off between *stress* and the *dimension* for JaroWinkler (JW), Jaccard coefficient, Longest common subsequence (LCS), Levenshtein (LV) and Qgrams (q=2). B) The distribution of JW distances. Compared to the other distributions, JW distances exhibits a spike where all the dissimilar surnames take one value.

Figure 4.3-(a) compares the Euclidean distances against the LV distances in $K = 6$. There is a strong linear relationship between the two distance distributions. Hence, we confirmed that LSMDS successfully approximated the LV distances between real names into the Euclidean distances among the name-like vectors. Since the $\sigma$ is not zero, there are some errors in the approximation.

We looked at the distribution of those errors in the approximation in Figure 4.3-(b). It shows the differences between the LV distances and the corresponding approximated Euclidean distances of each point shown in Figure 4.3-(a). By looking at the histogram, we can see that approximation errors are small, less than $|0.5|$ for the majority.

We compared Shepard diagrams obtained for 8, 10, 15, and 20 dimensions using the same dataset to see the performance of the distance approximation in higher dimensions. The comparison confirmed that more dimensions do not significantly increase the linearity of the LSMDS approximation. The variations around the line decrease, but the line does not fit better. Considering the trade-off between the simplicity of the model and the accuracy of the approximated distances, we selected the $K$=6 Euclidean space for our simulation model.

We preferred LV distances for further analysis, as they are popular among string distance metrics and perform well with the LSMDS approximation. However, the results with other metrics (except JW) mirror those presented.




(a) (b)

Figure 4.3: (a) The Shepard diagram [3] of the LV distances vs Euclidean distances in $K$=6. There is a strong linear relationship between the two distance values as desired. (b) The distribution of the approximation errors. The errors are the differences between the LV distances and the approximated Euclidean distances of each point shown in (a).

### 4.4.5 Multivariate Normal Analysis

We first evaluated the multivariate normality (MVN) of the approximated vectors in Euclidean space. The combined test results of the multivariate skewness and kurtosis tests failed to indicate the MVN of the name-like vectors. We used a Q-Q plot (see Figure 4.4) to understand the failures of the tests. It depicts several large Mahalanobis distances (outliers) that imply possible departures from the MVN distribution, particularly in the upper tail. However, the body of the distribution mainly lies on the $y = x$ line, suggesting that a normal distribution is a reasonable model for most of the data.

To diagnose the reason for the deviation from MVN, one can perform univariate tests on each variable distribution. In addition, checking univariate plots is also very useful. However, Kolmogorov–Smirnov (K-S) and Shapiro–Wilk tests [176] rejected the normality assumptions for each of the variables in the name-like vectors.

Figure 4.5 shows the histograms corresponding to the distribution of each variable in the approximated name-like vectors. The relevant Q-Q plots showed similar results. However, the histograms and Q-Q plots suggested that the normal distribution reasonably well approximates variables 5 and 6. The other variables have slightly skewed distributions where the body still follows a moderately normal distribution.

These results suggested that the problems with MVN seem to be relatively minor for

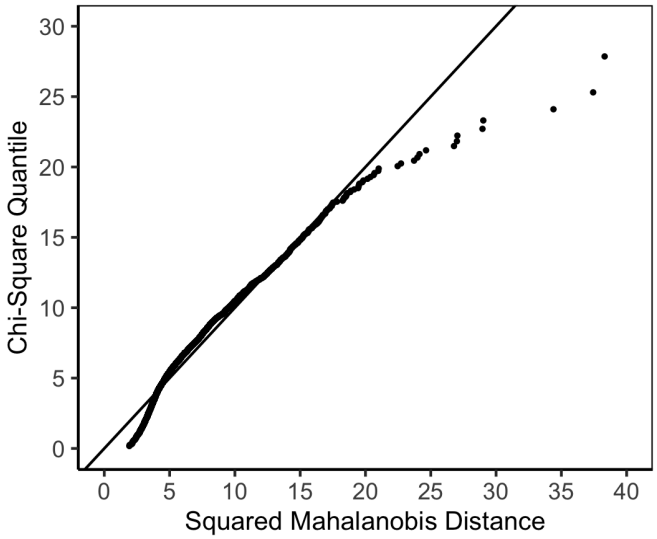



Figure 4.4: The chi-square Q-Q plot for the distribution of the all 6-dimensional vectors. There are some deviations from the straight line indicating the possible departures from a multivariate normal distribution.

name-like numbers. Even though we can consider other general distribution models for testing, it is unlikely to improve the distance approximation significantly. Therefore, we concluded that the normal distribution is a sufficient approximation model for simulating name-like vectors.

### 4.4.6 Name-like Vector Simulation

Given that the simulation model is based on the normal distribution $N(\overline{x}, \Sigma)$, we then estimated the parameters mean ($\overline{x}$) and the covariance ($\Sigma$). The unbiased estimator for $\Sigma$ is derived from the existing sample of approximated name-like vectors in the $K$=6 Euclidean space. The estimated $\overline{x}$ was close to zero. Hence, we kept $\overline{x} = 0$.

The calculated covariance matrix has values very near zero on off-diagonals. We tested for independence between each variable using a correlation test and Hoeffding's D statistics [183]. Both approaches verify that we can assume independence between the $K$=6 variables.

One test for evaluating the simulator is to compare the distribution of distances created by the simulation model to those of the original data. We generated 5000 simulated name-like vectors using the estimated normal distribution. Then, we compared the distances between simulated name-like vectors and the initial sample of 5000 surnames.

Figure 4.6 shows a Q-Q plot that compares distributions of LV distances between surnames and Euclidean distances between simulated name-like vectors. The Q-Q plot

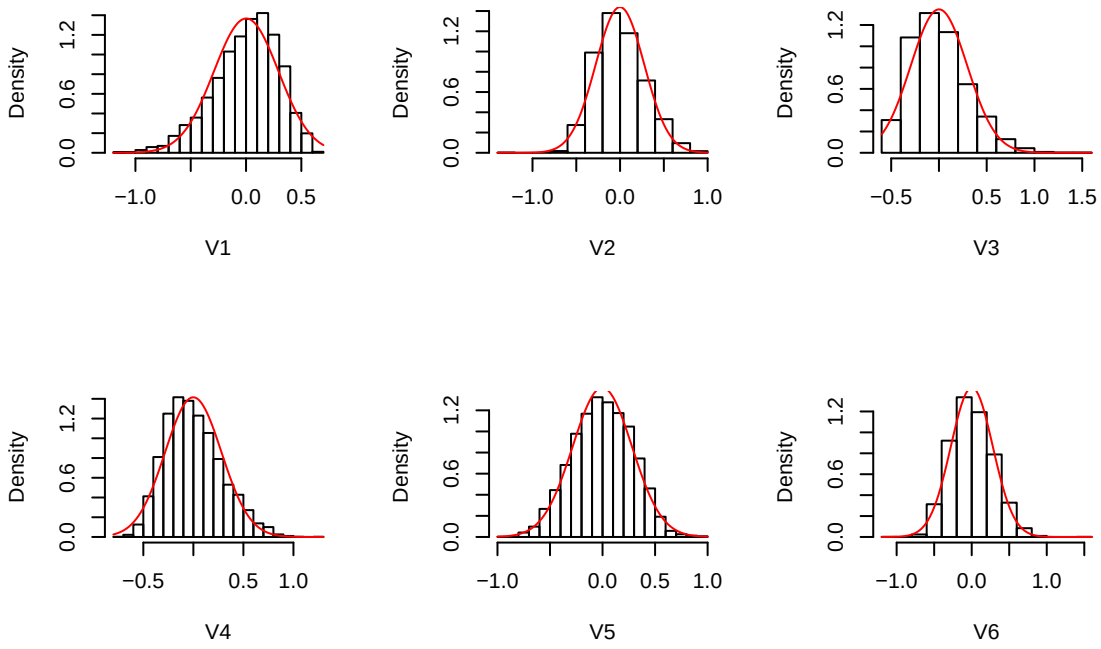



Figure 4.5: Histograms of the approximated name-like vectors. The variables V5 and V6 of the coordinates shows an approximately normal distribution while the others have slightly skewed distributions.

exhibits a reasonable similarity between the distributions even if they do not precisely fall along the $y = x$ line. We can see the discrete nature of the LV distances and the continuous Euclidean distances along with the points of the small vertical lines. There is a distortion created through the approximation process, but apart from this, the simulated vectors behave very similarly.

### 4.4.7 Error Simulation

We explored the nature of the errors in an actual namespace to simulate those in the Euclidean space. First, we looked at the variance of errors once LSMDS approximated them in a $K$=6 Euclidean space. Then, we compared the two types of vectors, approximated name-like vectors and their errors in Euclidean space, using the ratios of covariances.

A random sample of 25 surnames was selected from the initial dataset of surnames, and 100 variations of edit distance errors were generated for each of them. The errors for the original surnames are generated using Geco [164]. Then, we applied LSMDS on a total of 5000 surnames. The dataset includes 2500 distinct surnames along with 2500

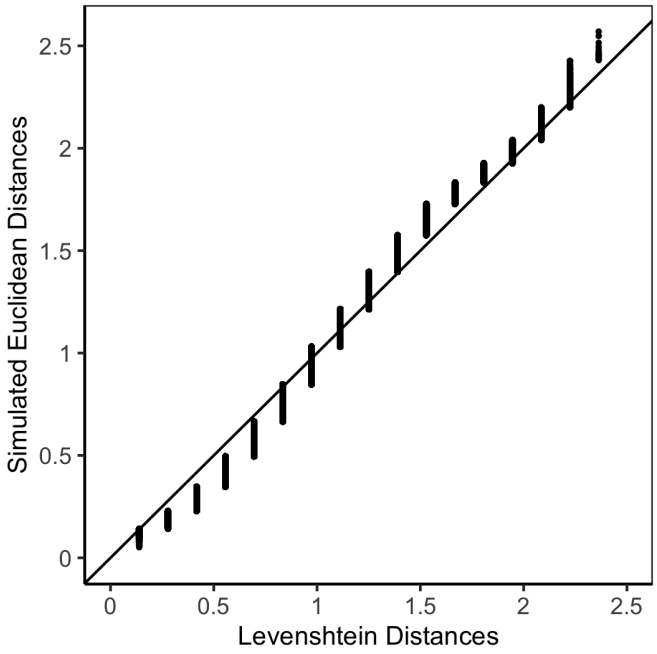



Figure 4.6: Q-Q plot comparing the distributions of LV distances of the real namespace with the Euclidean distances of simulated vectors. Points along the straight line indicate a similarity agreement between the two distributions.

errored names. We used these approximated name-like vectors to explain the variance of the errors in the Euclidean space.

We examined the variance of erroneous vectors relative to their correct version. Figure 4.7 illustrates a comparison between the distribution of a whole sample of name-like vectors (all the surnames and their errors) with a distribution of errors that belong to a single name-like vector (errors of a single surname). We have only shown results from one coordinate since the other coordinates exhibited similar results. The top histogram represents the distribution of a single variable ($X6$: the $6^{th}$ coordinate) in name-like vectors and their errors. In contrast, the bottom histogram represents the distribution of the same variable considering a single name-like vector and its errors. The small spread of the histogram that contains only the error distribution implies that we can add appropriate noise by adding a small relative variance to a name-like vector to simulate its errors in the Euclidean space.

Finally, we investigated the variance-covariance matrices corresponding to each of the erroneous surnames and their errors. These values were small compared to the variance-covariance matrix of the whole sample, indicating that these small edit distance errors are closer to their real surname values than other names. By applying the relative eigenvalue analysis to the variance-covariance matrices, we found the relevant eigenvalues and

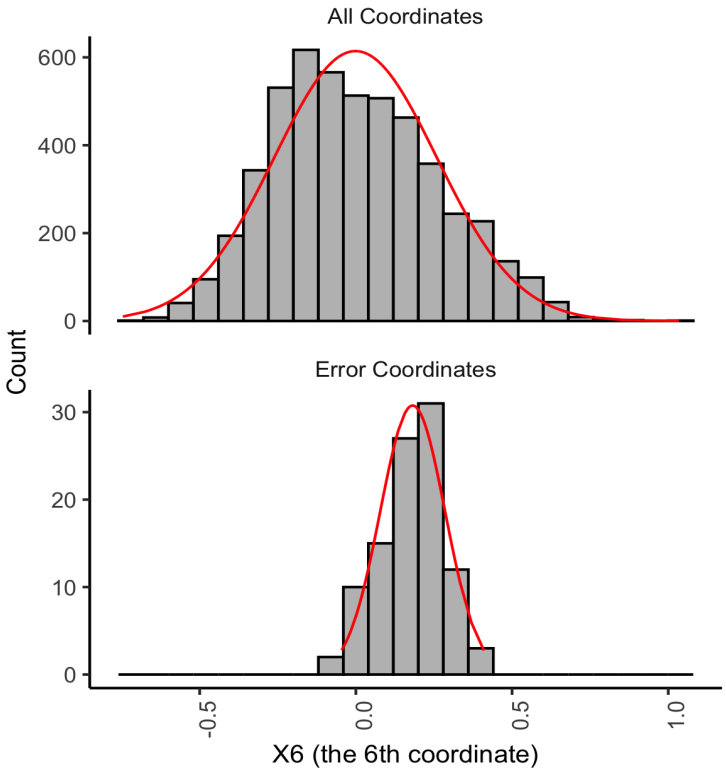



Figure 4.7: The top histogram shows the spread for a single coordinate (X6: the sixth coordinate of a name-like vector) in the whole sample (all the surnames and their errors) in $K$=6. The bottom row shows the spread of the same coordinate that considers a single surname and its errors. The spread of errors is much smaller compared to the spread of surnames and their errors as desired.

eigenvectors. The first eigenvalue $\gamma_1$ refers to the maximal ratio of covariance between the errors, and the name-like vectors took a value closer to 0.1. Therefore, even in the worst-case scenario, the errors have a small variance compared to the name-like vectors in the Euclidean space.

Hence, our model can simulate simple edit distance errors by adding Gaussian noise to an existing Euclidean space. The noise is generated from a normal distribution $N(0, \Sigma_e)$. We calculated the unbiased estimator of the covariance matrix ($\Sigma_e$) using the pooled covariance estimate $\hat{\Sigma}_p$ [179]. Therefore, we can simulate errors of simulated name-like vectors such that they preserve small distances to their initial name-like vectors.

### 4.4.8 Computational Complexity

In this section, we discuss the practical use of our simulator when developing big data ER algorithms. These algorithms require large-scale test data and efficient pairwise calcula-

tions for testing. We mainly focused on the computational time of generating large-scale name-like vectors and pairwise comparisons between them to measure the performance.

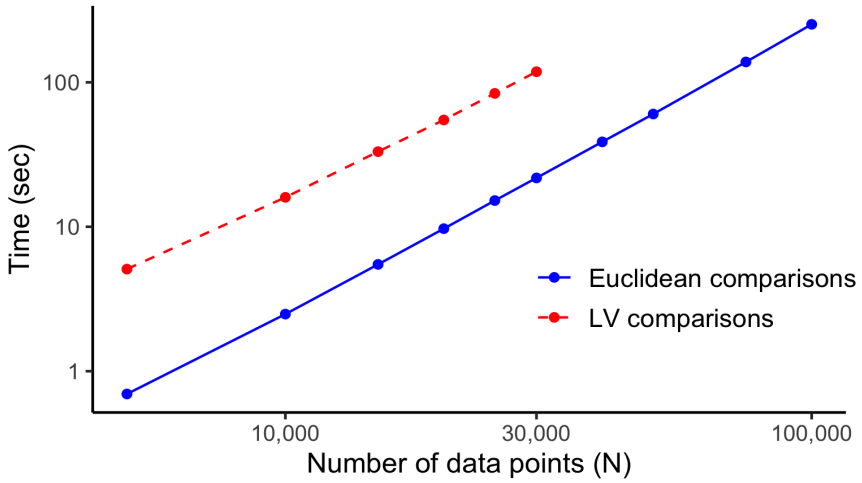



Figure 4.8: The computational time to calculate pairwise distances between name-strings and name-like vectors. Calculating the Euclidean distances is much faster than calculating LV distances.

Pairwise comparisons between records are challenging when the majority of the entity records contain strings. In contrast, name-like vector comparisons of a complete dataset would only take much lower computational time. Figure 4.8 compares the computational time of the pairwise comparisons between a dataset containing real surnames and a simulated set of name-like vectors. The surname comparisons are based on the LV distances, whereas the name-like vector comparisons are measured using Euclidean distances. The time grows quadratically for both methods, but our approach is nearly ten times faster. This efficiency is beneficial for the rapid testing of algorithms that apply to large-scale data.

We also measured the computational time to simulate datasets of different sizes. The results showed that the time grew linearly, and it took less than a minute to generate one billion name-like vectors.

### 4.4.9 Comparing Different Datasets

We used three datasets for the analysis. We tested all three of them as the input to our model in the data analysis, and the results were substantially the same.

We also used these data to understand the universality of the results. Figure 4.9 compares the distributions of pairwise LV distances of the random samples of 5000 surnames for each dataset and the distribution of pairwise Euclidean distances of a simulated

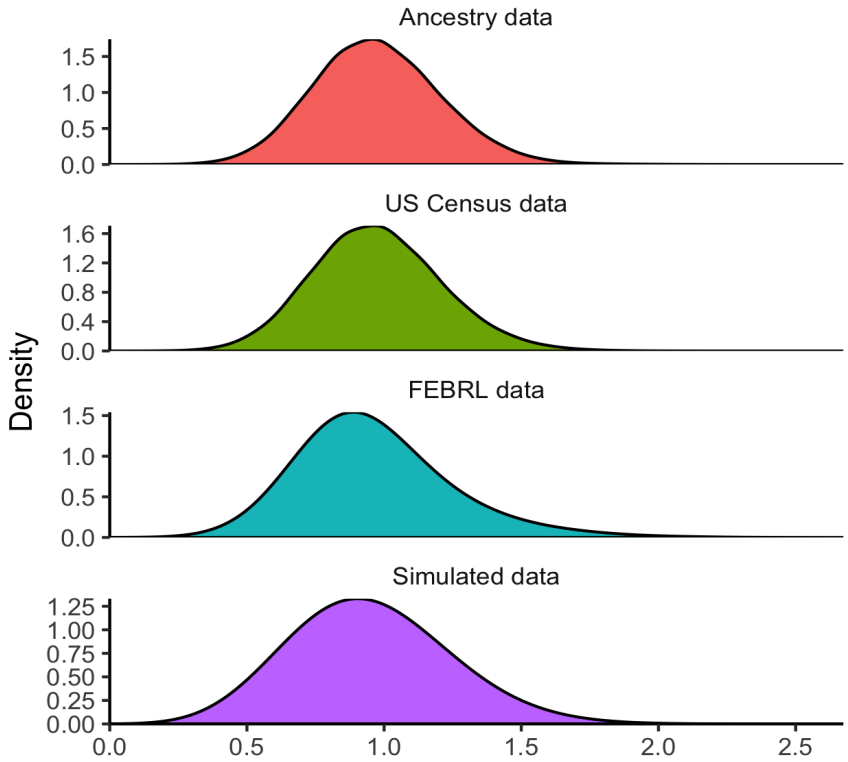



Figure 4.9: The distributions of pairwise LV distances of three datasets that contain unique surnames and the Euclidean distances of a simulated namespace. The plots are quite similar in shape sharing similar statistical properties.

dataset of 5000 name-like vectors. These smoothed density plots have similar shapes and values for *standard deviations* and *means* for the four distributions. According to the results, we can produce a simulated distribution similar to the last one by using any of the first three distributions. The only input to our simulation method is the pairwise distances between a set of strings, which we used to estimate the parameters mean and variance. These experiments show that we can adapt our simulator to work with any sample dataset derived from a string space.

## 4.5 Discussion

We can synthesise large-scale datasets of name-like vectors using our simulation model, but the model has limitations. To obtain the full benefit of the proposed simulator, we still need to model other variables required for linking, such as addresses and gender.

As a result, the dimensions of the vectors need to be extended accordingly to facilitate complete records. However, numerical or categorical data such as age or gender are easier to simulate.

Surnames (or names) follow a Zipf distribution [180], where few high-frequency names account for most of the population with many low-frequency names. We can easily modify our model to capture the frequencies of names in simulating name-like vectors by repeated sampling. Additionally, the nature of the data allows an arbitrarily large number of synthetic data points. For very large datasets, one can also increase the dimension of the space to reduce the density in which it is populated. The volume of such a space grows exponentially with the dimension.

Traditional simulation tools are often biased. For instance, existing entity identification keys such as names are typically biased towards English, Welsh, Scottish and Irish for many datasets. Resampling from these datasets to simulate records repeats those biases in the names. We aimed to avoid this bias using a vector representation in our simulation model.

The results discussed in this chapter based on the Ancestry.com [181] dataset also include biased names towards Americans and Europeans. We realized that this could distort the results and include some bias in the simulation model. Hence, we constructed the model using multiple datasets and compared the results for consistency. However, these datasets also have some biased names, which is unavoidable in many publicly available real datasets of names.

## 4.6 Summary

Our goal here was to generate synthetic data to develop and test ER algorithms appropriate for big data. We proposed a simple, inexpensive, and fast simulation model that can generate name-like vectors, including simple errors. This chapter discussed how to simulate simple vectors in a space that approximates the properties of names.

One of the significant contributions in this work is the name approximation using vectors. We used MDS to transform a set of name strings into vectors that approximate an actual namespace. This approximation shows that we can preserve the relationship, such as dissimilarities between names in a Euclidean space. Since we can approximate similar names closer and different names apart in this space, we used the concepts of name approximation to construct our metric space-based ER algorithms. Further details will be discussed in Chapter 5 and Chapter 6.

# Chapter 5

# Em-K Indexing for Approximate Query Matching in Large-scale ER

It is becoming increasingly important in real-world applications to develop ER solutions that produce prompt responses for entity queries on large-scale databases. In this chapter, we investigate the query matching problem in ER and propose an indexing method suitable for approximate and efficient query matching. We aimed to develop an indexing approach for a fast query process within a fixed time, returning as many as candidate matches.

## 5.1 Introduction

Typical ER algorithms have a quadratic running time, which is computationally prohibitive for large-scale data collections. Indexing techniques aim to reduce the potential number of comparisons between non-matching record pairs [184]. Therefore, indexing-based ER solutions run much faster than brute-force ER solutions [64].

However, most existing indexing methods still have quadratic time complexity and are too slow to deal with very large-scale data collections [60, 68, 184, 185]. Also, some non-scaling ER algorithms tend to generate a large number of blocks and a large number of candidate records in each block for large-scale scenarios [184].

Traditional ER solutions often process databases offline in batch mode, and no further action is required once a pair of matches are determined. However, many organisations are moving online where they have to provide their services through prompt responses. Hence, many newer, real-world scenarios require real-time query processing against large-scale databases. Some of the applications demand entity query matching against large-scale reference databases within a fixed time. We refer to this as the "query matching" problem in ER. In such configurations, traditional ER solutions become less efficient or unusable.

In order to motivate the problem context and illustrate the usefulness of the approach presented in this chapter, we provide the following real-world example. The application is in criminal investigations, where law enforcement officers need to query a database to identify potential suspects. Suspects usually lie to police investigators about their true identities, e.g., names, birth dates, or addresses [33]. Hence, finding an exact match for falsified data against the real identity recorded in a law enforcement computer system is problematic. Detecting deceptive identities is a time-consuming activity that involves large amounts of manual information processing in real-life scenarios [186]. Therefore efficient ER solutions that support real-time and approximate query matching against existing datasets can be invaluable.

Toward the challenge of real-time approximate query matching, we present an indexing technique that reduces the number of pairwise comparisons needed in the ER. The proposed indexing method transforms a set of records into a set of vectors in a metric space, specifically a lower-dimensional Euclidean space using MDS. These vectors have two main attractive attributes in the context. First, comparisons between vectors in a metric space are much cheaper than string comparisons. Second, these vectors support efficient indexing data structures. We utilise these properties to propose an indexing approach that classifies similar vectors in a low-dimensional Euclidean space. We call our method *Em-K indexing*, as our method operates by embedding data in a K-dimensional space.

Exact query matching is easy against a reference database as we can search based on lexicographical order using an efficient data structure, e.g., binary trees. However, exact query matching is impossible with many real-world databases due to various data quality issues, and approximate query matching is the common approach [187] (also discussed in Chapter 2). Hence we focus on approximate matching. The proposed indexing method addresses the problem of real-time and approximate query matching. We will be mainly using the term "query matching" instead of "approximate query matching" in this chapter.

Our main objective is to develop a fast indexing method for query matching. We want our method to be efficient for large-scale data and robust to errors in the data. The proposed method searches for a block of records in the reference database as a set of candidate-matches to the querying record. Hence we avoid comparing the query against all the records in the referencing database. However, the search is done in the Euclidean space using a Kd-tree data structure, where each record in the reference database and a query record become a vector in this space.

Given the block size is $B$, we search for this number of nearest neighbours (NNs) for the query point by traversing the Kd-tree nodes that store the reference database. The search has a $O(B \log N)$ complexity, given $N$ is the number of records in the reference database. The method embeds query records in a pre-mapped Euclidean space to search against the reference data points. We propose an out-of-sample embedding (OSE) technique that uses a fraction of the original data (defined as landmarks in Subsection 3.2.2) for query

record embedding. For $L$ landmarks, the embedding requires $O(L^2)$ operations, and we can choose $L$ such that $L \ll N$. Then the query embedding is $O(L)$.

In this chapter, we explore the use of metric space indexing for efficient and approximate query matching. In particular, our contributions are,

- We formulate the query matching and deduplication problems in ER to provide metric space based solutions. First, we propose an indexing approach based on landmarks as a motivating example to explore the basic building blocks needed for query matching.

- We propose a landmarks based indexing technique for query matching to provide a quick and accurate block of potential matches. Our method can process a stream of queries against a large-scale data set within a fixed short time. By doing so, we obtain as many of the matching records as possible where the processing time of a single query takes a sub-second time. The technique is robust to noisy data that contains errors and allow efficient approximate query matching.

## 5.2 Related Work

Indexing is a crucial component for improving the efficiency and the scalability of ER solutions. A detailed discussion of indexing and some well-known existing indexing methods are given in Chapter 2. Here, we only survey a few closely related works since many of the existing methods are orthogonal to the focus of this chapter.

Canopy clustering uses cheap comparison metrics to group similar records into overlapping canopies and creates blocks from records that share a common-canopy [113]. The method depends on global threshold values, and that reduces its flexibility. It also uses similarity measures such as TF-IDF and Cosine distance that can be computationally expensive [184].

The work by Ramadan *et al.* [60] on real-time indexing for ER proposed three dynamic indexing techniques and a blocking key learning algorithm suitable for real-time ER. The proposed DySimII method uses a similarity-aware inverted index that updates whenever a new query record arrives. The other two methods (DySNI and F-DySNI) use tree-based indexing structures with sorted neighbourhood indexing that is tailored for real-time ER. In our work, we are proposing mapping-based indexing techniques suitable for real-time ER.

Mapping-based indexing methods map records to objects in a Euclidean space, preserving the original distances between them. Jin *et al.* [16] proposed the StringMap algorithm that maps records into a similarity preserving Euclidean space (with dimension between 15-20). Similar pairs are determined by building an R-tree [149]. StringMap has linear complexity, but it requires tuning several parameters. Moreover, the performance

of such approaches tends to decrease with more than 20 dimensions. In contrast, our method operates in much smaller dimensions.

The Double Embedding scheme [15] uses two-dimensions, $K$ and $K'$ for embedding records such that $K' < K$. Similarity joins are performed in the metric space using a Kd-tree and k-NN search to find candidate matches. The method is faster than the Stringmap algorithm. However, it attempts to keep the embedding contractive by increasing the distance computations and is not suitable for large-scale data.

Another metric space indexing technique utilised an M-tree to produce complete and efficient ER results. The cost of the method and the quality of the results have remained similar to existing indexing techniques [188]. However, it does not scale for large-scale data since it has a single step that combines the indexing, comparison, and classification steps.

Sehili *et al.* [189] has proposed a pivot-based indexing method for privacy-preserving ER. This method uses bloom filters to encode entity records into bit-vectors and applies metric space similarity measures for similarity search and ER. The authors have used the M-tree indexing structure to organize the bit vectors in the metric space and facilitate a fast similarity search. While the method is highly effective for privacy-preserving ER, the running time of the method increases quadratically with the data size. The basic idea behind this method and the proposed Em-K indexing is similar, e.g., match entity records in a metric space. However, we use MDS to convert entity records into numerical vectors instead of converting them to bit vectors. Moreover, we focus on query matching in ER instead of privacy-preserving ER, which gives us more flexibility to choose any embedding method that works with our data.

In summary, many existing mapping-based indexing techniques have two components. The first is to map the records into a metric space. The second is to perform similarity joins in the metric space using a tree-like indexing structure.

These methods were developed offline by applying a mapping technique to map all the records into a multidimensional metric space. The spatial mapping of records acts as a filtering step before the actual record matching. Hence, the focus is on matching similar records without grouping them into blocks. However, none of them except the [189] handles new records by reusing the properties of an existing multidimensional metric space. Therefore, query matching is not achievable. In this chapter, similar ideas are significantly extended to accommodate the benefits of the Euclidean space for efficient query matching.

## 5.3 Problem Formulation

In this section, we formulate our problem using the concepts and the definitions presented in the previous section. The proposed Em-K indexing method functions as a preprocessing approach for a more detailed query matching ER. We use deduplication as a motivating

example that explains the basic building blocks of the query matching problem. Thus we defined the following two problems:

**Query Matching**: is the problem of finding similar records given two entity collections, $E_r$ a reference database, and $Q$ a stream of queries. The size of each dataset is denoted by $|E_r|$ and $|Q|$ respectively and we assume $|E_r|$ is fixed and $|Q| \to \infty$. Let $Q = q_1, q_2, .., q_{|Q|}$, where each query $q_i$ in $Q$ represents a record of an entity $e_j$. A query record $q_i$ has the same attribute schema as the records in $E_r$. Hence for each query $q_i$ in $Q$, the records in $E_r$ that belong to the same entity ($e_j$) need to be found. In real-life problems, the query rate might be very fast. There may or may not be a matching record for every $q_i \in Q$ that belongs to the entity $e_j$.

**Definition 4** *(Indexing for Query matching): For two datasets (one is a reference database and the other is a stream of queries) with overlapping records, run the best algorithm for a given amount of time to find the block of records containing as many matches as possible for the querying record. This is a pre-processing step to increase the efficiency of the subsequent detailed query match.*

We also consider the deduplication problem here, primarily as an explanatory model.

**Deduplication**: For a given entity collection $E_1$, that contains entities $e_1, e_2, ..., e_n$, finds all the records $r_i$ that represent each entity $e_i \in E_1$. We assume $|E_1|$, is fixed.

**Definition 5** *(Indexing for deduplication): For a given entity collection containing duplicates, group similar records into blocks to reduce the number of comparisons needed in subsequent detailed deduplication while missing as few matches as possible.*

Traditional indexing techniques split the database into non-overlapping blocks, only comparing the records within any block [68, 113, 184, 190, 191]. Blocks of similar pairs are determined by building an indexing structure that takes a set of records as input and classifies them according to some criteria. Usually, this criterion is based on matching a *blocking key* consisting of a single or several attribute values of records [184]. Our method uses *blocking values* to create blocks of records that transform blocking into a k-NN search. We combine the spatial joins with block building to convert blocking values to a similarity preserving Euclidean space. The result is overlapping blocks of records.

The proposed method requires mapping blocking values into multidimensional vectors. Since many comparisons in ER are between strings of characters, we focus on entity attributes that contain string values here. Unlike the similarity between string values, the similarity between numerical values is easy to compute using $L^p$ norms in a metric space [64]. Hence, we used this property of the metric space to propose a scalable indexing technique for query matching.

If we assume strings as elements of a complicated high-dimensional space, the distance between two different strings is typically large. However, misspelled strings tend to be

located near correctly spelled strings [192]. The embedding of a string database into a metric space needs to preserve these two properties. Thus, coordinates for blocking values are determined in a configuration space such that the associated Euclidean distances approximate the dissimilarities between the original blocking values. Following the name approximation proposed in Chapter 4, we use LSMDS for the embedding.

We can map a set of blocking values to a lower-dimensional configuration space by applying LSMDS such that the distances between vectors preserve the dissimilarities between them. This embedding leaves similar blocking values closer in the configuration space allowing efficient, geometric-based indexing.

We propose an OSE technique for LSMDS to avoid the computational overhead of applying LSMDS to large-scale data and to map new data. Refer to Chapter 3, Section 3.1 for more details. Hence, we are able:

1. To embed large-scale reference databases.

2. To embed previously unseen data to a pre-mapped configuration space.

We scale our OSE solution to accommodate large-scale data by only considering a fraction of the existing configuration space, defined as *landmarks*. Refer to Chapter 3, Subsection 3.2.2 for relevant definitions and details. We discuss the characteristics of a good set of landmarks and landmark selection methods in Section 5.5 of this chapter.

Once the configuration space includes all the required data, we formulate our indexing method to generate blocks that group close-by vectors in the configuration space. Searching for similar points in configuration space is less expensive and quick since we can use efficient data structures such as Kd-trees. We use Kd-trees and k-NN search to find similar vectors in the configuration space (definitions are given in Chapter 3, Section 3.3).

An efficient and scalable indexing method can facilitate accurate and efficient k-NN search that supports large-scale datasets. Here we are interested in finding the $k$ nearest neighbours (k-NNs) where $k$ may be moderately large. Searching a Kd-tree for k-NNs is $O(k \log N)$, which is the key to fast indexing.

Table 5.1: Summary of the main notations used in this chapter.

| | |
|---|---|
| $N$ | Input data points to a MDS algorithm |
| $\Delta$ | Dissimilarity matrix of objects $N$ |
| $X$ | Coordinate matrix of objects $N$ in the configuration space |
| $D$ | Distance matrix for $X$ in Euclidean space |
| $\delta$ | Dissimilarity or distance measure that calculate the distance between objects $N$ |
| $d$ | The Euclidean distance between coordinates $X$ |
| $K$ | Output dimension of a MDS configuration |
| $\sigma$ | Stress of a MDS configuration |
| $E$ | Collection of Entities |
| $Q$ | Stream of queries |
| $y$ | Out-of-sample object |
| $\hat{y}$ | Mapping coordinates of the out-of-sample object $y$ in the configuration space |
| $L$ | Number of landmarks in the set $L_s$ |
| $k$ | Number of nearest neighbours in a k-nearest neighbour search |
| $B$ | The size of the block return by the Em-K indexing method |

## 5.4 Methods of Em-K Indexing

We first summarise the terminology and the notation used in this chapter in Table 5.1.

### 5.4.1 Indexing for Deduplication

Deduplication refers to identifying matching records within a single database and has many applications in database and business contexts. For instance, many businesses maintain databases of customer information that are utilised for advertising purposes, e.g., emailing flyers. Duplicate entries might arise because of errors in data entry or address changes. Deduplication techniques are useful to remove duplicate entries and to improve the quality of the collected information. Duplicate-free customer information databases prevent emailing several copies of flyers to the same customer, which reduces the cost of advertising, but there are many other benefits.

Indexing is a preprocessing step to avoid the need to perform detail-level comparisons between $O(N^2)$ pairs of records. The proposed indexing method utilises the properties of vectors and Kd-trees in a configuration space, specifically a Euclidean space. Hence we will be using both terms synonymously in this chapter. The method has two main steps,

1. Embedding the blocking values: We select the blocking criteria based on the attributes of the records in a dataset. For instance, given a set of records with entity

identifying attributes such as first name, last name, date of birth, or postcode, we may choose one or several values of them in our indexing method. We embed the blocking values of a dataset/database into the configuration space by applying LSMDS. The embedding depends on the size of the dataset. We propose two techniques:

(a) *Complete LSMDS*: For a given dataset of size $N$, apply LSMDS for the blocking values of all the records.

(b) *Landmark LSMDS*: Apply complete LSMDS only to a fraction of the dataset (the landmark records). Then the remaining records are embedded applying OSE against the landmark points. We explained this approach in Chapter 3, Subsection 3.2.2.

2. Nearest neighbour search: Searching for similar points is a two-step procedure.

   (a) The first step is to build the Kd-tree in the configuration space using all the points that represent the blocking values in this space.

   (b) The second step is to create blocks of similar points by searching the k-NNs of the Kd-tree nodes.

   Since the Kd-tree construction uses all the available points in the configuration space, each record becomes a node in the Kd-tree. Likewise, each node becomes a query against the rest of the nodes in the k-NN search. Each node has a fixed number of nearest neighbours (NNs) allocated for them as we keep the k-NN search fixed for every querying node. Hence, each node in the tree becomes a small block of records that contains its NNs as the members. The block sizes are uniform and depend on the number of NNs allocated for a node.

Once all the blocks are determined by the indexing method, we return the pairs of similar points in each block as potential matching records. Then, we retrieve the original blocking values that correspond to these points and compare them to classify the pairs as candidate matches or non-matches based on a pre-selected threshold.

A detailed level comparison among the other attribute values will be only required between the pairs that indexing identifies as candidate matches. Thus our indexing method act as a filtering step that reduces the total number of detailed comparisons one has to perform when identifying similar records in a dataset.

### Complexity

We can quantify the complexity of the proposed indexing method. The method has two components: a relatively slow step where the records (blocking values) are mapped

to a configuration space, followed by a relatively fast step that creates blocks in the configuration space.

Assume that we have $N$ records in the underlying database. Complete LSMDS requires calculating the distance between all pairs of blocking values, hence, $O(N^2)$ operations for the embedding. This complexity dominates the LSMDS calculations. However, for large $N$, we can choose a set of landmarks $L$ and apply LSMDS with a complexity of $O(L^2)$. The embedding of the rest of the points has a linear complexity of $O(ML)$ operations, where $M = N - L$. Hence the overall complexity of landmark LSMDS is $O(L^2 + ML)$.

The second phase builds and searches the Kd-tree to create blocks of records. Building the Kd-tree requires $O(N \log N)$ operations, and searching for $k$ nearest neighbours for $N$ points requires $O(Nk \log N)$ operations, where the size of a block ($B$) is equal to $k$. Hence the overall complexity of the indexing step is $O((1 + k)N \log N)$.

Since a complete LSMDS requires $O(N^2)$ operations, we recommend using the landmark LSMDS and as small as possible $k$ to reduce the complexity of the proposed method.

### 5.4.2 Indexing for Query Matching

Query matching in this chapter refers to querying a stream of records against a reference database to find records that refer to the same entity as the query (see Section 5.3). Query processing should be quick and accurate for many ER solutions to increase their usability in real-time applications. In some applications, a stream of queries might need to be processed within a fixed time, collecting as many matches as possible. In such settings, we have to trade accuracy against speed when detecting matching records. This section presents a scalable indexing method for real-time, approximate query matching against a large-scale database. The ideas presented in Subsection 5.4.1 serve as the basic building blocks for the proposed method.

Following the *clean-clean ER* scenario, we consider a large-scale reference database $E_r$ and a batch of streaming queries $Q$. Similar to the previous indexing method, we first embed all the blocking values of the reference database in a Euclidean space. We use a set of landmarks and the OSE of LSMDS to reduce the overhead of embedding the database $E_r$. The embedding is a two-step procedure: 1) apply LSMDS over the landmarks 2) then map the rest of the blocking values using OSE of LSMDS based on the distances to these landmarks. We then construct a Kd-tree using all the points in the configuration space, where each data point becomes a node in the tree.

For streaming queries, we process a single query record at a time. First, we embed the blocking value of the query record in the existing configuration space. In general, OSE would require calculating all the distances from the new query to the pre-mapped blocking values in the original string space, which is not desirable with large-scale databases. Hence, we only calculate the distances to the landmarks when mapping a new record. The mapping position is determined by applying the OSE of LSMDS to the new query

record. The process is similar to mapping the blocking values that are not landmarks in the reference database. Figure 5.1 shows the steps of our Em-k indexing method.

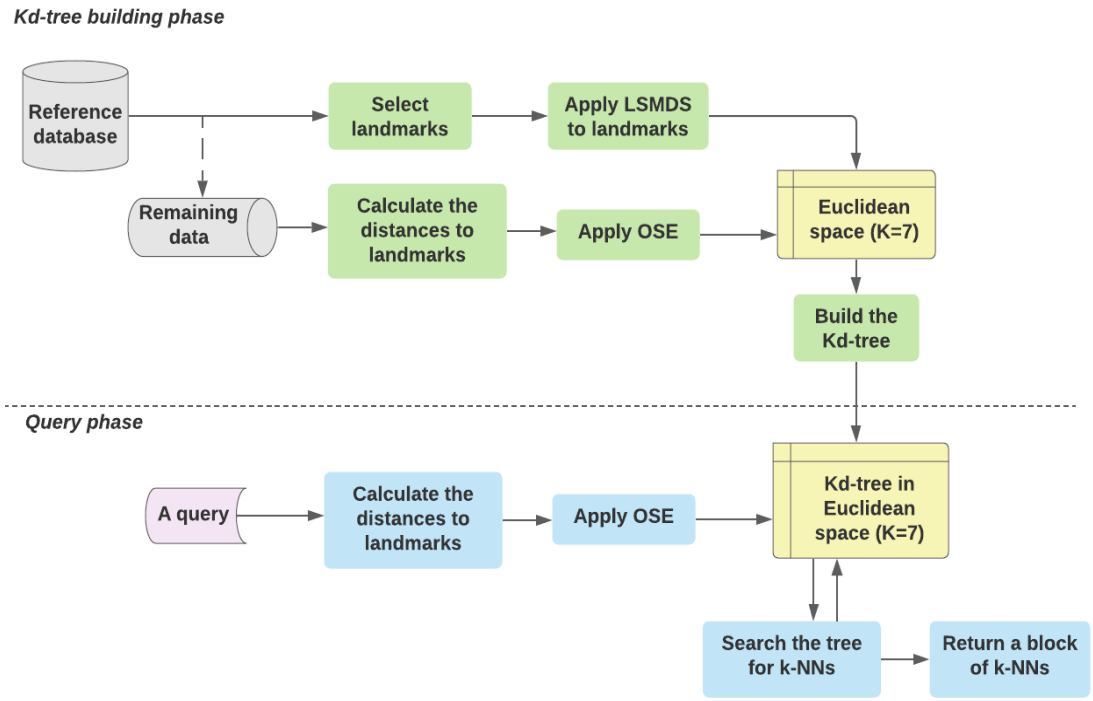



Figure 5.1: The process of our Em-K indexing method. In the Kd-tree building phase, the reference database is embedded into a configuration space by applying landmark LSMDS. Once all the points are embedded, a Kd-tree is built using all the existing nodes. In the query phase, each query needs to be embedded in the existing configuration space using OSE. Then we search for k-nearest neighbours of a querying record in the Kd-tree. The final output is a set of similar points to the query.

The only inputs we need for the mapping are the distances from a query to the pre-selected landmark blocking values. Then we search for the k-NNs of the new point using the existing Kd-tree structure of the reference database. A new block of points that contains similar points for the query point will be determined. The block size $B$ depends on the $k$ of the k-NN search. We retrieve the original blocking values for the block of similar points and use a pre-selected threshold to filter out the potential matching pairs. The records that contain these as blocking values are the matching records to the querying record. A detailed level comparison between the candidate matching pairs may be required after the initial step of indexing.

The algorithm needs to process a query within a fixed time, potentially sub-second.

Since we are querying a stream of query records against a large-scale database, the processing time of a single query is limited. The overall system performance can be improved by trading accuracy against efficiency and scalability. We will fine-tune our method to trade-off many comparisons against accuracy to offer a greater number of detections within a fixed time. Hence, we select a set of optimal parameters that scale our data by considering the trade-off between accuracy and scalability.

### Complexity

We can quantify the complexity of different stages in the Em-K indexing for query matching. Similar to the previous method, it contains two components: a relatively slow step where a new query is mapped to an existing configuration space, followed by a relatively fast step that creates a block of similar points which contains candidate matches for the query. We assume below that the reference database has already been embedded since this cost is amortised across many queries.

Suppose that we have $N$ data points in our reference database. Typically it would require $O(N)$ operations to compare the existing records with a query point, which is not feasible for larger $N$. We avoid this complexity by applying OSE of LSMDS using a set of landmarks $L$, which requires only $O(L)$ operations to embed a new data point. The embedding is efficient if $L$ is chosen such that $L \ll N$.

In the second phase, we search for $k$ nearest neighbours for the query. Therefore, the block size $B$ is equal to $k$. It will cost $O(k \log N)$ operations to search the tree and $O(k)$ operations to compare a new query point within a block. The total cost of indexing for a query is $O(L + k \log N)$. However, $k \ll L \ll N$ in large-scale applications. The cost of the k-NN search is insignificant due to the efficient indexing structure of the Kd-tree. Thus OSE step dominates the complexity of the proposed method.

The number of landmarks $L$ will affect both the accuracy and scalability of this method. We will discuss the selection of landmarks and results in the next section.

## 5.5 Experimental Validation

We evaluated the two proposed Em-K indexing methods under various settings (e.g., different dimensions, varying block sizes and datasets, and error rates).

Two main questions to study in the experiments are: (1) Are these proposed methods robust over various settings? (2) Does Em-K indexing achieve high accuracy and good scalability?

All algorithms are implemented in R and executed on a desktop with Intel Core 5 Quad 2.3GHz, 16GB RAM, and MacOS Big Sur.

### 5.5.1 Set Up

#### Data Sets

We examined the performance of our methods over two synthetic datasets. They can be manipulated to have significant variations in their size and characteristics (e.g., error rates). In this work, we used the Levenshtein (LV) distance (defined in Chapter 2, Subsection 2.5.3) to measure the similarity at the attribute level of entities. We do not use the name-like vectors approach described in Chapter 4 since we need to test realistic embeddings. Instead, we use,

- ***Dataset-1:*** The first data set contains records with synthetically generated biographic information. Each record has a given name and a surname. They are generated using the tool Geco [164]. We introduced duplicate records with errors by slightly modifying the values of randomly selected entries. In record generation, we assumed a record has only one duplicate with a maximum of two typographical errors (substitutions, deletions, insertions, and transpositions) in both attribute values.

  For the deduplication datasets, there is one duplicate for a particular record within the dataset. Similarly, in query matching, each query has one duplicate within the reference database, and the reference database is duplicate free.

- ***Dataset-2:*** The second dataset is the benchmark dataset presented in [193]. It is based on personal records from the North Carolina voter registry and synthetically generated duplicates (using Geco [164]). Each record has several attribute fields. We cannot control the errors of the duplicate records in this dataset since it has been formulated as a benchmark dataset. However, after careful analysis of the dataset, we estimated that a duplicate record has a maximum of three edit distance errors for this work. We have only considered the first name and the last name fields in our experiments.

  Similarly, we selected the deduplication data such that there is only one duplicate for a particular record within the dataset. In the query matching, we choose the queries to have only one duplicate within the reference database and the reference database to be duplicate free.

In this chapter, we mainly used 5000 randomly selected entity records as the reference database and flexible entity query sizes for testing from the above datasets, which allowed us to perform resampling to evaluate the results for consistency. However, we show that our methods are scalable to large-scale datasets.

### Matching Rates

We control the number of duplicates in our experiments in order to understand how well the method works in different circumstances. We define two matching rates: Consider the datasets $E$, $E_r$ and $Q$ with sizes $|E|$, $|E_r|$ and $|Q|$,

1. *Deduplication matching rate (DMR)*: the matching rate for deduplication is defined to be the $DMR(E) = \frac{E_d}{|E|}$ where $E_d$ is the number of duplicate records in $E$.

2. *Query matching rate (QMR)*: the matching rate for query matching of $Q$ against a reference database $E_r$ is defined to be the $QMR = \frac{M_{RQ}}{|Q|}$ where $M_{RQ}$ is the number of records in $E_r$ that has a matching record (duplicate) in $Q$.

We control the number of duplicates within and between datasets by changing the $DMR$ and $QMR$. We represent $DMR$ and $QMR$ as percentages in our experiments.

### Performance Evaluation Metrics

We use two measures to quantify the efficiency and the quality of our indexing method proposed for deduplication [194].

- Reduction Ratio (RR): Measures the relative reduction of the comparison space, given by $RR = 1 - \frac{N_b}{|E|(|E|-1)}$ where $N_b$ is the number of potential matching pairs produced by an indexing algorithm. This quantifies how useful the indexing is at reducing the search space for detailed comparisons.

- Pair Completeness (PC): This is given by $PC = \frac{N_m}{M}$, where $N_m$ denotes the detected number of real matches by the indexing algorithm and $M$ represents the number of all real matches in $E$. This is a measure of how accurate the indexing is.

Both PC and RR are defined in the interval $[0, 1]$, with higher values indicating higher recall and efficiency, respectively. However, PC and RR have a trade-off: more comparisons (higher $N_b$) allow high PC but reduce the RR. Therefore, indexing techniques are successful when they achieve a fair balance between PC and RR. However, choosing which measure to weigh depends on the application and a user's decision. If the application is more concerned with the quality of the matching results, then we want to achieve a high PC. Similarly, for applications more concerned with scalability, high RR is preferred.

We define two measures to evaluate the performance of our indexing method of query matching. These are different from the standard measures of indexing defined above as we combine indexing with query matching here. Hence, in this context, we are interested in measuring the efficiency of the method in terms of time and speed of processing a query.

The Em-K indexing method returns a block of records that contains both true positive (TP) and false positive (FP) matches per query as a final result. Hence the following measures are used.

- Number of true positives per computational effort: Measures the number of true matching records determined by the indexing method when processing queries within a set period.
- Precision: Measures the accuracy of the query matching in terms of precision. The precision $P$ is denoted by $P = \frac{|TP|}{|TP|+|FP|}$, where $|TP|$ is the number of true positives and $|FP|$ is the number of false positives.

All the CPU running times are measured in *seconds* and denoted by RT.

### 5.5.2 Choice of Parameters

Several factors may impact the performance of the proposed methods; some of them (e.g., dimension ($K$), block size ($B$), landmarks ($L$) ) are control parameters that we rely on to fine-tune the performance, while others (e.g., dataset size or error rate) are parameters determined by data sets.

In this section, We discuss the choices of the $K$, $B$, $L$ parameters in our indexing implementations (and their rationale). The robustness of the methods is measured with respect to the varying sizes and matching rates of the data sets.

**K**: The dimension of the Euclidean space ($K$) plays an important role in the performance of our indexing methods. We applied LSMDS to a sample of 5000 records selected from Dataset-1. The dissimilarities between records are measured using LV distance. In Figure 5.2, the first y-axis shows the stress ($\sigma$ defined in Equation 3.2) decreases as $K$ increases.

A good value of $K$ should differentiate similar objects from dissimilar ones by approximating the original distances. If $K$ is too small, we will have high-stress values where dissimilar pairs will not fall far enough from each other. It could also place dissimilar pairs closer and similar pairs further apart. Conversely, high $K$ values will have low-stress values. However, in terms of RT, higher dimensions increase the embedding time. In Figure 5.2, the second y-axis represents embedding RT. It takes more than 30 minutes to embed the dataset in 19 or 20 dimensions.

Considering the trade-off between the *stress vs dimension* and the *embedding time vs dimension*, a reasonable value of $\sigma$ is found around 6-8 dimensions. This is consistent with the embedding name strings in the Euclidean space, as discussed in detail in Chapter 4. We will use $K = 7$ here.

**B**: Block size is a dominating factor that directly affects the effectiveness and efficiency of many indexing techniques. Large block sizes increase RT in the indexing step and have

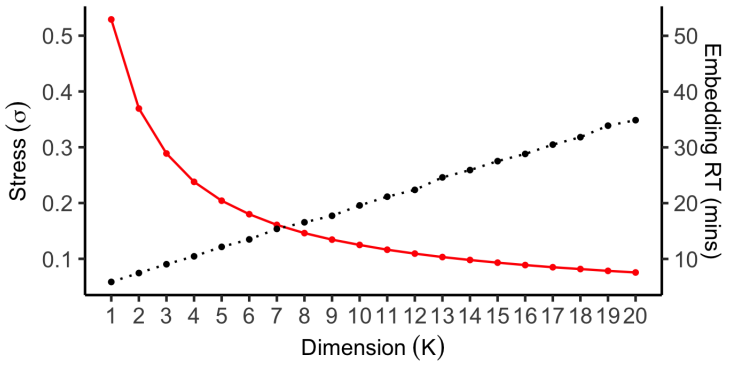



Figure 5.2: The trade-off between the dimension ($K$) vs stress ($\sigma$) and embedding time. The first y-axis represents $\sigma$, while the second y-axis represents the embedding time. The $\sigma$ tends towards a small but non-zero asymptote when $K$ increases. The running time increases linearly when K increases. Higher dimensions allow lower $\sigma$ values for the embedding but increase the embedding time, for marginal benefit.

low RR and high PC values. In contrast, small block sizes lead to high RR values with fewer comparisons within each block. However, this may result in low PC values due to missing some matches. Blocks of similar points are determined by k-NN search in Em-K indexing methods. Hence, $B$ is equal to $k$ (number of nearest neighbours). We will consider the choice of $B$ in detail in the following sections.

**L**: The number of landmarks ($L$) is another important factor in our indexing methods. Landmarks are utilised for two different purposes in this work. First, we use landmarks for embedding large-scale data into a Euclidean space when the standard LSMDS method becomes inefficient. Second, we use landmarks to map the out-of-sample queries in the existing configuration space. We discuss the role of landmarks in deduplication and their impact on the proposed indexing method in the following sections.

In the following experiments we investigated how $B$ and $L$ impact our indexing algorithms.

### Varying Block Sizes ($B$)

To test the choice of $B$, we used sample datasets containing 5000 records each from the two data sources. We set the $DMR = 10\%$ for the data selected from Dataset-1, which means there are 500 duplicates within the selected 5000 records in the sample dataset. Similarly, we used $DMR = 7.5\%$ for the second data sample selected from Dataset-2. Hence within the 5000 records, 375 of them are duplicates.

We performed deduplication indexing on the two datasets. Figure 5.3 illustrates the trade-off between PC and RR of our indexing method for different block sizes using the

Dataset-1. In each instance, we changed $B$ by varying $k$ in the k-NN search. PC increases with the increase of $B$. In contrast, RR decreases due to the increment in the number of comparisons within each block of records.

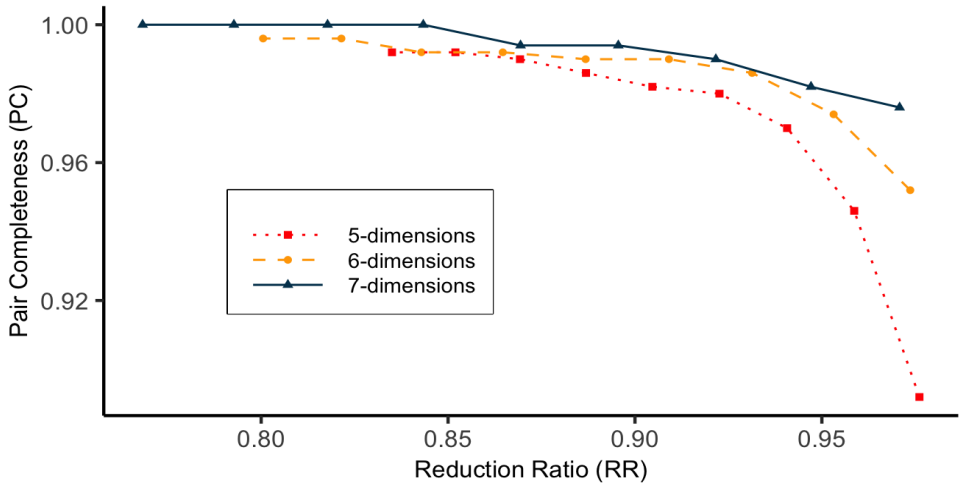



Figure 5.3: The trade-off between the reduction ratio (RR) and pair completeness (PC) in different dimensions. Ideally, RR=PC=1, hence we prefer methods whose results lie as close to the top-right corner of the graph. The block sizes are 100, 90, 80, 70, 60, 50, 40, 30, 20. Large blocks achieve higher PC but lower RR. The three curves illustrate that the higher the dimension, the better the results, up to the point of diminishing returns.

The results also indicate that dimensions around 7 are good at shifting the PC-RR curve to the top-right corner delivering a good ratio between RR and PC. However, higher dimensions also mean higher computational costs (e.g., high RT) for the embedding. Based on Figure 5.2 and Figure 5.3, therefore, we conclude that using $K = 7$ and $B = \{50, 60\}$ gives the best compromise PC-RR ratio and RT overall for the given data set. In subsequent experiments, we used $K = 7$.

In most comparisons, we observed similar results for both datasets and discussed only the results of Dataset-1. However, we present one comparison that has comparably different results here. Figure 5.4 compares the two datasets in a fixed dimension ($K = 7$), varying $B$ in each instance. Both data sets achieve similar RR values with different PC values. PC values are comparatively low for Dataset-2 and are around 70% for most occurrences.

Dataset 1 shows better results compared to Dataset-2 in Figure 5.4. This behaviour is expected since the two datasets have different characteristics, e.g., different matching rates and a different number of errors in each field. Furthermore, we used different pre-selected thresholds ($\theta_m$) when validating candidate matching pairs in the two datasets. These thresholds are selected based on the errors in the two datasets. In Dataset-1, duplicates have a maximum of two typographical errors and therefore $\theta_m = 2$. For Dataset-2, we

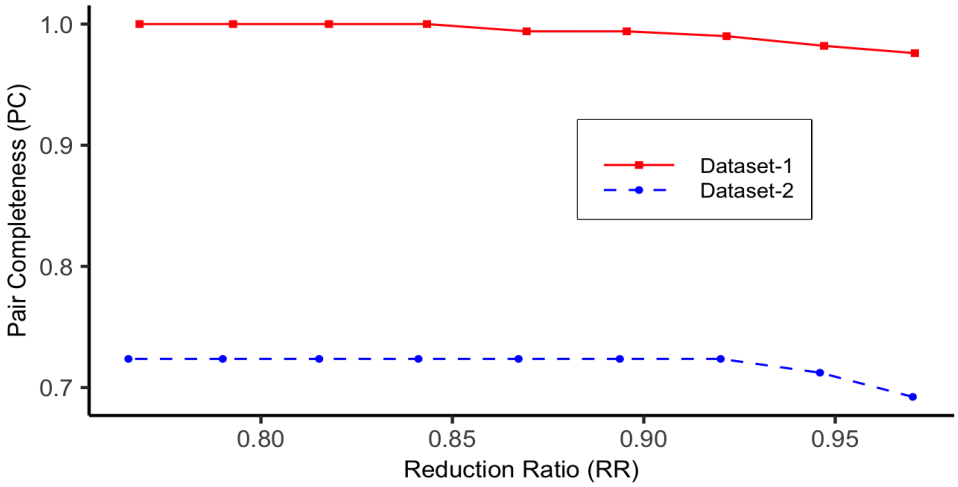



Figure 5.4: The trade-off between the reduction ratio (RR) and pair completeness (PC) for two different datasets for complete Em-K indexing. The block sizes are 100, 90, 80, 70, 60, 50, 40, 30, 20. PC is very low for the second dataset while RR values are closer.

estimated that each duplicate record has a maximum of three typographical errors, and we set $\theta_m = 3$ therein.

### The Effect of Landmarks

The following experiment investigated the effect of the two different embedding techniques on the proposed indexing method. Performance is measured using the PC and RR curves. First, we applied complete LSMDS similar to Figure 5.2, keeping the $K = 7$ fixed. Second, we applied LSMDS only to a set of landmarks in the same dimension. The remaining points are embedded using the OSE of LSMDS and the distances to landmarks. Blocksize $B$ is varied as before.

Figure 5.5 compares the trade-off between PC and RR for complete LSMDS and landmark LSMDS (for different $L$). Each instance represents a different block size. PC and RR change similarly to the previous experiment for different block sizes. However, Figure 5.5 suggests that we can get similar results by choosing an approximate embedding that uses landmarks instead of complete LSMDS. The use of landmarks decreases the distance calculations, and we can avoid the inherent complexity and inefficiency of LSMDS when processing large-scale data.

Using our indexing solution, we can solve deduplication applications in ER. The method requires embedding records into a configuration space in order to apply the indexing technique. Since complete LSMDS is not suitable for large-scale data, we recommend using the landmark LSMDS. We applied the farthest point sampling (FPS) [140] for reproducible results in landmarks selection; however, random selection works well in prac-

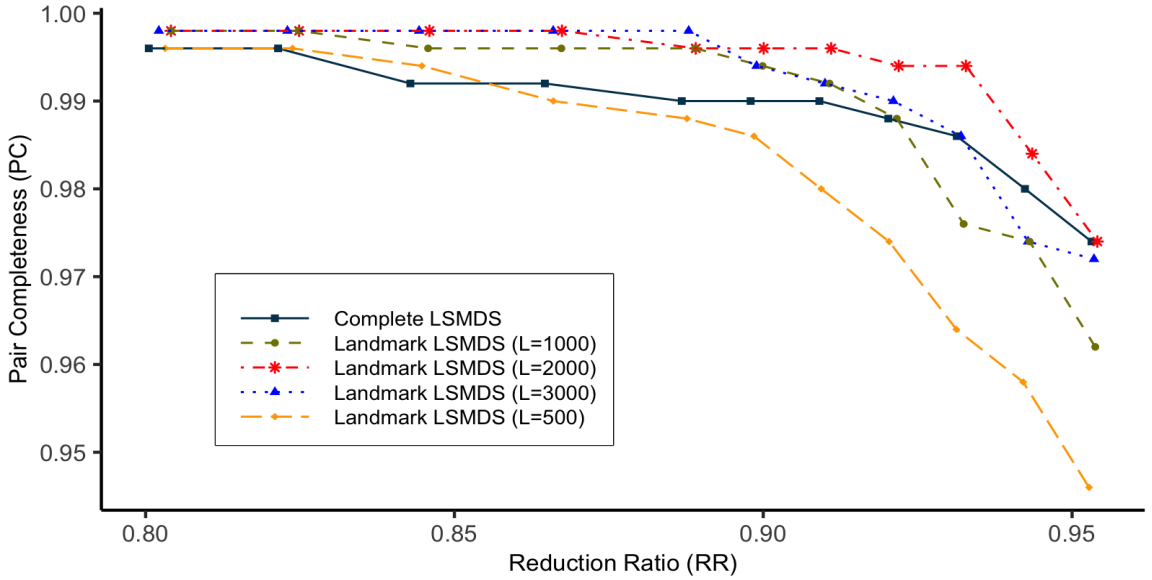



Figure 5.5: The trade-off between the reduction ratio (RR) and pair completeness (PC) for complete Em-K indexing based on different embeddings.The results are based on a complete LSMDS and landmark based LSMDS for different number of landmarks. The block sizes are 100, 90, 80, 70, 60, 55, 50, 45, 40, 35, 30.

tice [130]. The optimal parameter setting for our data is $K = 7$, $B = 50$ and $L = 1500$. We used the proposed indexing method of deduplication as a motivating example for the next set of experiments.

### 5.5.3 Comparison with StringMap

The StringMap algorithm proposed by Li *et al.* [16] is one of the popular mapping-based indexing techniques. It has been frequently used as the base method when developing other mapping-based techniques in ER. Hence, we compare our method with the StringMap algorithm as follows.

The underlying concept of LSMDS and StringMap is to map strings to a multidimensional Euclidean space that preserves domain-specific similarity. Refer to Section 5.2 for more details. First, we measured the accuracy of the embedding created by the above mapping techniques. We considered the complete LSMDS, Landmark LSMDS, and StringMap in the comparison. The accuracy is measured in terms of *stress*.

We selected the same deduplication dataset of 5000 name strings and applied each of the mapping techniques to embed them in different dimensions. We then calculated the stress values for each setting. For Landmark LSMDS, we have chosen $L = 1500$ following the previous experimental results.

Figure 5.6 shows the trade-off between stress vs dimension for three methods. Com-

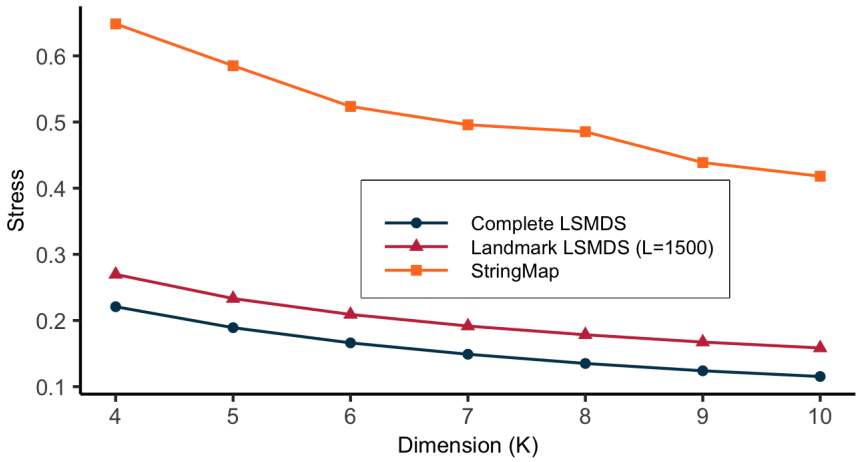



Figure 5.6: The trade-off between the dimension ($K$) vs stress ($\sigma$). The $\sigma$ tends towards a small but non-zero asymptote when $K$ increases for all the methods. Complete LSMDS and Landmark LSMDS ($L = 1500$) has a low stress value in each dimension compared to StringMap method.

plete LSMDS has the lowest stress values, while StringMap has the highest stress values in each dimension. Landmark LSMDS has slightly higher stress values than complete LSMDS. This phenomenon is expected since we are only using 1500 landmarks for the initial embedding. The rest of the strings are embedded using the distances to the landmark points in the configuration space. This further distorts the original distances than the original LSMDS method. However, the results suggested that the LSMDS methods provide more accurate embeddings than the StringMap method.

Next, we applied StringMap and Landmark LSMDS with deduplication to compare the results. However, in the implementation of StringMap based indexing evaluated in the experiments, the originally implemented R-tree data structure has been replaced with a Kd-tree. First, we obtained two different embeddings for our Dataset-1 by applying StringMap and Landmark LSMDS. We kept $K = 7$. Then we build KD-trees for both embeddings to find matching entity records. Similar to the previous experiments, we created blocks for each node in the trees to group similar records. Blocksize varies from 20 to 80. The optimal parameter settings for Landmark LSMDS suggested using a block size around 50-60. Hence we explored the neighbourhood around these block sizes. Similarly, we used $L = 1500$.

Figure 5.7 compares the trade-off between PC and RR for StringMap based indexing and landmark LSMDS based indexing for the same deduplication task. Each instance represents a different block size. PC and RR change similarly to the previous experiments for different block sizes. Landmark LSMDS maintains a high level of PC between 1 and 0.96 for all instances. PC values for StringMap vary between 0.98 and 0.79. Small

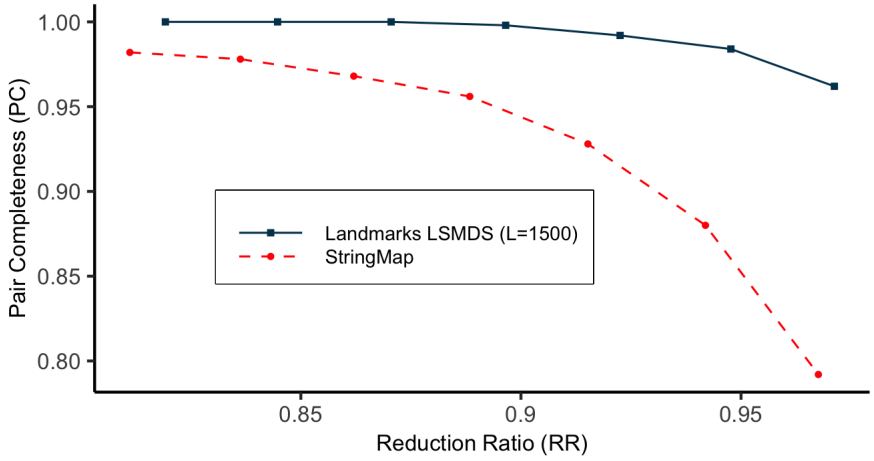



Figure 5.7: The trade-off between the reduction ratio (RR) and pair completeness (PC) for Landmark LSMDS and StringMap based indexing methods. The block sizes are 80, 70, 60, 50 ,40 ,30, 20 and $L = 1500$. Landmark LSMDS has higher PC values compared to StringMap based results.

block sizes reduces PC for both methods. However, PC decreases drastically with small block sizes for the StringMap. Both methods have similar reduction ratio (RR) values for each instance due to similar block sizes. Based on the results, we concluded that the Landmark LSMDS provides a more accurate and efficient indexing solution than the StringMap method for deduplication.

### 5.5.4 Indexing for Query Matching

In our experiments, we used a reference database with $E_r = 5000$ records. The streaming query dataset $Q$ is flexible in size because we only consider streaming queries within a period. Each query in $Q$ has a duplicate record in the reference database, i.e., $QMR = 1$. We made this assumption to keep the experiment more efficient instead of mimicking a real-world ER problem. In a real-world scenario, each query may not have a matching record within the reference database, or the same query may appear in the stream to search against the reference database. However, the time required for searching is the same when no match is present.

Similar to the previous method, several factors affect the performance of the proposed indexing method for querying, e.g., $K, L, B$. Since the embedding of the reference database is the same as before, we keep $K = 7$. The block size is $B$ is equal to $k$, the number of nearest neighbours in k-NN search.

We used landmarks to support the out-of-sample embedding. The number of landmarks, $L$, directly impacts the running time (RT) of the proposed method since each

query needs to be embedded in the Euclidean space. The other costs that contribute to RT are the distance calculations and k-NN search.

First, we embedded the reference database generated from Dataset-1 applying landmark LSMDS. We chose the landmarks based on the FPS. Then, we built a Kd-tree using all the reference data points in the Euclidean space. Once the Kd-tree is built, we passed queries to search the tree for k-NNs. Hence, each query in $Q$ needs to be mapped in the Euclidean space. We used the same set of landmarks among the reference data points to map the query points. Distance calculations are required among a new query point and the landmarks when applying the OSE of LSMDS. A block of similar points is determined for a new query by searching k-NNs in the existing Kd-tree.

Our method process a single query at a time. In the following experiment, we processed 500 queries against the reference database. For each query, we measured the embedding RT and the distance calculation RT separately. Then we calculated the mean values for both categories, processing all the 500 queries. Figure 5.8 shows the comparison of the mean RT of distance calculations and OSE for varying numbers of landmarks. Increasing $L$ linearly increases the embedding RT of a single query. The distance calculation RT also increases linearly with $L$ but is negligible compared to out-of-sample embedding RT.

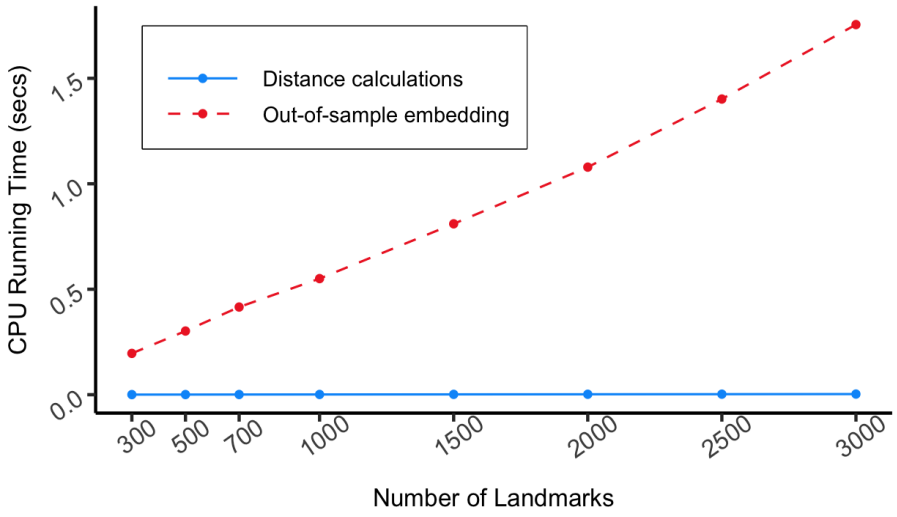



Figure 5.8: The calculation times for distance calculations and out-of-sample embedding for query matching. Both depend on the number of landmarks, but distance calculations are much faster.

We also measured the cost of the k-NN search when creating a block of records for a new query. This search can be done efficiently in the Euclidean space using the Kd-tree and priority queues. It takes less than a millisecond, which is insignificant compared to the total RT of the embedding process. Moreover, increasing $k$ has a smaller impact due

to its efficient implementation with priority queues [159].

A scalable query matching method should be able to process as many queries as possible within a period. In our indexing method, increasing $L$ limits the number of queries processed within a set period. On the other hand, small $L$ tends to decrease the accuracy of the embedding. As a result, a new query may be not mapped closer to its duplicate, reducing the probability of grouping them as similar points. Hence an optimal $L$ is required to maintain the scalability without degrading the quality of the results.

We measured the scalability and the efficiency of our method using two quantitative measures with respect to time: the number of true positive ($|TP|$) matches detected per computational effort and the precision (P) per computational effort. Hence we processed a stream of queries $Q$ against a reference database $E_r$ within a given period, varying the control parameters such as $L$ and $B$ in different instances. Then we calculated $|TP|$ and $|FP|$ found by our method in each instance. The accuracy of the results is measured in terms of precision. It measures the rate of TP against all the positive results (sum of $|TP|$ and $|FP|$) returned by the method within the given period. Subsequently, we determined an optimal set of parameter values for our data that returns the highest TP matches and precision within a fixed period.

In the following experiments, we used a reference database of $E_r = 5000$ records and the stream of query records $Q = 500$. We applied landmark LSMDS to embed the records in $E_r$ to the configuration space in each experiment. Then the queries are processed using the same set of landmarks. Hence, every instance of a different $L$ has a disparate embedding of the reference database, then used for query embedding and searching.

Figure 5.9 shows the comparison of the $|TP|$ against varying the number of landmarks and k-NNs. In each instance, we processed queries within a fixed period (60 seconds). Increasing $L$ decreases $|TP|$ because more landmarks allow fewer queries to be processed. This phenomenon is expected since a large $L$ increases the running time of a single query processing. With more landmarks, there is a higher probability of finding matches for those queries due to the increasing accuracy of the embedding. In contrast, few landmarks allow more queries to be processed within a period since the running time (RT) of embedding a single query is small.

Figure 5.9 suggests that we only need 100 landmarks to detect the highest number of TP matches for the given data. The method has processed all the 500 queries within a minute, detecting 432 TP matches. The average time for processing a single query is 0.07 seconds.

Based on Figure 5.9, we conclude that setting $k = 150$ and $L = 100$ gives the best trade-off between the quality of the results and the RT. This result is consistent with existing real-time query matching techniques that process a query within a sub-second time [195].

To validate the robustness of the method, we compared the results of two scenarios. Hence, we selected two reference databases and two streams of queries derived from

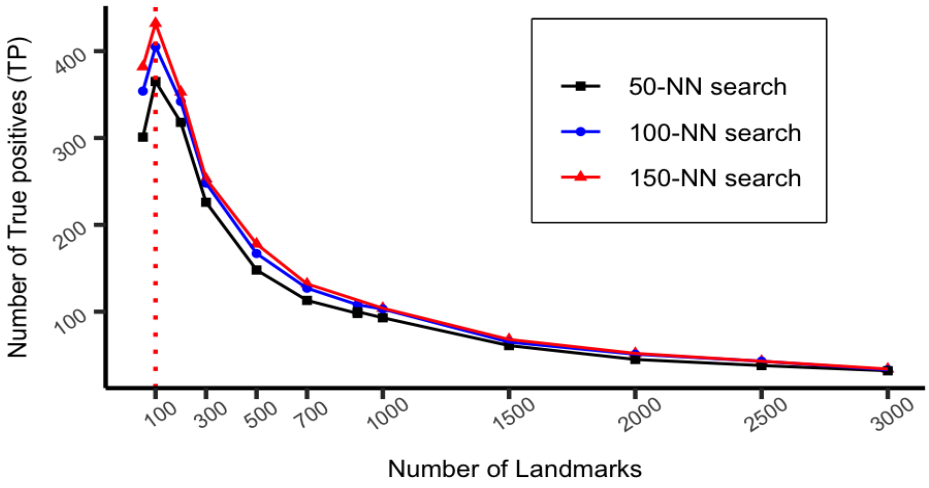


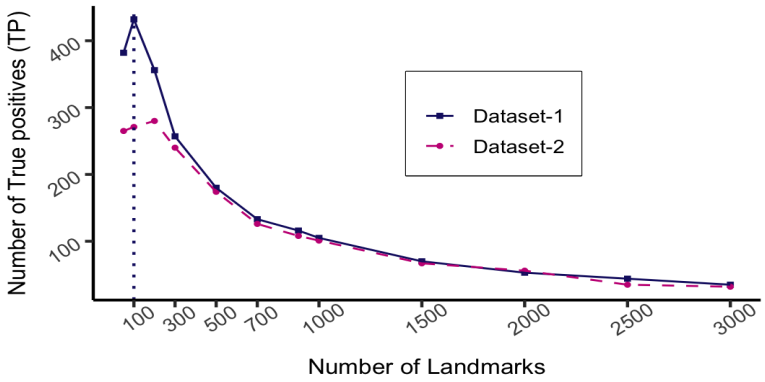

Figure 5.9: The trade-off between the number of true positives and the number of landmarks for three different block sizes. Here the block size is equal to the number of k-NNs. Many landmarks allow fewer queries to be processed, while many k-NNs allow more true match detection.

Dataset-1 and Dataset-2 and applied the Em-K indexing (see Figure 5.10). Both reference databases contained 5000 records. The two streams of queries are processed against the two reference databases separately with similar parameter settings. The query matching rate ($QMR$) is equal to 1 for both. Keeping the control parameters fixed at $K = 7$, $B = 150$ and $T = 60$ seconds, we vary $L$ to compare different results.

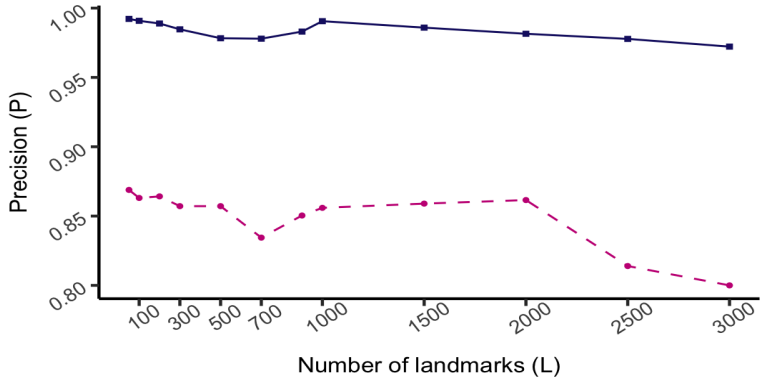
Figure 5.10-(a) compares the trade-off between the $|TP|$ per computational effort, and the number of landmarks. Increasing $L$ decreases the $|TP|$ for query matching in both datasets similar to Figure 5.9. Few landmarks allow processing all the queries within 60 seconds, whereas many landmarks allow processing fewer queries. While both reference databases are similar in size, the total queries processed are different in size. A total of 500 queries are processed against Dataset-1, and 375 queries are processed against Dataset-2 in the experiment. Hence, we observe fewer TP matches for Dataset-2 than Dataset-1 in their highest performance. However, the curves illustrate similar trends for both datasets.

Based on the results produced by the previous experiment, Figure 5.10-(b) compares the two datasets in terms of the precision (P) against varying $L$ In each instance, the queries are processed within T=60 seconds. Hence the P values are measured per computational effort. Dataset-2 has lower P values compared to Dataset-1. This behaviour is expected since we assume that the duplicate records in Dataset-2 contain a maximum of three edit distance errors. The pre-selected thresholds ($\theta_m$) are different for the two datasets. However, we do not have the exact details of error rates for this benchmark dataset. Hence, the method may allow more false positives in the final block of records

retrieved for a given query, reducing the P.




(a) The trade-off between the number of TP and the number of landmarks for two different datasets.




(b) The trade-off between the precision and the number landmarks for two different datasets.

Figure 5.10: The two figures (a) and (b) compares the Dataset-1 and Dataset-2 in terms of number of TP and precision varying the $L$ The k-NNs are fixed to $k = 150$ and $T = 60$ seconds in each instance. The curves illustrate similar trends for similar parameter settings in (a). However, in (c), Dataset-2 exhibits low precision compared to Dataset-1.

However, the overall results suggest that the proposed method is robust over different datasets. The optimal parameter set that provides the best compromise for our data is $K = 7$, $B$ (or $k$)$= 150$ and $L = 100, 300$. We need a few landmarks to achieve the most number of TP matches for a stream of queries within a period. Few landmarks make the embedding efficient. As a result, a single query can be processed within less than a second against a reference database to find matches. Hence our method scales well for approximate query matching against a large-scale reference database for ER.

## 5.6 Discussion

The Em-K indexing methods embed a set of strings in a metric space, particularly a lower-dimensional Euclidean space. There is a trade-off between the dimension and the accuracy of the embedding. Higher dimensions allow the embedding to be more accurate. However, it does not scale well for large datasets. In our experiments, we only selected two blocking variables for embedding and indexing. We can also use other blocking variables, such as addresses and gender. As a result, the dimension of the Euclidean space needs to extend accordingly to facilitate those.

The most costly part of the method is the amount of time to embed queries, which increases linearly with the number of landmarks. However, this approach is easily parallelisable since each query is processed separately. Depending on the application, we can tune the parameters for a fast, less accurate query matching or a slow, more accurate method.

Our method is designed to solve approximate query matching rather than exact query matching. It means we expect the queries to contain errors in their attribute values. We could easily perform an exact search based on lexicographical order considering the query and the reference dataset using a data structure such as a binary tree to find exact matches as a pre-filter for our method.

Our indexing method for query matching uses distance computations to embed the reference database in the Euclidean space. This embedding is a slow process that requires a minimum of $O(L^2)$ operations where $L$ is the number of landmarks. We consider this as the training phase, which only needs to be performed once. However, with the Kd-tree built, we can perform similarity queries with significantly fewer distance computations than a sequential scan of the entire dataset.

## 5.7 Summary

Indexing techniques reduce the pairwise comparisons in ER solutions. Many existing mapping-based indexing techniques work in offline mode with fixed-size databases. Hence, these techniques are not suitable for applications that require real-time query matching, especially if it involves big, fast, or streaming data. Our method investigated the query matching problem in ER by using spatial mapping of records into a multidimensional Euclidean space. The use of vectors in the Euclidean space that represent records allowed fast query matching with comparable accuracy. The proposed method proved fast running time as well as scalability along with the data size.

# Chapter 6

# High Performance Out-of-sample Embedding Techniques for Multidimensional Scaling

In previous chapters, we described MDS as an embedding technique for large-scale ER. In Chapter 5, we proposed an out-of-sample embedding (OSE) technique based on an optimisation that extends the LSMDS algorithm. In this chapter, we investigate the OSE problem in detail to propose a machine-learning approach based on an artificial neural network.

## 6.1 Introduction

Dissimilarities between entities are a significant component of ER, and MDS can analyse any data that represent how similar (or dissimilar) objects or events are to one another. In our previous chapters, we applied MDS algorithms to propose metric space-based solutions for large-scale ER.

Many embedding techniques, including LSMDS, do not provide a fixed transformation for the training data initially used to construct the configuration space. Hence, computing the output transformation of new test data is not trivial. The problem of mapping previously unseen data in an existing configuration is called the *out-of-sample embedding problem* [133].

MDS is a popular embedding technique, but it can be inefficient for large-scale data. Traditional MDS algorithms such as CMDS and LSMDS are computationally expensive. We have discussed their drawbacks in Chapter 3, Subsection 3.2.1. The expense of calculation means that it is impractical to recompute the entire MDS every time new data is added.

The provision of a fast OSE technique solves both problems (discussed in Chapter 3)

in the LSMDS algorithm. Hence, we can easily add new points, making it possible to build a larger embedding using a subset of landmark points. A high-performance OSE technique extends the LSMDS algorithm to address two main concerns:

1. It helps scale the LSMDS algorithm for large-scale data.

2. It provides a means querying or adding out-of-sample data into an existing configuration space.

Our proposed OSE approach uses only a fraction of the existing configuration to map new points. We call these "landmarks" and keep them fixed (defined in Chapter 3). The mapping of a new object is found by minimising the errors in distances to landmarks. Hence our primary solution is an optimisation approach that minimises a cost function to solve the OSE problem. We introduced the OSE problem and our optimisation based solution briefly in Chapter 3.

As discussed in Chapter 5, the number of landmarks ($L$) plays an essential role in the optimisation based OSE method. It produces a quick and reasonable distance approximation for the out-of-sample data with a few landmarks. Increasing $L$ improves the quality of the distance approximation and provides a more accurate embedding. However, using many landmarks is problematic because it increases the embedding RT. Hence, we further investigate the OSE problem to provide an efficient ML based solution.

In this chapter, we propose an OSE technique based on an artificial neural network (ANN) to address the deficiencies of the optimisation-based OSE method introduced in Chapter 3, Subsection 3.2.2. The ANN is trained using the distances to the landmarks from the input data and their relevant coordinate values in the existing configuration space created by MDS. Out-of-sample data is then mapped into this space by the ANN. The training may be slow, but once the ANN is trained, the OSE is fast, so the cost of training is amortised over many OSEs. Thus, this approach is ideal for the query matching problem.

Both approaches use landmarks to reduce the computational overhead of dissimilarity calculations. However, the experimental results suggested that the ANN model is on average $3.8 \times 10^3$ times faster than the optimisation approach (with its optimal parameter setting) for predicting a coordinate of a new data point, and both techniques exhibit similar accuracy for our data. Thus the ANN model is the most efficient OSE technique for large-scale LSMDS. We can perform very fast query matching against large-scale reference databases by combining the neural-network OSE solution with our Em-K indexing (proposed in Chapter 5).

As discussed in Chapter 5, we can use the OSE technique to apply landmark LSMDS as an alternative to complete LSMDS on a whole dataset. The procedure is, (1) select a relatively small subset of landmarks from a large-scale dataset (2) apply LSMDS to the selected landmarks (3) determine the coordinates of the remaining points by applying

OSE. Hence, the major contribution here is the use of ANN to perform the OSE step efficiently.

For a given large-scale data set consisting of $N$ objects, we first apply LSMDS for a selected subset of $L \ll N$ landmarks, with a complexity of $O(L^2)$. These $L$ data points are embedded in a $K$ dimensional space, and we call this a configuration space. We can map the remaining $M = N - L$ objects to the existing configuration applying the proposed OSE approaches with only $O(LM)$ computations. Therefore our approach dramatically reduces the computational complexity.

Similarly, mapping a new object to the above configuration requires only the distances between landmarks and the new object. When mapping a new object, the OSE approaches use landmarks to reduce the complexity of $O(N)$ computations to $O(L)$. Moreover, in the ANN, the computations required are much more efficient than the original LSMDS optimisation.

In summary, our primary contributions are:

- We formally introduce the OSE problem for the LSMDS algorithm. We explain the use of OSE solutions for handling large-scale input data in LSMDS.
- We propose an OSE technique for LSMDS using an optimisation approach.
- We propose an ANN model of OSE for LSMDS that finds the mapping of a new object faster than the optimisation-based solution.

The combination makes LSMDS much more practical for large data or online streaming data.

## 6.2 Related Work

The OSE problem has been actively researched in recent years to improve dimensional reduction algorithms. Beningo *et al.* [196] provides a unified framework of OSE extensions for different DR algorithms such as Local Linear Embedding (LLE), Isomap, Laplacian Eigenmaps, MDS and Spectral Clustering. Many of these algorithms use eigendecomposition to obtain a lower-dimensional embedding of the data on a non-linear manifold. The proposed solutions use the Nyström sampling (a technique that speeds up kernel method computations) to train a model for out-of-sample points without recomputing eigenvectors. Hence, it is only applicable to CMDS. Furthermore, these non-parametric OSE extensions have scalability limitations since the complexity linearly increases with the size of the datasets.

In contrast, Trosset and Priebe [133] extended the underlying optimisation problem to resolve the out-of-sample problem with eigenmaps constructed by CMDS. They solve the optimisation problem numerically, considering the least-squares approximation of the

pairwise inner products of the existing space. We use a similar concept to introduce an embedding approach for LSMDS that directly approximates the dissimilarities with distances rather than pairwise inner products.

Anderson and Robinson [197] have also proposed an approach for placing a new observation into a configuration space based on inter-point distances. The method uses CMDS to produce a configuration space of the data and discriminant analysis to test group differences. Both methods proposed by Trosset and Priebe [133], and Anderson and Robinson [197] have produced identical results when embedding a single point to an existing configuration. However, these methods calculate all the distances between a new point and existing points in the original space for mapping it into the existing configuration and are not suitable for large-scale MDS. In contrast, our method only calculates distances to a fixed set of landmarks when mapping a new object in an existing space created by LSMDS.

Gower [198] added a single observation to an existing configuration by adding a dimension, which alters the configuration space. In contrast, we are interested in embedding a new object into an existing configuration without changing it because we aim to embed many out-of-sample objects.

Bae *et al.* [131] proposed an interpolation approach similar to k-nearest neighbour classification. The method, namely interpolation-MDS (I-MDS), extends the SMACOF implementation of the LSMDS algorithm. In contrast, our methods extend the LSMDS algorithm for its gradient descent based implementation. We follow a similar idea as in [131]; however, there are significant differences between the two approaches.

- The I-MDS method works well with large-scale data, and the input data is usually Euclidean distances. In contrast, our solutions work well for large-scale MDS applications that process non-Euclidean input data.

- I-MDS uses a subsample of the existing objects to find the mapping of the new data. For each new object, it calculates the relevant nearest neighbours and maps them in the existing space by solving an optimisation problem. There are two notable limitations; firstly, the method depends on the nearest neighbours (NNs). Thus, selecting many NNs for each new object reduces the efficiency. Secondly, k-NN search only works for data in a metric space, e.g., input data are points in a Euclidean space. Hence, I-MDS is not applicable to non-metric input data. These limitations reduce the generalisability of the I-MDS algorithm. In contrast, our methods can take non-metric dissimilarities as inputs when handling large-scale LSMDS problems.

The OSE problem has appeared in various contexts. Some of them are bioinformatics [131, 199], sensor networking [135], high performance computing [131] and image processing [200]. For instance, in sensor networks, this problem occurs when one seeks to

automatically map the sensors' locations given the pairwise distances between them and then infer the locations of new targets as they appear [201]. This problem also appears in the Euclidean embedding of network distances [135]. The idea is to assign coordinates to the node in the network such that the Euclidean distances approximate the network delay. The initial configuration is obtained by applying CMDS to a fixed set of designated nodes called 'landmarks'. Then remaining node in the system measures the distances to each landmark and perform nonlinear optimisation to compute their coordinates [202].

ANNs have been used in modelling manifold embedding due to their robust learning framework in recent years. Jansen *et al.* [203] used modern deep learning to perform OSE in graph embedding. Their experiments suggested that deep neural network models outperform the Nyström methods. In [204], t-SNE was combined with deep feed-forward networks to obtain a highly non-linear embedding function. Bunte *et al.* [205] formalised well-known non-parametric DR techniques based on cost optimisation to define explicit mapping functions that facilitate OSE extensions. ANNs and parameterised non-linear functions are used in their formalisation process. These methods can handle large-scale data since the mapping functions learn from a small random subset of the data. However, the current settings of this process only work for Euclidean data.

By looking at the existing methods of OSE for MDS, we found three main limitations:

- Many of them only provide solutions for CMDS, not LSMDS and as a byproduct focus on embedding Euclidean data.

- Most of these existing solutions do not address the large-scale MDS problems. Instead, the focus is on investigating the technical difficulty that arises by inserting a small number of additional points into previously constructed configurations.

- All the methods are tested only on Euclidean data. Hence, the OSE methods for non-metric MDS applications have not been studied well.

## 6.3 Methods

The proposed OSE techniques extend the capabilities of LSMDS for large-scale data. These techniques enable querying/adding new points of previously unseen data in an existing configuration. In this section, we formally introduce the OSE problem. Other relevant definitions, key concepts on the LSMDS algorithm, embedding and landmarks are given in Chapter 3. We first summarise the terminology and the notation used in this chapter in Table 6.1.

Table 6.1: Summary of the main notations used in this chapter.

| | |
|---|---|
| $N$ | Input objects in a metric or non-metric space |
| $\Delta_N$ | Dissimilarity matrix of objects $N$ |
| $X$ | Coordinate matrix of objects $N$ in the configuration space |
| $\delta$ | Dissimilarity or distance measure between objects $N$ |
| $d$ | The Euclidean distances between coordinates $X$ |
| $K$ | Output dimension of a MDS configuration |
| $\sigma$ | Stress of a MDS configuration |
| $y$ | Out-of-sample object |
| $\hat{y}$ | Mapping coordinates of the out-of-sample object $y$ in the configuration space |
| $L_s$ | Set of Landmarks selected from objects in a high dimensional space |
| $L$ | Number of landmarks in the set $L_s$ |
| $\Delta$ | Input training data matrix for the neural network |
| $\delta^i$ | $i^{th}$ training input of the neural network |
| $x^i$ | $i^{th}$ training label of the neural network |
| $k$ | Number of nearest neighbours in a k-nearest neighbour search |
| $B$ | The size of the block return by the Em-K indexing method |

### 6.3.1 Out-of-sample Embedding Problem

The out-of-sample embedding (OSE) maps additional points to a previously constructed configuration. Suppose we have a configuration of $N$ points in a $K$-dimensional Euclidean space obtained by applying LSMDS. Let $y$ be an out-of-sample object, with measured pairwise dissimilarities from each of the original $N$ objects. The OSE problem is to embed the new $y$ object into the existing $K$-dimensional Euclidean space such that it optimises some criteria.

**Definition 6** *Out-of-sample embedding: Finding the coordinates (location) of a new object $y$, from a metric or non-metric space $(N, \delta)$ by mapping it to an existing configuration space $(X, d)$.*

The main idea of our OSE approach is to use only a fraction of the input data that we call landmarks. For more details on landmarks, refer to Chapter 3. The proposed OSE approach extends the LSMDS algorithm to accommodate embedding of large-scale data and query matching against an existing configuration space. When used with large-scale data, the first step is to apply LSMDS only on landmarks. Then the remaining data can be added to the existing configuration as out-of-sample points. Similarly, one can use only a fraction of the existing data (landmarks) to query against a current configuration where the query points are also out-of-sample data.

We have briefly discussed our optimisation OSE technique in Chapter 3. We will describe it in more detail next.

### 6.3.2 Optimisation Method for OSE

We proposed an optimisation approach similar to the stress minimisation of the LSMDS algorithm to solve the OSE problem for LSMDS. However, we accomplish the OSE method with a different objective function because the original LSMDS objective involves all $N^2$ distances in the original set. In contrast, the OSE involves only the distances between a new object and the landmarks.

The proposed method is implemented as follows. For a given dataset of $N$ objects in a metric or non-metric space with dissimilarities ($\Delta_N = [\delta_{ij}]$), a configuration of $\mathbf{X} = [x_1, x_2, .., x_N]$ points in a $K$-dimensional Euclidean space, where $x_i \in \mathbb{R}^K$ is constructed by applying the LSMDS algorithm.

Then we choose a set of landmarks $L_s = l_1, ...l_L$, from the $N$ objects and the corresponding $\hat{l_1}, ...\hat{l_L}$ points from the $N$ pre-mapped points in the configuration space.

Finally, the mapping of the new point $y$ to $\mathbb{R}^K$ is calculated based on the pre-mapped positions of $L_s$ and the corresponding proximity information $\delta_{l_i y}$. Finding a new mapping position is considered a minimisation problem of stress similar to a standard LSMDS problem with $m$ points, where $m = L + 1$. However, only one point $y$ is movable among $m$ points. Hence the OSE of a new object $y$ is obtained by minimising the following objective function,

$$\hat{\sigma}(y) = \sum_{i=1}^{L} \Big( \left\| \hat{l_i} - \hat{y} \right\|_2 - \delta_{l_i y} \Big)^2. \tag{6.1}$$

The $\delta_{l_i y}$ represent the original dissimilarity value between $i^{th}$ landmark point and the new embedding point $y$ in the original space. The Euclidean distance between $i^{th}$ landmark and point $y$ in the existing configuration space in $K$ dimensions is given by $\left\| \hat{l_i} - \hat{y} \right\|_2$. Thus $\hat{\sigma}(y)$ is used to match the original dissimilarities of the object $y$ to the landmarks into the Euclidean distances in the configuration space.

We seek to find a position of $\hat{y}$ that minimises $\hat{\sigma}(y)$. The proposed OSE algorithm simplifies the complicated non-linear optimisation problem to a small non-linear optimisation problem, such that $N$ points to $L + 1$ points non-linear optimisation problem, where $L \ll N$. Note, however, that it still requires the solution of a quadratic optimisation problem.

### 6.3.3 Neural Network Method for OSE

An ANN is a machine learning model that creates predictions based on existing data (training data). These are inspired by the biological neural networks, which process

information in the human brain using neurons [61]. ANNs can consist of several layers or many layers (deep neural networks). They solve different types of problem areas, including classification, regression, multi-output or multi-class problems.

First, we provide an overview of the ANNs in the following prior to our model explanation.

**Overview**

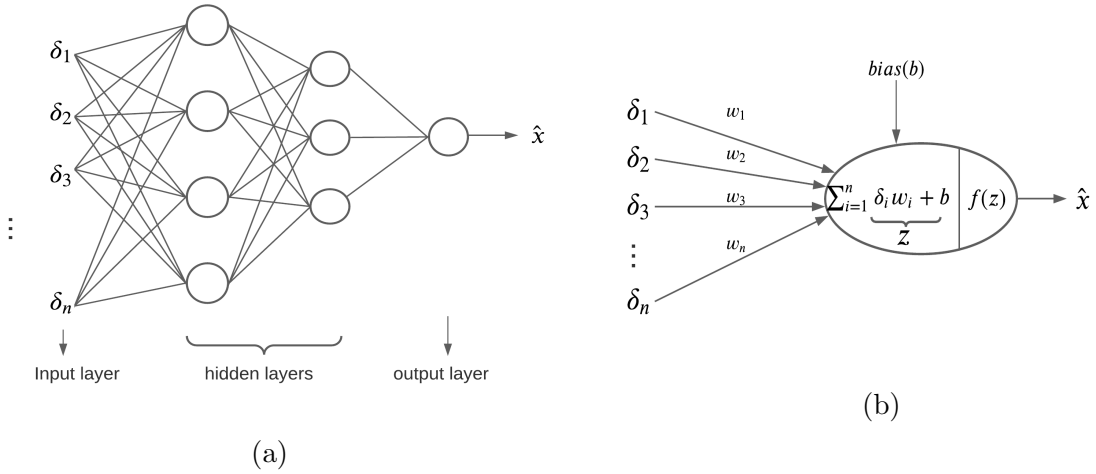



Figure 6.1: (a) A neural network architecture with two hidden layers. It takes the input vector $\delta_1, \delta_2, .., \delta_n$ and processes it through the hidden layers to produce the output $\hat{x}$. (b) Shows the input, output and functions of a single neuron. It calculates output of the linear function $\sum_{i=1}^{n} \delta_i w_i + b$ for all the inputs and then passes it through the activation function $f$ to get the output $a$.

An ANN consists of many simple processing elements, called neurons, each possibly having a small amount of local memory. Figure 6.1-b shows the structure of a single neuron. These are organised into a sequence of layers linked by weights. The basic architecture of an ANN consists of:

- Input layer: loads the input data from external sources to feed them into the model.
- Hidden layers: intermediate layers that do all the computations and feature extraction from the input data.
- Output layer: takes input from preceding hidden layers and comes to a final prediction based on the model's learnings.

The networks are learned by processing the inputs and comparing the resulting outputs with the desired outcomes. The difference is then propagated to the network by adjusting its weights to minimise the error. It is an iterative process, and the weights are continually adjusted until the system converges to an optimum solution. Figure 6.1-(a) illustrates the basic architecture of a simple neural network with four layers: the input layer, two hidden layers and the output layer.

At an abstract level, an ANN is a function. Hence we can define the function $f_\theta : \Delta_y \rightarrow \hat{y}$ parameterised by $\theta$ which takes the input $\delta \in \mathbb{R}^L$ and produce the output $\hat{y} \in \mathbb{R}^K$ [206].

### Neural Network Model

We formulate the OSE problem as a regression problem. Regression refers to predictive modelling problems that involve predicting a numeric value for a given input. Our task requires predicting more than one numerical value, usually known as the multi-output regression. Among the family of NN models, we choose the multilayer perceptron (MLP), i.e., a multilayer, fully connected feed-forward network, to solve our out-of-sample regression problem. MLPs are general-purpose, flexible, nonlinear models that can approximate virtually any function to any desired degree of accuracy given enough hidden neurons and data.

As before, a configuration is obtained for $N$ points in $K$ dimensional space by applying the standard LSMDS algorithm. Then we choose a set of landmarks $L_s = l_1, ...l_L$ from the $N$ pre-mapped points.

Then the input data to the NN model is as follows.

- Input training data: The pairwise distances between the landmarks $L_s$ and the $N$ points measured in the original space. Hence the input data is a matrix $\Delta_{LN} \in \mathbb{R}^{L \times N}$.

  The column vector of the $i^{th}$ sample in $\Delta_{LN}$ has the form $\delta^i = [\delta_{il_1}, \delta_{il_2}, \delta_{il_3}, .., \delta_{il_L}] \in \mathbb{R}^L$, where $\delta_{il_j}$ is the distance between a landmarks and the $i^{th}$ point given $i \in 1, 2, ..., N$ and $j \in 1, 2, ..., L$.

- Output training labels: The corresponding coordinate representation of $N$ points in the $K$-dimensional Euclidean space. The output labels of the $i^{th}$ sample is $x^i \in \mathbb{R}^K$.

- Input testing data: The pairwise distances between the landmarks and a new point $y$ measured in the original space. Similar to the training data, the input testing data takes the form $\delta^y = [\delta_{yl1}, \delta_{yl_2}, \delta_{yl_3}, ..., \delta_{yl_L}] \in \mathbb{R}^L$. Testing data would usually include labels; however, we do not have labels for the out-of-sample points.

Our MLP model is a by-now standard model with three hidden layers that use the ReLU activation function [206, 207]. The size of the input layer is equal to the number

of landmarks $L$, whereas the size of the output layer is $K$, representing the dimension of the output data. The sizes of the hidden layers are estimates of the intrinsic dimension of the previous layers.

In each learning step, the ANN is presented with an input column vector $\delta^i$ and the label $x^i$. We will fit the model using the mean absolute error (MAE) loss function and the Adam version of stochastic gradient descent [206].

The MAE measures the average magnitude of absolute differences between $N$ label vectors $X = x_1, x_2, .., x_N$ and predicted vectors $\hat{X} = \hat{x_1}, \hat{x_2}, ..., \hat{x_N}$, with the corresponding loss function; written as,

$$Loss_{MAE}(X, \hat{X}) = \frac{1}{N} \sum_{i=1}^{N} ||x_i - \hat{x_i}||, \tag{6.2}$$

where, $||x_i - \hat{x_i}||$ is the Euclidean distance between the label vector and the predicted vector of the $i^{th}$ sample of the input data.

Starting with initial random weights, MLP train the model to minimise the loss function by repeatedly updating these weights. The backpropagation algorithm is used in NN models to feed the loss backwards from the output layer to the previous layers, updating weights to decrease the loss.

A comprehensive introduction to these topics is outside the scope of this thesis. See instead [208, 206] for a general introduction to deep learning, and [209], for backpropagation.

The out-of-sample LSMDS steps for both methods described above are as follows; (1) apply LSMDS on the training data, (2) choose landmarks from the training data, and (3) apply any of the OSE techniques for the remaining data points based on the embedding results of the training data. Algorithm 3 shows the steps of the OSE techniques when used to embed large-scale data applications. We will discuss the results in Section 6.4.

## 6.4 Experimental Analysis

We evaluated the performance of the two methods under various settings. Two main questions studied in these experiments are: (1) Are these proposed methods robust over multiple settings? And (2) Do any of the methods outperform the others in terms of scalability and accuracy?

All algorithms are implemented in R and executed on a desktop with an Intel Core 5 Quad 2.3GHz, 16GB RAM, and MacOS Big Sur. We use Keras API and Tensorflow as its backend for developing our neural network model.

**Algorithm 3** Out-of-sample embedding algorithm.

**Input:** Dataset $N$, *Method*
Select $L$ landmarks from $N$, randomly or using FPS.
Apply LSMDS to $L$ (using Algorithm 2)
Select the remaining points $M = N - L$
Measure the distances to landmarks
# string distances are measured on LV distances
# OPT = optimisation approach and NN= neural network model
**if** *Method*==OPT **then**
  **for** $y$ in $M$ **do**
    Apply the optimisation method using (6.1)
    predict the mapping position of $y$
  **end for**
**else if** *Method*=NN **then**
  Train the neural network
  **for** $y$ in $M$ **do**
    predict the mapping position of $y$
  **end for**
**end if**
Return $\hat{y}$.

### 6.4.1 Data Sets

We examined the performance of our methods over synthetic datasets that contain entity name strings. These data are generated using the Geco tool in FEBRL [164], where we can manipulate the data generation to significantly vary the size and characteristics (e.g., error rates) of the data. The Geco tool allows us to create unique entity names as well as duplicate entries of entity names. Each entity has a given name and a surname. We introduce duplicate entity names that contain errors by slightly modifying the values of randomly selected entries. In data generation, we assume a single entity has only one duplicate and a maximum of two typographical errors (substitutions, deletions, insertions, and transpositions) in both of its attribute values. In this chapter, we mainly used 5000 randomly selected name strings and 500 name strings as out-of-sample points for testing, which allowed us to perform resampling to evaluate the results for consistency. However, we show that our methods are scalable to large-scale datasets.

In this work, we used the Levenshtein (LV) distance (defined in Chapter 2, Subsection 2.5.3) to measure the similarity between entity name strings.

### 6.4.2 Performance Evaluation

We evaluate the results of the two OSE techniques based on an error metric $Err(m)$, similar to the *stress* defined by the equation (3.2) in Chapter 3. It measures the distortion of the distances between new and existing points before and after the OSE transformation. The point error, $PErr(y)$ measures the distances between a new point and the existing points before and after the OSE transformation. We defined them as follows;

The point error $PErr(y)$ of embedding a new point is given by,

$$PErr(y) = \sum_{i=1}^{N} (\delta_{iy} - ||x_i - \hat{y}||)^2, \tag{6.3}$$

where the distance between the $i^{th}$ pre-mapped point and a new point $y$ in the original high dimensional space is given by $\delta_{iy}$, and the Euclidean distance between the same points in $K$ dimensions is given by $||x_i - \hat{y}||$.

The error $Err(m)$ of embedding $m$ new data points in a $K$ dimensional Euclidean space with a pre-mapped $N$ data points is given by,

$$Err(m) = \sum_{i,j=1}^{N,m} \frac{(\delta_{iy_j} - ||x_i - \hat{y_j}||)^2}{\delta_{iy_j}^2}, \tag{6.4}$$

where $\delta_{ij}$ is the distance between the $i^{th}$ pre-mapped point and $y_j$ new point in the original high dimensional space, and $i \in 1, 2, ..., N$ and $y_j \in 1, 2, ..., m$. The Euclidean distance between the same points in $K$ dimensions is given by $||x_i - \hat{y_j}||$.

The error $Err(m)$ measures the embedding error of new points $m$ with respect to the $N$ points in the existing configuration. We calculated $Err(m)$ to compare the accuracy of the two OSE methods. Additionally, we used scatter plots to compare the point error values calculated for embedding each new point in $m$ using either OSE method.

This approach measures the accuracy of the embedding; however, we also want to understand its performance in ER query matching. To evaluate the performance of the two OSE methods for ER, we used the two measures defined in Chapter 5, Subsection 5.5.1. First, we use the number of true positives ($|TP|$) per computational effort to measure the number of true matching records determined by the Em-K indexing when processing queries within a fixed period. Second, we measure the precision (P) per computational effort.

We also compared the efficiency of the methods using the computational time of processing out-of-sample data. All the CPU running times are measured in *seconds* and denoted by RT.

### 6.4.3 Choice of Parameters

Several factors impact the performance of the proposed OSE methods; some of them (e.g., dimension ($K$) and landmarks ($L$) ) are control parameters that we rely on to fine-tune the performance. The datasets determine the other important factors, such as dataset size and duplicate rates.

In this section, We discuss the choices of $K$, $L$ parameters in our OSE implementations (and their rationale). In detail, we applied LSMDS and the proposed OSE techniques to a sample of data generated using Geco [164]. We chose 5000 name strings as the reference data points to create the initial configuration and separate 500 names as out-of-sample points. The reference data points and the out-of-sample points represent unique entities.

**K**: The dimension of the Euclidean space ($K$) plays an essential role in the performance of the LSMDS algorithm. We have discussed good values of $K$ appropriate for embedding string data, such as entity names in previous chapters. Following the findings based on the trade-off between *stress vs dimension* in Chapter 5 and Chapter 6, we set $K = 7$.

**L**: The number of landmarks is a dominating factor that directly affects the effectiveness and efficiency of the proposed OSE techniques. Landmarks reduce the computational overhead of distance calculations. Since only a fraction of the initial data is used to map a new point to the configuration space, it also reduces the total running time of the OSE process.

We applied two landmark selection methods: FPS and random selection (explained in Chapter 3, Section 3.2). The following results are based on the FPS. The number of landmarks to be used remain to investigate in what follows.

The use of a few landmarks only requires a few distance calculations. Hence, the method becomes efficient for large-scale data processing. In contrast, fewer landmarks might produce a less accurate approximation of the mapping positions for new data. We have to consider this trade-off when choosing an adequate number of landmarks appropriate for our data. The following experiments measure the effect of landmarks on the performance of the proposed OSE techniques. The performance is measured based on the accuracy of the distance approximation and the average RT of mapping a single point using either of the methods.

#### Comparing the Total Error

First, we applied LSMDS to the selected 5000 strings of reference points. The dissimilarities between strings are measured using LV distance. Then the remaining 500 strings (out-of-sample points) are mapped using the proposed OSE methods, only measuring the distances to the landmarks. We measured the total error ($Err(m)$) generated by the out-of-sample data mapping in the configuration space for different instances. We obtained two sets of results (in terms of total error) for both OSE techniques by keeping the $K = 7$ fixed and varying $L$.

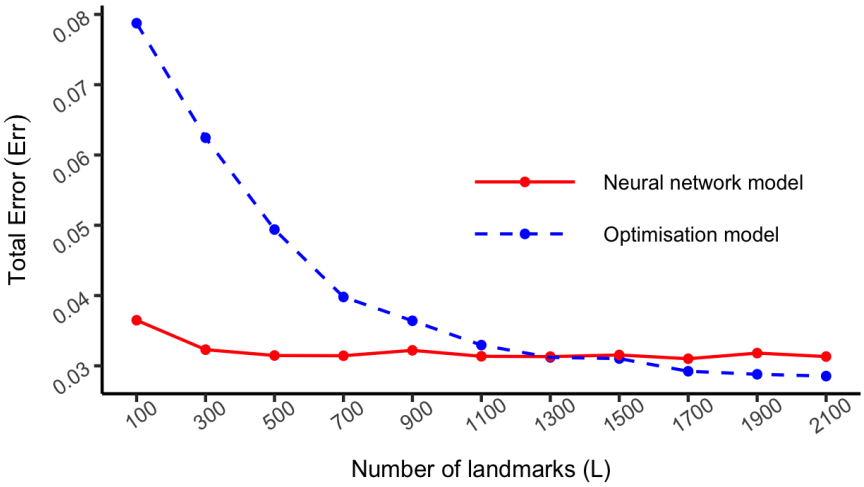



Figure 6.2: A comparison between the proposed out-of-sample techniques: neural network model and the optimisation method. The total error ($Err(m)$ in Equation 6.4.2) represents the distance distortion created through the mapping process when mapping out-of-sample data from the high dimensional space to the Euclidean space. The $Err(m)$ decreases rapidly for the optimisation method. In contrast, the total error fluctuates in small amounts for the neural network when the number of landmarks increases in each instance.

Figure 6.2 compares the two OSE techniques; the NN model and the optimisation method. We utilised the same set of landmarks when applying both methods in each instance. Let $Err_o(m)$ and $Err_{nn}(m)$ be the total errors produced by the optimisation method and the NN model, respectively.

The optimisation method reduces the $Err_o(m)$ drastically until the number of landmarks is 1000. Then $Err_o(m)$ reduction gradually slows down. The $Err_o(m)$ approaches a non zero asymptotic, and it depicts that the increasing number of landmarks increases the accuracy of the OSE.

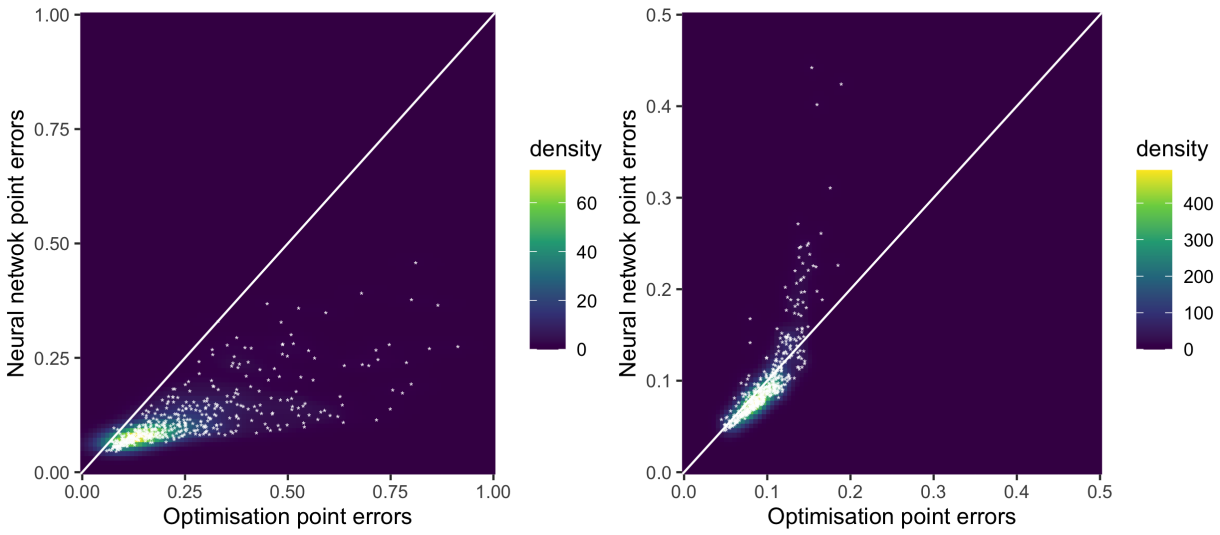
Small $L$ tends to produce higher $Err_o(m)$ values in the optimisation method since only a few distance calculations are used in the approximations. If $L$ is set very high ($L = 2000$), the $Err_o(m)$ decreases as we have more distances to approximate the mapping of the new data points. However, having a large number of landmarks can increase the processing time of the OSE. Thus $L$ should be set neither too high nor too low, suggesting landmarks between 1000 to 1500 gives a reasonable $Err_o(m)$ for the overall approximation of the given data set when applied the optimisation approach.

In contrast, the NN model only shows a significant improvement when landmarks increase from 100 to 300. After that, $Err_{nn}(m)$ does not significantly reduce with the increasing number of landmarks used in the model training. The results suggested that the neural network OSE method performs well with fewer $L$ and sufficient training data.

In contrast, the optimisation method requires more landmarks to achieve the same level of accuracy as the NN model. While both methods perform similarly around 1100-1500 landmarks, the NN model works very efficiently around 100-300 landmarks with a slightly reduced accuracy.

### Comparing the Point Errors

Next, we looked at the error of distortion for each out-of-sample point separately. The experiment settings are the same as before, and instead of the total error, we calculated the distance approximation errors separately for each out-of-sample mapping. We applied the OSE methods to map each new string to the existing configuration space and calculated $PErr(y)$ (Equation 6.4.2) for every single mapping. We obtained two sets of results produced by the two methods. Figure 6.3 compares those point error values obtained for the underlying out-of-sample data points. Here we have normalised the $PErr(y)$ by $\delta_{iy_j}$ which is the sum of the dissimilarities between all the new points and the existing points in the original space.




(a) L=100 (b) L=1500

Figure 6.3: The comparisons between the point errors of embedding out-of-sample data using the neural network and the optimisation methods. The NN model is better for the points below the $y = x$ line, and optimisation is better for the points above the line. a) The point errors produced by both OSE approaches with 100 landmarks. The NN model produces small errors compared to the optimisation b) The point errors created by both OSE approaches with 1500 landmarks. Both methods make similar errors with many landmarks, and the magnitude of the point errors are much smaller.

Ideally, we want point errors to be zero for each OSE point. However, they cannot be zeros since our methods only approximate the distances between new points and landmarks, excluding the distances between the remaining points and the new points in the embedding calculations. On the other hand, no exact solution exists for our data since the embeddings are invariant to position, and the methods only approximate the relative distances. The point errors are calculated considering the distances from a new point to all the existing points to measure the error of distance approximation produced by the proposed OSE techniques. Hence, our methods tolerate a small amount of error in mapping a new data point to provide a more efficient solution at the expense of slight accuracy loss.

Figure 6.3 describes how the point errors change with the number of landmarks when the proposed OSE techniques are applied for new data points. We have measured the point errors of mapping 500 out-of-sample entity name strings in a 7-dimensional Euclidean space. The plots are drawn only for $L = 100, 1500$, showing significant improvements with more landmarks.

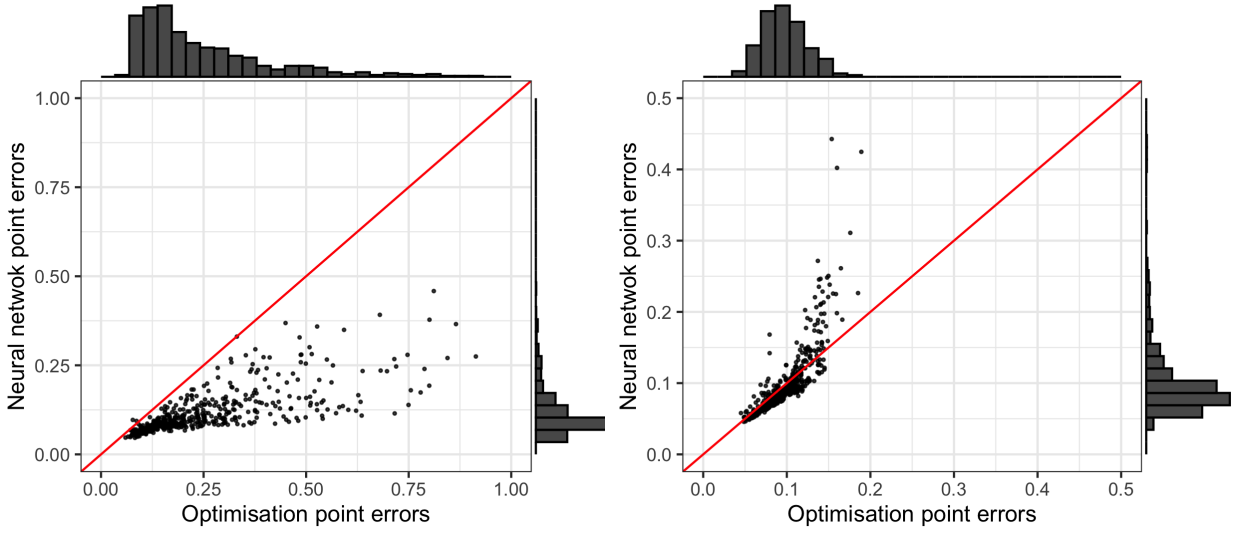
Figure 6.3-(a) illustrates $PErr(y)$ comparisons of the mapping based on $L = 100$ landmarks. All the point errors produced by the NN model are less than 0.5, whereas the point errors produced by the optimisation method are less than 0.75. It depicts that the NN model can approximate the mapping of a new point with few landmarks producing a small error. In contrast, the optimisation method creates comparatively large point errors in the mapping with few landmarks.

Figure 6.3-(b) illustrates $PErr(y)$ comparisons of the mapping based on $L = 1500$ landmarks. Increasing the number of $L$ has improved the accuracy of both methods. All the point error values produced by the optimisation method are below 0.25. Similarly, 95% of the point error values produced by the NN model lies below 0.25. Hence, with a sufficient amount of landmarks, we can improve the accuracy of both methods where the optimisation method remains as good as the NN model.

Next, we looked at the $PErr(y)$ distribution produced by both approaches. Figure 6.4 compares the point errors and their distributions of the two OSE methods. These figures extend the $PErr(y)$ values shown in Figure 6.3 to their distributions. When $L = 100$, the error distribution of the neural network results shows a small spread indicating small variance, whereas the optimisation results show a comparatively wider spread indicating slightly higher variance. In contrast, both distributions have similar shapes and a smaller spread indicating small variances when $L$ increases to 1500. Nevertheless, using many landmarks in the embedding can be inefficient for large-scale data. In the following experiment, we explain the running time complexities of the proposed methods.

### Comparing the Average Running Time

We evaluated the efficiency of the methods measuring the RT of mapping a new point in the existing configuration space. The number of landmarks $L$ directly impacts the




(a) L=100 (b) L=1500

Figure 6.4: The comparisons between the point errors of mapping out-of-sample points and their distributions. The neural network model and the optimisation method are applied to map out-of-sample points into a 7-dimensional Euclidean space. a) The point errors and their distributions created by both out-of-sample approaches with 100 landmarks. The point error distribution of the neural network results shows a small spread indicating small variance, whereas the optimisation results show a comparatively wider spread indicating high variance. b) The point errors and their distributions created by both out-of-sample approaches with 1500 landmarks. Both distributions have similar shapes and a smaller spread indicating small variances when the number of landmarks increases to 1500.

running time (RT) of the proposed OSE methods since each new point is mapped to the existing configuration space using landmarks. Both methods map a single out-of-sample point at a time.

Figure 6.5 compares the average running time (RT) of mapping an out-of-sample point in the configuration space for both methods, varying the numbers of landmarks. Increasing $L$ increases the average mapping time of a single point linearly in the optimisation method. The average mapping time increases linearly with $L$ for the neural network model, which is not visible due to the scaling. When $L = 100$, the average RT of mapping an out-of-sample point only takes $0.013 \times 10^{-2}$, and with $L = 2100$ the average RT is $0.2 \times 10^{-2}$ for the NN model.

The results show that the NN model is faster than the optimisation method regardless of the number of landmarks used in the mapping. Both methods spend less RT around 100 landmarks and increase linearly with $L$. The total rates of change in RT with respect to

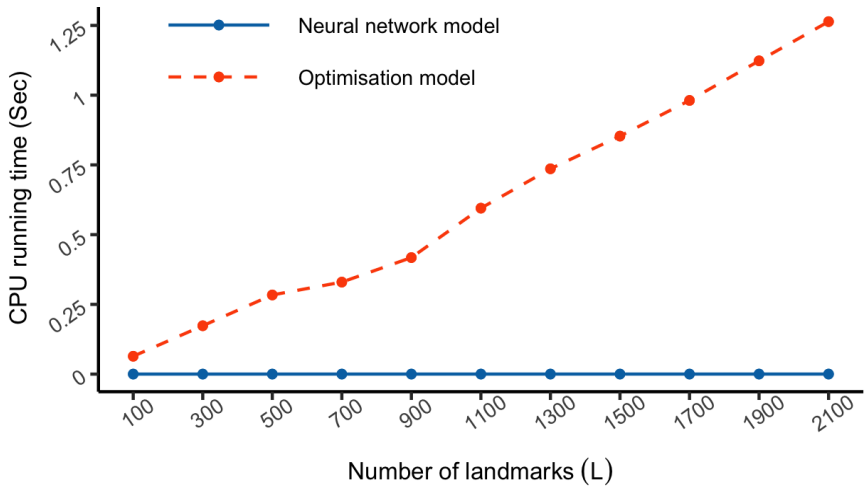



Figure 6.5: A comparison of the average running time of mapping a single query in an existing configuration space using the proposed out of sample methods: neural network and optimisation. Each instance represents a different number of landmarks used in the mapping. The running time increases linearly with the number of landmarks for both methods, although only the optimisation method shows a significant increase in the RT rate per landmark.

landmarks are 10% and $7.5\times10^{-4}$% for the optimisation and NN models, respectively. The RT grows gradually (small gradient) for the NN model and drastically (large gradient) for the optimisation model. However, a small number of landmarks can reduce the accuracy of both methods. This was a significant observation in our previous experiments, as illustrated in Figure 6.2.

These results also support our earlier remarks on the neural network that emphasise that the model can be very fast with a small cost of accuracy. We also discussed that both methods have similar accuracy levels between 1000 to 1500 landmarks. However, the NN model is on average $3.8 \times 10^3$ faster than the optimisation method around $L = 1000, 1500$ (where the best performance of optimisation occurs). Hence, we can achieve high accuracy and efficiency using the NN model with fewer resources than the optimisation method. Finally, we conclude that the NN model is a highly accurate and efficient method and an excellent substitution for the optimisation OSE method. With less than 1000 landmarks, the NN model can map an out-of-sample point within $1.7 \times 10^{-4}$ seconds for our data. Hence, the average running time of mapping an out-of-sample point into an existing configuration space takes less than a millisecond.

## 6.5 Application in Query Matching

The out-of-sample embedding is an important component in the Em-K indexing method proposed for query matching in ER, as it might be in other query matching approaches. We have discussed how OSE of LSMDS can handle the embedding of large-scale reference databases and map previously unseen data to existing configuration space. In the following experiments, we compare the performance of the proposed OSE methods when applied to the Em-K indexing methods.

The indexing method processes a single query at a time, applying OSE for mapping it to a point in the configuration space. We used both OSE techniques to obtain two sets of results for the query matching. We have discussed the choice of parameters $L$ and $K$ for both methods in the previous section. Based on those parameters, we measured the scalability and the accuracy of the Em-K indexing method in the following experiments.

Similar to the experiments explained in Chapter 5, Subsection 5.5.3, we used a reference database $E_r$ with 5000 entity records. The streaming set of 500 entity queries is selected to match against $E_r$. We assumed that each query has only one matching record in $E_r$, i.e., $QMR = 1$. The Em-K indexing method first embeds a query in the configuration space that contains the pre-mapped reference database. Then it returns a block of candidate matching points of the query by efficiently searching for k-NNs in the configuration space.

First, we embed $E_r$ applying the complete LSMDS. We will be using OSE only for query matching. Hence, the algorithm design differs slightly from the Em-K indexing method in Chapter 5, Subsection 5.4.2. The difference is that we embedded a reference database to the configuration space using landmark LSMDS in the previous work, whereas we use complete LSMDS here. Hence, the embedding of the reference database is consistent when comparing the proposed OSE techniques. However, in practice, for a very large-scale reference database, we need to apply landmark LSMDS since the complete LSMDS may become less efficient. Building the Kd-tree and the k-NN search for creating blocks of similar records for querying entities is the same as in Subsection 5.5.3. We will call the modified indexing method *Mod-Em-K indexing*.

We measured the scalability of the method based on the number of true positive ($|TP|$) match detection per computational effort. The query stream is processed within a given period based on the parameter selection ($L$ and $K$) in Subsection 5.5.2. We used both OSE methods to find matching records for queries utilising the existing configuration space and the same set of selected landmarks.

Figure 6.6 compares the proposed OSE methods on the number of true positives detected per computational effort. In each instance, we changed $L$ to measure the scalability and accuracy of the methods. We ran 500 queries against the reference dataset, limiting the total query time to one minute (60 seconds). The Mod-Em-K indexing method search for a block of matching records for each query using the k-NN search. Hence, we kept the

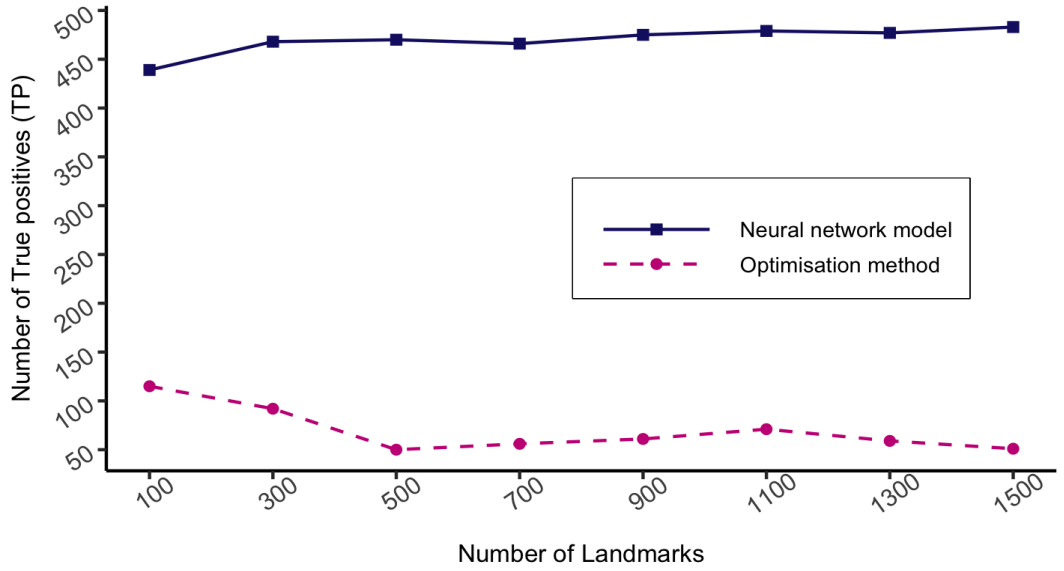



Figure 6.6: A comparison between the proposed out-of-sample methods: neural network and optimisation when applied with the Mod-Em-K indexing to solve the query matching problem in ER. The comparison is on the number of true positives found by each method per computational effort. We kept the parameters $K = 7, T = 60$ seconds and block size $(B)$=150. Many landmarks allow fewer queries to process for the optimisation method, whereas the neural network model process almost all queries regardless of the number of landmarks.

number of nearest neighbours ($k$, also defined as the block size, $B$) fixed to 150 based on our findings in previous experiments in Chapter 5, Subsection 5.5.3.

Figure 6.5 suggested that many landmarks significantly increase the embedding RT of a single query in the optimisation method. Similarly, when we increase $L$, the number of queries processed within a minute decreases. We call this the detected number of queries which decreases from 500 ($L = 100$) to 63 ($L = 1500$) for the optimisation method. However, increasing $L$ increases the $|TP|$ matches detected per computational effort. Figure 6.6 shows that the optimisation method only finds 115 true positives for all the detected 500 queries with 100 landmarks, whereas it finds 51 true positives for the detected 63 queries with 1500 landmarks.

In contrast, the NN model maintains a high level of accuracy, increasing the $|TP|$ detected each time with respect to the increase of $L$. The model finds candidate matches for all the queries within the given minute because of fast query processing (as shown in Figure 6.5). When $L$ is larger than 100, the NN model has detected more than 90% of the true matches. Based on the results, we conclude that both methods achieve a high level of accuracy, around 1000-1500 landmarks. However, the NN model outperforms the

optimisation method allowing fast query processing.

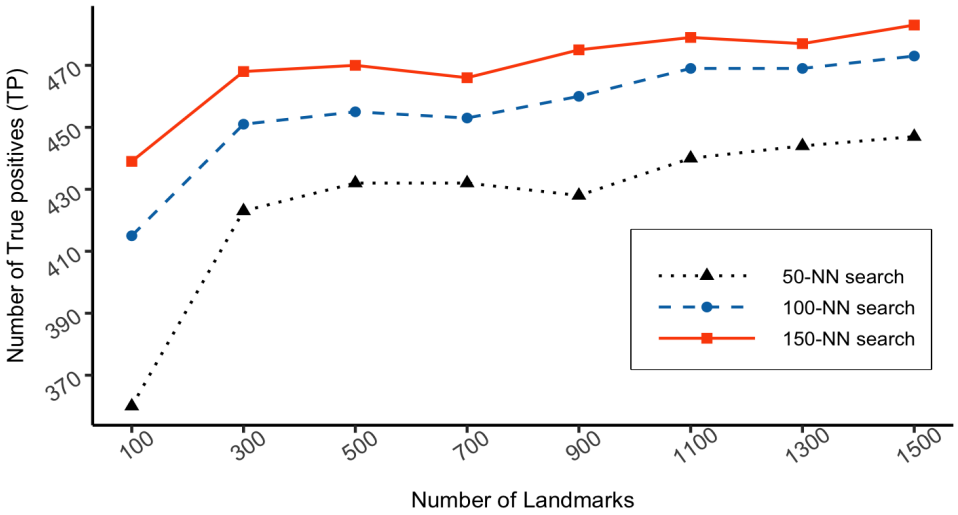



Figure 6.7: A comparison of three different block sizes ($B$) used in Mod-Em-K indexing based on the neural network OSE to find candidate matches for queries. We processed 500 queries against the reference data within a minute ($T = 60$ seconds), changing the number of landmarks and the block sizes (number of nearest neighbours) in each instance. There is a trade-off between the number of true positives and landmarks for different block sizes. The number of true positives increases with the number of landmarks and the block sizes.

We also looked at the effect of changing $B$ in the Mod-Em-K indexing performance when applied with the NN model. This is similar to the experiment described in Chapter 5, Figure 5.7 which uses the optimisation OSE technique in the Em-k indexing. We changed $B$ by changing $k$ for each query. Figure 6.7 shows that increasing $B$ increases the $|TP|$ found within the given period. Similar to the previous experiments, increasing $L$ increases the $|TP|$ detected by the model. Based on Figure 6.6, we conclude that setting $B = 150$ and $L = 1000, 1500$ gives the best trade-off between the quality of the results and the efficiency of the method for the given dataset.

The overall results suggested that the proposed NN model of the OSE improves the accuracy and efficiency of the Em-K indexing. We only need a few landmarks to achieve the most number of TP for a stream of queries within a set period. Unlike the optimisation method, we do not need to sacrifice accuracy to gain more efficiency. Even though the NN model training takes some time, the query processing is fast once the model is trained. As a result, a single query can be processed within less than milliseconds against a reference database to find candidate matches.

## 6.6 Discussion

The NN model maps a new point into an existing configuration space within less than a millisecond, achieving a good approximation for the actual distances.

In contrast, the optimisation method requires many landmarks to achieve the same level of accuracy. Many landmarks are computationally expensive since they increase the embedding RT. Hence, there is a trade-off between the accuracy and efficiency in the optimisation approach, which depends on $L$. However, this method directly predicts the mapping coordinates of a new object in an existing configuration space using distances to the landmarks. In contrast, the NN model requires training to build the model, which takes time. However, once the model has been trained, the OSE of new data is immediate. Since our NN model is a simple MLP with three hidden layers, it only takes approximately 1.2 seconds to train the model for our data.

The optimisation method requires an initial guess (or a "starting point") for the predicting value. The choice of a starting point determines how quickly the algorithm converges to a solution. We choose a vector of all 0s as the initial guess of mapping a new object in the configuration space in our experiments. Improving the initial guess is non-trivial since the initial configuration of the configuration space provided by the LSMDS is not invariant to orientation or position. Hence, one of the reasons for the optimisation method to provide less accuracy can be the sensitivity of the algorithm to the initial guess.

We applied the complete LSMDS to embed a reference database in our experiments. However, we can train an ANN using only a subset of the data for very large databases and map the remaining points using landmarks. Hence, the NN model can speed up large-scale data embedding using landmarks that scale the standard LSMDS algorithm.

## 6.7 Summary

In this work, we have proposed and tested two OSE methods that extend the LSMDS algorithm to large-scale datasets. The proposed techniques are based on an optimisation approach and an artificial neural network. Both methods proved fast running time and scalability along with the data sizes. However, the NN model is faster than the optimisation method. Hence we can efficiently map new data points into an existing configuration space created by LSMDS using the NN model. We have improved the performance of the Em-K indexing method by applying the NN model for query mapping.

We combined the ideas of landmarks and OSE to address the scalable issues in LSMDS. The basic idea is first to embed a subsample of the input data by applying LSMDS and then map the remaining data as out-of-sample points. Hence, we can apply the NN model to embed large-scale data sets into configuration space.

# Chapter 7

# Conclusions and Future Work

In this thesis, we have addressed several shortcomings in large-scale entity resolution (ER). This chapter summarises our contributions and discusses directions for future research.

## 7.1 Summary

ER is a traditional problem with many applications that have been studied for years. However, it remains an open research problem, especially for applications in large-scale data. In this thesis, we introduced methodologies for tackling large-scale ER to address the problems of the shortage of real-world test data and fast indexing algorithms. Our approach improves the state-of-the-art ER techniques in several ways.

First, we proposed a numerical test-data synthesis tool that produces realistic large-scale data to develop new methods when suitable public data is inaccessible. One of the key findings of this work is the approximation of vectors that represent names and their relationships, e.g., dissimilarities and similarities. In addition to the large-scale data simulation, it enables efficient nearest neighbour searches for approximate record matching using their vector representations. Based on this, we proposed a fast indexing technique suitable for real-time and approximate entity matching in large-scale ER.

We used LSMDS as our primary embedding technique for mapping string-based entity record values to a metric space, specifically a Euclidean space. However, LSMDS has some deficiencies that impede the capabilities of our methods. To overcome the inefficiencies in LSMDS and improve the efficiency of the proposed metric space-based solutions, we proposed two out-of-sample embedding (OSE) techniques.

We thoroughly tested our techniques using extensive experimental and simulation studies involving three synthetically generated and publicly available datasets. Their outcomes validate that our methods achieve an excellent balance between accuracy and efficiency under varying settings. In the following, we explain a few more detail about our distributions.

### 7.1.1 Summary of Our Contributions

The following contributions address the shortcomings identified in the existing large-scale ER literature (as discussed in Chapter 2, Section 2.7).

- Numerical simulation tool: As mentioned, the lack of large-scale test data collections and publicly available benchmark data sets is a primary concern in ER. We proposed a simple, inexpensive, and fast simulation model that can generate name-like vectors, including simple errors. Dissimilarities between entity identification keys are the main property of interest in ER comparisons. Hence, Our model simulates at the level of detail required to construct the distance between those keys.

  The simulation tool generates name-like vectors and their errors mimicking name dissimilarities of an actual namespace. These vectors can be used to evaluate the performance of large-scale data solutions, especially those that involve name matching. Our extensive experiments on simulating large datasets show that the proposed method is efficient and scalable for generating large-scale name-like vectors.

  In Chapter 4, we discussed how to simulate simple vectors in a space that approximates the properties of names as one step towards being able to generate large simulated datasets for large-scale testing of global matching techniques.

  One of the significant contributions in this work is the name approximation using vectors. We used LSMDS to transform a set of name strings into vectors that approximate an actual namespace. This approach shows that we can preserve relationships, such as dissimilarities between names in Euclidean space, by approximating them as vectors.

- Em-K indexing method: As discussed in Chapter 1, some real-time ER applications demand entity query matching against large-scale reference databases with prompt responses. Toward this challenge, we propose the Em-K indexing method that provides a quick and accurate block of candidate matches in the reference databases for a querying entity. In Chapter 5, we discussed the use of metric space indexing for efficient large-scale query matching.

  Following the concept of the name approximation, our indexing method transforms a set of records into vectors in a metric space, in particular a Euclidean space. These vectors support fast and approximate nearest neighbour search when used with efficient data structures. We used k-NN search and Kd-trees to find similar points for a querying point in the Euclidean space. The method finds a set of potential matches for a query to narrow down the candidates for a detailed level comparison. Hence, our method acts as a filtering method that reduces the number of comparisons needed in a query matching problem.

Our method can process a stream of queries against a large-scale data set within a fixed time, returning as many of the matching records as possible. The method is fast and robust for real-time and approximate query matching.

- Out-of-sample embedding techniques: LSMDS is a standard embedding technique we used as the basis of our metric space based ER solutions. Using LSMDS, we project entity records into a Euclidean space by approximating their dissimilarities or distances. LSMDS does not have an explicit OSE function, and mapping new data into an existing configuration is impossible without recalculating the entire configuration from scratch. LSMDS is a better approach for a complex problem, although it can be inefficient for large-scale data similar to other MDS techniques.

  To address these drawbacks, we proposed an OSE approach that extend the LSMDS algorithm utilising the embedding of only a subset of the given data. We present two OSE methods: the first based on an optimisation approach and the second based on a neural network model. We applied both methods to improve the efficiency of the Em-K indexing method. The results suggested that the NN model is faster than the optimisation approach. Further, we explained how to modify the LSMDS algorithm to solve various ER problems efficiently using the proposed OSE methods. We discussed our OSE methods and the improvements of the Em-K indexing in detail in Chapter 6.

## 7.2 Future Directions

Next, we discuss several possible future works of the methodologies proposed in this thesis.

- Culturally diverse synthetic data: One of the main objectives of the name approximation approach and the name-like vector simulator is removing cultural biases from name data. We attempted to avoid this bias using approximate vectors to represent names in our simulation model. However, obtaining unbiased data for our analysis is difficult since many publicly available actual datasets of names are biased towards American and European names. Hence, we expect to investigate this problem in future work to extend the model construction based on culturally diverse data and possibly artificially created unbiased datasets.

  In our name approximation approach, the main property of interest is the dissimilarities or the distances between names. One aspect of our name approximation is to provide a universal representation for entity names by approximating them as vectors. Hence, we are not bound to any culture or character set of the data we used to test our algorithms. Accordingly, our algorithms are not limited to certain types of data. However, as we discussed, testing this thoroughly is challenging due to bias in available datasets.

- Similarity measures for different character sets: We want to construct and test our simulation model based on a diverse set of names that are not biased towards English names with Roman characters, e.g., Chinese, Indian, or Arabic names. However, there are extensive difficulties in Asian or Arabic names. For instance, the character set differs for these datasets; hence many string dissimilarity measures become less meaningful or have no meaning.

  Many string comparison algorithms are alphabet-based, and they cannot naively handle similarity measurements in non-Latin alphabets such as Indian or Arabic. Moreover, Chinese characters do not perform as an alphabet as they are ideographs (a written symbol representing the meaning directly) [210]. There is an inherent problem of incorporating datasets that are not based on Latin characters in ER. Hence, developing ER techniques for entity records based on non-Latin characters is challenging due to the lack of suitable similarity measures. With appropriate alphabet-based similarity measures for non-Latin entity names, we can extend our name-like simulator to generate unbiased data.

- Global matching: We are interested in extending this work to solve the global matching problem for large-scale datasets. The idea is to avoid comparisons of unnecessary detail in the pairwise comparisons (global matching) and make the ER process more scalable for big data. We are interested in extending this work to solve the global matching problem for large-scale datasets. The idea is to avoid comparisons of unnecessary detail in the pairwise comparisons (global matching) and make the ER process more scalable for big data. We define global matching as the problem of finding similar entity records from large-scale datasets without performing all the detailed-level pairwise similarities. We would pick the matching entities globally, like in holistic approaches [107]. Hence, the simulation of the many details of identification keys is not required for test data when considering the global matching problem. The proposed name-like vector simulation model is one step towards being able to generate large simulated datasets for large-scale testing of global matching techniques.

- Query matching against dynamic databases: Our Em-K indexing method queries against a large-scale reference database, and we assumed that the reference database is static. However, reference databases grow with time in real-world ER applications, and the new entities need to be added incrementally. For example, some applications would require the addition of records corresponding to queries with no matching records in the reference database. To facilitate this, we need to extend the Kd-tree accordingly without repeating the embedding process that creates the initial tree. However, growing an existing Kd-tree can lead to the tree becoming unbalanced. Therefore, we should explore alternative tree structures such as the R-tree, which is robust against dynamic data. Then the current Em-K indexing can be extended

to other ER problems, such as querying against a dynamic database, iterative ER, and incremental ER.

- Improving matching quality for data sets with many errors: Real-world datasets can include errors, missing values, or incomplete values. These errors and other inconsistencies in data create many data quality issues that reduce the matching quality in ER algorithms. In our experiments, we have only considered the data with typographical errors. The results show that many typographical errors in the data reduce the quality of the results produced by the Em-K indexing algorithm. However, it remains to investigate how other data quality issues such as OCR errors, phonetic errors, incomplete values, and values that changed over time (e.g., addresses) would affect the algorithm and the overall matching quality. It is also important to explore how we can improve the quality of the query matching for datasets with poor data quality.

- Non-name data: In this work, we focus on names, but matching can involve other types of data, e.g., addresses, gender, date of birth, occupation. Accordingly, we might need to increase the dimension, number of landmarks, etc., and modify other settings of the proposed methods.

## 7.3 Closing Remarks

This thesis provides insights into the importance of ER and mainly focus on the problems with large-scale ER. We proposed solutions to address the issues in the scarcity of test data and the demand for fast indexing algorithms for large-scale ER. We believe that the proposed techniques can be used in real-world scenarios where large-scale ER testing is required.

# Bibliography


[1] P. Christen, *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Berlin Heidelberg, 2012, pp. 3–22. [Online]. Available: https://doi.org/10.1007/978-3-642-31164-2_1

[2] X. Chen, "Towards efficient and effective entity resolution for high-volume and variable data," Ph.D. dissertation, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, 2020. [Online]. Available: http://dx.doi.org/10.25673/35204

[3] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[4] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969.

[5] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: a survey," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, no. 1, p. 1–16, Jan. 2007.

[6] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis, "An overview of end-to-end entity resolution for big data," *ACM Comput. Surv.*, vol. 53, no. 6, Dec. 2020. [Online]. Available: https://doi.org/10.1145/3418896

[7] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James, "Automatic linkage of vital records." *Science (New York, N.Y.)*, vol. 130, pp. 954–959, 1959.

[8] T. Dalenius, "Finding a needle in a haystack or identifying anonymous census records," *Journal of Official Statistics*, vol. 2, no. 3, p. 329, 09 1986.

[9] F. Panse, , A. Düjon, W. Wingerath, and B. Wollmer, "Generating realistic test datasets for duplicate detection at scale using historical voter data." in *Proceedings of the 24th International Conference on Extending Database technology*, 2021, p. 570–581.

[10] C. Chen, D. Pullen, R. Petty, and J. Talburt, "Methodology for large-scale entity resolution without pairwise matching," 11 2015, pp. 204–210.

[11] A. Gruenheid, X. L. Dong, and D. Srivastava, "Incremental record linkage," *Proc. VLDB Endow.*, vol. 7, no. 9, p. 697–708, May 2014.

[12] H. Altwaijry, S. Mehrotra, and D. V. Kalashnikov, "QuERy: a framework for integrating entity resolution with query processing," *Proc. VLDB Endow.*, vol. 9, no. 3, p. 120–131, Nov. 2015.

[13] P. Christen and R. W. Gayler, "Adaptive temporal entity resolution on dynamic databases," in *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 558–569.

[14] P. Christen, "A comparison of personal name matching: techniques and practical issues," in *Sixth IEEE ICDM-Workshops (ICDMW'06)*. IEEE, 2006, pp. 290–294.

[15] N. Adly, "Efficient record linkage using a double embedding scheme," in *DMIN*, vol. 48, 05 2009, pp. 274–281.

[16] C. Li, L. Jin, and S. Mehrotra, "Supporting efficient record linkage for large data sets using mapping techniques," *World Wide Web*, vol. 9, pp. 557–584, 12 2006.

[17] M. A. A. Cox and T. F. Cox, *Multidimensional Scaling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[18] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. Whang, and J. Widom, "Swoosh: a generic approach to entity resolution," *VLDB J.*, vol. 18, pp. 255–276, 01 2009.

[19] A. Adelstein, "Record linkage techniques in studies of the ætiology of cancer," *Proceedings of the Royal Society of Medicine*, vol. 61, no. 7, p. 732, July 1968. [Online]. Available: https://europepmc.org/articles/PMC1902698

[20] E. D. Acheson and J. G. Evans, "The oxford record linkage study: a review of the method with some preliminary results," *Proceedings of the Royal Society of Medicine*, vol. 57, no. 4, pp. 269–274, 1964.

[21] D. Clark, "Practical introduction to record linkage for injury research," *Injury prevention : journal of the International Society for Child and Adolescent Injury Prevention*, vol. 10, pp. 186–91, 07 2004.

[22] M. Tromp, A. Ravelli, G. Bonsel, A. Hasman, and J. Reitsma, "Results from simulated data sets: probabilistic record linkage outperforms deterministic record linkage," *Journal of clinical epidemiology*, vol. 64, pp. 565–72, 10 2010.

[23] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," in *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '95. New York, NY, USA: Association for Computing Machinery, 1995, p. 127–138. [Online]. Available: https://doi.org/10.1145/223784.223807

[24] Q. He, Z. Li, and X. Zhang, "Data deduplication techniques," *2010 International Conference on Future Information Technology and Management Engineering*, vol. 1, pp. 430–433, 2010.

[25] D. Vatsalan, P. Christen, and V. S. Verykios, "A taxonomy of privacy-preserving record linkage techniques," *Information Systems*, vol. 38, no. 6, pp. 946 – 969, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306437912001470

[26] C.-L. Cheng, Y.-H. Y. Kao, S.-J. Lin, C.-H. Lee, and M. L. Lai, "Validation of the national health insurance research database with ischemic stroke cases in Taiwan," *Pharmacoepidemiology and Drug Safety*, vol. 20, no. 3, pp. 236–242, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/pds.2087

[27] E. L. Brook, D. L. Rosman, and C. D. J. Holman, "Public good through data linkage: measuring research outputs from the Western Australian Data Linkage System," *Australian and New Zealand Journal of Public Health*, vol. 32, no. 1, pp. 19–23, 2008.

[28] C. J. Bradley, L. Penberthy, K. J. Devers, and D. J. Holden, "Health services research and data linkages: issues, methods, and directions for the future," *Health Services Research*, vol. 45, no. 5p2, pp. 1468–1488, 2010.

[29] C. D. J. Holman, A. J. Bass, I. L. Rouse, and M. S. Hobbs, "Population-based linkage of health records in Western Australia: development of a health services research linked database," *Australian and New Zealand Journal of Public Health*, vol. 23, no. 5, pp. 453–459, 1999.

[30] "Population health research network program office. annual review 2010- 2011. technical report," 2011. [Online]. Available: https://www.phrn.org.au/media/77248/phrn%20overview.pdf

[31] K. Irvine, R. Hall, and L. Taylor, "Centre for health record linkage: expanding access to linked population data for NSW and the ACT, Australia," *International Journal of Population Data Science*, vol. 4, Oct. 2020.

[32] C. Phua, K. Smith-Miles, V. Lee, and R. Gayler, "Resilient identity crime detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 3, pp. 533–546, 2012.

[33] H. Chen, W. Chung, J. J. Xu, G. Wang, Y. Qin, and M. Chau, "Crime data mining: a general framework and some examples," *Computer*, vol. 37, no. 4, pp. 50–56, 2004.

[34] G. Wang, J. Xu, and H. Atabakhsh, "Automatically detecting criminal identity deception: an adaptive detection algorithm," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 36, pp. 988 – 999, 10 2006.

[35] J. Li, G. Wang, and H.-c. Chen, "Identity matching using personal and social identity features," *Information Systems Frontiers*, vol. 13, pp. 101–113, 03 2011.

[36] J. Li and G. Wang, "A framework of identity resolution: evaluating identity attributes and matching algorithms," *Security Informatics*, vol. 4, pp. 1–12, 12 2015.

[37] J. Talburt and C.-C. Chiang, "Attributed identity resolution for fraud detection and prevention," in *2009 International Conference on Computing, Engineering and Information*, 2009, pp. 97–99.

[38] W. E. Winkler, "Overview of record linkage and current research directions," BUREAU OF THE CENSUS, Tech. Rep., 2006.

[39] W. Winkler, "Matching and record linkage," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, 09 2014.

[40] A. Aizawa and K. Oyama, "A fast linkage detection scheme for multi-source information integration," in *Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration*, ser. WIRI '05. USA: IEEE Computer Society, 2005, p. 30–39.

[41] C. Giles, K. Bollacker, and S. Lawrence, "Citeseer: an automatic citation indexing system," *Proceedings of 3rd ACM Conference on Digital Libraries*, 04 2000.

[42] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 01 2012.

[43] C. Batini and M. Scannapieco, *Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[44] T. Herzog, F. Scheuren, and W. Winkler, *Data Quality and Record Linkage*, 01 2007.

[45] J. Talburt, *Principles of Entity Resolution*, 12 2011, pp. 1–37.

[46] P. Christen, T. Ranbaduge, and R. Schnell, *Linking Sensitive Data: Methods and Techniques for Practical Privacy-Preserving Information Sharing*. Cham: Springer International Publishing, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-59706-1_1

[47] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *PVLDB*, vol. 3, pp. 484–493, 09 2010.

[48] K. M. Campbell, D. Deck, and A. Krupski, "Record linkage software in the public domain: a comparison of link plus, the link king, and a 'basic' deterministic algorithm," *Health Informatics Journal*, vol. 14, no. 1, pp. 5–15, 2008, pMID: 18258671. [Online]. Available: https://doi.org/10.1177/1460458208088855

[49] A. Ferrante and J. Boyd, "A transparent and transportable methodology for evaluating data linkage software," *Journal of Biomedical Informatics*, vol. 45, no. 1, pp. 165–172, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046411001729

[50] P. Christen and T. Churches, "FEBRL - Freely extensible biomedical record linkage," Tech. Rep., 2002.

[51] M. Sariyar and A. Borg, "The recordlinkage package: detecting errors in data," *The R Journal*, vol. 2, no. 2, pp. 61–67, 2010. [Online]. Available: https://doi.org/10.32614/RJ-2010-017

[52] M. Elfeky, V. Verykios, and A. Elmagarmid, "Tailor: a record linkage toolbox," in *Proceedings 18th International Conference on Data Engineering*, 2002, pp. 17–28.

[53] P. Konda, S. Das, P. Suganthan G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra, "Magellan: toward building entity matching management systems," *Proc. VLDB Endow.*, vol. 9, no. 12, p. 1197–1208, aug 2016. [Online]. Available: https://doi.org/10.14778/2994509.2994535

[54] M. S. Ali, M. Y. Ichihara, L. C. Lopes, G. C. Barbosa, R. Pita, R. P. Carreiro, D. B. dos Santos, D. Ramos, N. Bispo, F. Raynal, V. Canuto, B. de Araujo Almeida, R. L. Fiaccone, M. E. Barreto, L. Smeeth, and M. L. Barreto, "Administrative data linkage in Brazil: potentials for health technology assessment," *Frontiers in Pharmacology*, vol. 10, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphar.2019.00984

[55] D. Karapiperis, A. Gkoulalas-Divanis, and V. S. Verykios, "Femrl: A framework for large-scale privacy-preserving linkage of patients' electronic health records," in *2018 IEEE International Smart Cities Conference (ISC2)*, 2018, pp. 1–8.

[56] M. Franke, Z. Sehili, and E. Rahm, "Primat: a toolbox for fast privacy-preserving matching," *Proc. VLDB Endow.*, vol. 12, no. 12, p. 1826–1829, aug 2019. [Online]. Available: https://doi.org/10.14778/3352063.3352076

[57] P. Christen, "Privacy-Preserving data linkage and geocoding: current approaches and research directions," *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, pp. 497–501, 2006.

[58] P. Christen, R. Schnell, D. Vatsalan, and T. Ranbaduge, "Efficient cryptanalysis of bloom filters for privacy-preserving record linkage," in *Advances in Knowledge Discovery and Data Mining*, J. Kim, K. Shim, L. Cao, J.-G. Lee, X. Lin, and Y.-S. Moon, Eds. Cham: Springer International Publishing, 2017, pp. 628–640.

[59] A. Gkoulalas-Divanis, D. Vatsalan, D. Karapiperis, and M. Kantarcioglu, "Modern privacy-preserving record linkage techniques: An overview," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4966–4987, 2021.

[60] B. Ramadan, "Indexing techniques for real-time entity resolution," Ph.D. dissertation, College of Engineering & Computer Science, Australian National University, 2016.

[61] N. Barlaug and J. A. Gulla, "Neural networks for entity matching: a survey," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 3, Apr. 2021.

[62] Y. Zhang, K. S. Ng, T. Churchill, and P. Christen, "Scalable entity resolution using probabilistic signatures on parallel databases," ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3269206.3272016

[63] F. Naumann and M. Herschel, *An introduction to duplicate detection.* Morgan and Claypool Publishers, 2010.

[64] H.-s. Kim and D. Lee, "HARRA: Fast iterative hashed record linkage for large-scale data collections," in *Proceedings of the 13th International Conference on Extending Database Technology*, ser. EDBT '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 525–536. [Online]. Available: https://doi.org/10.1145/1739041.1739104

[65] W. E. W. Edward H. Porter, "Approximate string comparison and its effect on an advanced record linkage system," *Advanced Record Linkage System. U.S. Bureau of the Census, Research Report*, pp. 190–199, 1997.

[66] E. Rahm and H. H. Do, "Data cleaning: problems and current approaches," *IEEE Bulletin of the Technical Committee on Data Engineering*, vol. 23, pp. 3–13, 01 2000.

[67] P. Christen and K. Goiser, "Quality and complexity measures for data linkage and deduplication," *Studies in Computational Intelligence*, vol. 43, 01 2007.

[68] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas, "Blocking and filtering techniques for entity resolution: a survey," *ACM Comput. Surv.*, vol. 53, no. 2, Mar. 2020. [Online]. Available: https://doi.org/10.1145/3377455

[69] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 9, pp. 1537–1555, 2012.

[70] T. de Vries, H. Ke, S. Chawla, and P. Christen, "Robust record linkage blocking using suffix arrays and bloom filters," *ACM Trans. Knowl. Discov. Data*, vol. 5, no. 2, Feb. 2011. [Online]. Available: https://doi.org/10.1145/1921632.1921635

[71] V. Vykhovanets, J. Du, and S. Sakulin, "An overview of phonetic encoding algorithms," *Automation and Remote Control*, vol. 81, pp. 1896–1910, 10 2020.

[72] D. Holmes and M. McCabe, "Improving precision and recall for soundex retrieval," in *Proceedings. International Conference on Information Technology: Coding and Computing*, 2002, pp. 22–26.

[73] L. Philips, "The double metaphone search algorithm," *C/C++ Users Journal*, vol. 18, pp. 38–43, 06 2000.

[74] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, no. 1, p. 31–88, Mar. 2001. [Online]. Available: https://doi.org/10.1145/375360.375365

[75] M. P. J. V. D. Loo, "The STRINGDIST package for approximate string matching," *The R Journal*, vol. 6, no. 1, pp. 111–122, 2014.

[76] C.Chatfied and A.J.Collins, *Introduction to multivariate analysis*, 1980. [Online]. Available: https://www.springer.com/gp/book/9780412160301

[77] X. Chen, E. Schallehn, and G. Saake, "Cloud-scale entity resolution: current state and open challenges," *Open J. Big Data*, vol. 4, pp. 30–51, 2018.

[78] E. Sauleau, J.-P. Paumier, and A. Buemi, "Medical record linkage in health information systems by approximate string matching and clustering," *BMC medical informatics and decision making*, vol. 5, p. 32, 02 2005.

[79] W. Yancey, "Evaluating string comparator performance for record linkage," *United States Census Bureau*, 01 2005.

[80] R. Baxter, P. Christen, and T. Churches, "A comparison of fast blocking methods for record linkage," *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop*, pp. 25–27, 2003.

[81] M. A.Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, Florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, 1989. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1989.10478785

[82] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string metrics for matching names and records," *KDD Workshop on Data Cleaning and Object Consolidation*, vol. 3, pp. 73–78, 2003.

[83] T. Ho, S.-R. Oh, and H. Kim, "New algorithms for fixed-length approximate string matching and approximate circular string matching under the hamming distance," *The Journal of Supercomputing*, vol. 74, 05 2018.

[84] L. Gu and R. Baxter, *Decision Models for Record Linkage*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 146–160. [Online]. Available: https://doi.org/10.1007/11677437_12

[85] K. Goiser and P. Christen, "Towards automated record linkage," ser. AusDM '06. AUS: Australian Computer Society, Inc., 2006, p. 23–31.

[86] P. Christen, M. Hegland, S. Roberts, O. Nielsen, T. Churches, K. Lim, and S. Branch, "Parallel computing techniques for high-performance probabilistic record linkage," 04 2002.

[87] T. Victor and R. Mera, "Record linkage of healthcare insurance claims," *Studies in health technology and informatics*, vol. 84, pp. 1409–13, 02 2001.

[88] K. Harron, C. Dibben, J. Boyd, A. Hjern, M. Azimaee, M. L. Barreto, and H. Goldstein, "Challenges in administrative data linkage for research," *Big Data & Society*, vol. 4, no. 2, p. 2053951717745678, 2017, pMID: 30381794. [Online]. Available: https://doi.org/10.1177/2053951717745678

[89] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," *ACM Sigmod Record*, vol. 24, no. 2, pp. 127–138, 1995.

[90] W. Fan, X. Jia, J. Li, and S. Ma, "Reasoning about record matching rules," *Proc. VLDB Endow.*, vol. 2, no. 1, p. 407–418, Aug. 2009. [Online]. Available: https://doi.org/10.14778/1687627.1687674

[91] L. Li, J. Li, and H. Gao, "Rule-based method for entity resolution," *IEEE Transactions on Knowledge & Data Engineering*, vol. 27, no. 01, pp. 250–263, jan 2015.

[92] H. Abu Ahmad and H. Wang, "An effective weighted rule-based method for entity resolution," *Distributed and Parallel Databases*, vol. 36, 09 2018.

[93] P. Christen, "Automatic record linkage using seeded nearest neighbour and support vector machine classification," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 151–159. [Online]. Available: https://doi.org/10.1145/1401890.1401913

[94] O. F. Reyes-Galaviz, W. Pedrycz, Z. He, and N. J. Pizzi, "A supervised gradient-based learning algorithm for optimized entity resolution," *Data & Knowledge Engineering*, vol. 112, pp. 106 – 129, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169023X16302038

[95] W. W. Cohen and J. Richman, "Learning to match and cluster large high-dimensional data sets for data integration," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 475–480. [Online]. Available: https://doi.org/10.1145/775047.775116

[96] A. Arasu, M. Götz, and R. Kaushik, "On active learning of record matching packages," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 783–794. [Online]. Available: https://doi.org/10.1145/1807167.1807252

[97] Z. Wang, B. Sisman, H. Wei, X. L. Dong, and S. Ji, "CorDEL: A contrastive deep learning approach for entity linkage," 2020.

[98] I. Bhattacharya and L. Getoor, "Collective entity resolution in relational data," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, p. 5–es, Mar. 2007. [Online]. Available: https://doi.org/10.1145/1217299.1217304

[99] M. Herschel, F. Naumann, S. Szott, and M. Taubert, "Scalable iterative graph duplicate detection," *IEEE Trans. on Knowl. and Data Eng.*, vol. 24, no. 11, p. 2094–2108, Nov. 2012. [Online]. Available: https://doi.org/10.1109/TKDE.2011.99

[100] M. Sumon Shahriar, "Application of machine learning to streamline clerical review in data linkage," *International Journal of Population Data Science*, vol. 5, no. 5, Dec. 2020. [Online]. Available: https://ijpds.org/article/view/1571

[101] T. Brasileiro Araújo, K. Stefanidis, C. E. Santos Pires, J. Nummenmaa, and T. Pereira da Nóbrega, "Incremental blocking for entity resolution over web streaming data," in *IEEE/WIC/ACM International Conference on Web Intelligence*, ser. WI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 332–336. [Online]. Available: https://doi.org/10.1145/3350546.3352542

[102] B. Ramadan, P. Christen, H. Liang, and R. W. Gayler, "Dynamic sorted neighborhood indexing for real-time entity resolution," *J. Data and Information Quality*, vol. 6, no. 4, oct 2015. [Online]. Available: https://doi.org/10.1145/2816821

[103] I. Bhattacharya, L. Getoor, and L. Licamele, "Query-time entity resolution," ser. KDD '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 529–534. [Online]. Available: https://doi.org/10.1145/1150402.1150463

[104] E. Ioannou, W. Nejdl, C. Niederée, and Y. Velegrakis, "On-the-fly entity-aware query processing in the presence of linkage," *Proc. VLDB Endow.*, vol. 3, no. 1–2, p. 429–438, Sep. 2010.

[105] B. Y. Kun Ma, "Stream-based live entity resolution approach with adaptive duplicate count strategy," *Int. J. Web Grid Serv.*, vol. 13, no. 3, p. 351–373, Jan. 2017. [Online]. Available: https://doi.org/10.1504/IJWGS.2017.085167

[106] T. B. Araújo, K. Stefanidis, C. E. Santos Pires, J. Nummenmaa, and T. P. da Nóbrega, "Schema-agnostic blocking for streaming data," ser. SAC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 412–419.

[107] E. Rahm, "The case for holistic data integration," in *Advances in Databases and Information Systems*, J. Pokorný, M. Ivanović, B. Thalheim, and P. Šaloun, Eds. Cham: Springer International Publishing, 2016, pp. 11–27.

[108] L. Kolb, A. Thor, and E. Rahm, "Multi-pass sorted neighborhood blocking with mapreduce," *Comput. Sci.*, vol. 27, no. 1, p. 45–63, Feb. 2012. [Online]. Available: https://doi.org/10.1007/s00450-011-0177-x

[109] A. Saeedi, E. Peukert, and E. Rahm, "Comparative evaluation of distributed clustering schemes for multi-source entity resolution," in *Advances in Databases and Information Systems*. Cham: Springer International Publishing, 2017, pp. 278–293.

[110] ——, "Using link features for entity clustering in knowledge graphs," in *The Semantic Web*. Cham: Springer International Publishing, 2018, pp. 576–592.

[111] W. McNeill, H. Kardes, and A. Borthwick, "Dynamic record blocking: efficient linking of massive databases in MapReduce," in *QDB*, 2012.

[112] A. Aizawa and K. Oyama, "A fast linkage detection scheme for multi-source information integration," ser. WIRI '05. USA: IEEE Computer Society, 2005, p. 30–39.

[113] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 169–178. [Online]. Available: https://doi.org/10.1145/347090.347123

[114] B. Kenig and A. Gal, "MFIBlocks: an effective blocking algorithm for entity resolution," *Information Systems*, vol. 38, p. 908–926, 09 2013.

[115] B. Ramadan and P. Christen, "Unsupervised blocking key selection for real-time entity resolution," in *Advances in Knowledge Discovery and Data Mining - 19th Pacific-Asia Conference, PAKDD 2015*, vol. 9078. Springer, 2015, pp. 574–585. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-18032-8_45

[116] G. Papadakis, E. Ioannou, C. Niederée, and P. Fankhauser, "Efficient entity resolution for large heterogeneous information spaces," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, ser. WSDM '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 535–544. [Online]. Available: https://doi.org/10.1145/1935826.1935903

[117] V. Efthymiou, G. Papadakis, G. Papastefanatos, K. Stefanidis, and T. Palpanas, "Parallel meta-blocking for scaling entity resolution over big heterogeneous data," *Inf. Syst.*, vol. 65, no. C, p. 137–157, Apr. 2017.

[118] L. Kolb, A. Thor, and E. Rahm, "Dedoop: efficient deduplication with hadoop," *Proc. VLDB Endow.*, vol. 5, no. 12, p. 1878–1881, Aug. 2012. [Online]. Available: https://doi.org/10.14778/2367502.2367527

[119] B. Gufler, N. Augsten, A. Reiser, and A. Kemper, "Handling data skew in mapreduce." in *CLOSER 2011 - Proceedings of the 1st International Conference on Cloud Computing and Services Science*, 01 2011, pp. 574–583.

[120] M. Nentwig, A. Groß, and E. Rahm, "Holistic entity clustering for linked data," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 2016, pp. 194–201.

[121] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Foundations of metric space searching.* Boston, MA: Springer US, 2006, pp. 5–66.

[122] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, Mar 1964.

[123] N. Saeed, H. Nam, M. I. U. Haq, and D. B. Muhammad Saqib, "A survey on multidimensional scaling," *ACM Comput. Surv.*, vol. 51, no. 3, May 2018. [Online]. Available: https://doi.org/10.1145/3178155

[124] M. C. Hout, M. H. Papesh, and S. D. Goldinger, "Multidimensional scaling," *WIREs Cognitive Science*, vol. 4, no. 1, pp. 93–103, 2013.

[125] P. Groenen and I. Borg, "The past, present, and future of multidimensional scaling," Erasmus University Rotterdam, Erasmus School of Economics (ESE), Econometric Institute, Econometric Institute Research Papers EI 2013-07, Jan. 2013. [Online]. Available: https://ideas.repec.org/p/ems/eureir/39177.html

[126] J. B. Kruskal and M. Wish, *Quantitative applications in the social sciences: multidimensional scaling*, 1978. [Online]. Available: http://methods.sagepub.com/book/multidimensional-scaling

[127] S. France and J. Carroll, "Two-way multidimensional scaling: a review," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, pp. 644 – 661, 10 2011.

[128] A. Ghodsi, "Dimensionality reduction a short tutorial," *Department of Statistics and Actuarial Science, University of Waterloo, Ontario*, p. 38, 01 2006.

[129] J. De Leeuw, *Classical Scaling.* New York, NY: Springer New York, 2005, pp. 261–267. [Online]. Available: https://doi.org/10.1007/0-387-28981-X_12

[130] V. Silva and J. Tenenbaum, "Sparse multidimensional scaling using landmark points," *Technical Report, Department of Mathematics, Stanford University.*, 01 2004.

[131] S.-H. Bae, J. Y. Choi, J. Qiu, and G. C. Fox, "Dimension reduction and visualization of large high-dimensional data via interpolation," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 203–214.

[132] J. de Leeuw and P. Mair, "Multidimensional scaling using majorization: SMACOF in R," *Journal of Statistical Software*, vol. 31, no. 3, pp. 1–30, 2009.

[133] M. W. Trosset and C. E. Priebe, "The out-of-sample problem for classical multidimensional scaling," *Computational Statistics and Data Analysis*, vol. 52, no. 10, pp. 4635–4642, 2008.

[134] V. d. Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, ser. NIPS'02. Cambridge, MA, USA: MIT Press, 2002, p. 721–728.

[135] L. Tang and M. Crovella, "Virtual landmarks for the internet," ser. IMC '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 143–152.

[136] S. Lee and S. Choi, "Landmark mds ensemble," *Pattern Recognition*, vol. 42, no. 9, pp. 2045–2053, 2009.

[137] C. Faloutsos and K.-I. Lin, "Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '95. New York, NY, USA: Association for Computing Machinery, 1995, p. 163–174.

[138] J. T.-L. Wang, X. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro, and K. Zhang, "Evaluating a class of distance-mapping algorithms for data mining and clustering," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 307–311.

[139] J. Platt, "Fastmap, metricmap, and landmark mds are all nyström algorithms," *AISTATS 2005 - Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 10 2004.

[140] P. Kamousi, S. Lazard, A. Maheshwari, and S. Wuhrer, "Analysis of farthest point sampling for approximating geodesics in a graph," *Computational Geometry*, vol. 57, pp. 1–7, 2016.

[141] G. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces," *ACM Transactions on Database Systems (TODS)*, vol. 28, pp. 517–580, 12 2003.

[142] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, "Searching in metric spaces," *ACM Comput. Surv.*, vol. 33, no. 3, p. 273–321, Sep. 2001.

[143] P. Zezula, M. Batko, and V. Dohnal, *Indexing metric spaces.* Boston, MA: Springer US, 2009, pp. 1451–1454.

[144] M. Houle and J. Sakuma, "Fast approximate similarity search in extremely high-dimensional data sets," in *21st International Conference on Data Engineering (ICDE'05)*, 2005, pp. 619–630.

[145] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[146] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, 1990.

[147] P. N. Yianilos, "Data structures and algorithms for nearest neighbor search in general metric spaces," in *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '93. USA: Society for Industrial and Applied Mathematics, 1993, p. 311–321.

[148] S. Sahinalp, M. Tasan, J. Macker, and Z. Ozsoyoglu, "Distance based indexing for string proximity search," in *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, 2003, pp. 125–136.

[149] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '84. New York, NY, USA: Association for Computing Machinery, 1984, p. 47–57.

[150] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, p. 209–226, Sep. 1977.

[151] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: an efficient eccess method for similarity search in metric spaces," in *Proceedings of the 23rd International Conference on Very Large Data Bases*, ser. VLDB '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, p. 426–435.

[152] P. Indyk and R. Motwani, "Approximate nearest neighbours: towards removing the curse ofddimensionality," *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, vol. 604-613, 10 2000.

[153] F. Turrado García, L. J. García Villalba, A. L. Sandoval Orozco, F. D. Aranda Ruiz, A. Aguirre Juárez, and T.-H. Kim, "Locating similar names through locality sensitive hashing and graph theory," *Multimedia Tools and Applications*, vol. 78, 2019.

[154] M. Cochez, "Locality-sensitive hashing for massive string-based ontology matching," in *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 1, 2014, pp. 134–140.

[155] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, p. 509–517, Sep. 1975.

[156] C. Eastman and S. Weiss, "Tree structures for high dimensionality nearest neighbor searching," *Information Systems*, vol. 7, no. 2, pp. 115–122, 1982.

[157] B. S. Kim and S. B. Park, "A fast k nearest neighbor finding algorithm based on the ordered partition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 761–766, 1986.

[158] A. J. Broder, "Strategies for efficient incremental nearest neighbor search," *Pattern Recognit.*, vol. 23, pp. 171–178, 1990.

[159] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *J. ACM*, vol. 45, no. 6, p. 891–923, Nov. 1998.

[160] D. Talbert and D. Fisher, "An empirical analysis of techniques for constructing and searching k-dimensional trees," 01 2000, pp. 26–33.

[161] C. C. Aggarwal and P. S. Yu, "The igrid index: reversing the dimensionality curse for similarity indexing in high dimensionalsspace," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '00. New York, NY, USA: Association for Computing Machinery, 2000, p. 119–129.

[162] J.-M. Herrera, A. Hogan, and T. Käfer, "BTC-2019: the 2019 billion triple challenge dataset," in *International Semantic Web Conference*. Springer, 2019, pp. 163–180.

[163] M. Arehart and K. J. Miller, "A ground truth dataset for matching culturally diverse romanized person names," in *LREC*, 2008.

[164] P. Christen and D. Vatsalan, "Flexible and extensible generation and corruption of personal data," in *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, ser. CIKM 13. New York, NY, USA: Association for Computing Machinery, 2013, p. 1165–1168. [Online]. Available: https://doi.org/10.1145/2505515.2507815

[165] C. for Epidemiology and Research, "NSW department of health. New South Wales mothers and babies 2002," *NSW Public Health Bull*, vol. 14, pp. s–3, 2002.

[166] P. Christen, "Preparation of a real temporal voter data set for record linkage and duplicate detection research," *Technical Report, ANU*, 2014. [Online]. Available: http://users.cecs.anu.edu.au/~Peter.Christen/publications/ncvoter-report-29june2014.pdf

[167] P. Bertolazzi, L. Santis, and M. Scannapieco, "Automatic record matching in cooperative information systems," in *Proceedings of the ICDT'03 International Workshop on Data Quality in Cooperative Information Systems (DQCIS'03)*, 2003.

[168] J. R. Talburt, Y. Zhou, and S. Y. Shivaiah, "SOG: A synthetic occupancy generator to support entity resolution instruction and research," *ICIQ*, vol. 9, pp. 91–105, 2009.

[169] M. Tromp, A. C. Ravelli, G. J. Bonsel, A. Hasman, and J. B. Reitsma, "Results from simulated data sets: probabilistic record linkage outperforms deterministic record linkage," *Journal of Clinical Epidemiology*, vol. 64, no. 5, pp. 565–572, 2011.

[170] T. Bachteler and J. Reiher, "Tdgen: a test data generator for evaluating record linkage methods," *NO. WP-GRLC-2012-01*, 2012.

[171] P. Christen, "FEBRL: an open source data cleaning, deduplication and record linkage system with a graphical user interface," *Proceeding of the 14th ACM SIGKDD*, pp. 1065–1068, 2008.

[172] S. Katharina Sienčnik, "Adapting word2vec to named entity recognition," in *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*. Linköping University Electronic Press, Sweden, May 2015, pp. 239–243.

[173] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[174] A. Mazeika and M. H. Böhlen, "Cleansing databases of misspelled proper nouns," in *In CleanDB Workshop*, 2006, pp. 63–70.

[175] C. J. Mecklin and D. Mundfrom, "On using asymptotic critical values in testing for multivariate normality." InterStat, 2003.

[176] S. Korkmaz, D. Goksuluk, and G. Zararsiz, "MVN: an R package for assessing multivariate normality," *The R Journal*, vol. 6, no. 2, pp. 151–162, 2014.

[177] T. Burdenski, "Evaluating univariate, bivariate, and multivariate normality using graphical and statistical procedures," *Multiple Linear Regression Viewpoints*, vol. 26, pp. 15–28, 01 2000.

[178] P. Christen, "Probabilistic data generation for deduplication and data linkage," *Intelligent Data Engineering and Automated Learning*, pp. 101–107, 2005.

[179] F. Bookstein, "Comparing covariance matrices by relative eigenanalysis, with applications to organismal biology," *Evolutionary Biology*, vol. 41, no. 2, pp. 336–350, 2014.

[180] M. Cristelli, M. Batty, and L. Pietronero, "There is more than a power law in Zipf," *Scientific Reports*, vol. 2, p. 812, 11 2012.

[181] J. Sukharev, L. Zhukov, and A. Popescul, "Parallel corpus approach for name matching in record linkage," in *2014 IEEE ICDM*, 2014, pp. 995–1000.

[182] J. Comenetz, "Frequently occurring surnames in the 2010 census," Bureau of the census, USA, Tech. Rep., 2016. [Online]. Available: https://www.census.gov/topics/population/genealogy/data/2010_surnames.html

[183] W. Hoeffding, *A non-parametric test of independence*. New York, NY: Springer New York, 1994, pp. 214–226.

[184] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 9, pp. 1537–1555, 2012.

[185] G. Papadakis, E. Ioannou, C. Niederée, T. Palpanas, and W. Nejdl, "Eliminating the redundancy in blocking-based entity resolution methods," in *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*, ser. JCDL '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 85–94.

[186] G. Wang, H. Chen, and H. Atabakhsh, "Automatically detecting deceptive criminal identities," *Commun. ACM*, vol. 47, no. 3, p. 70–76, Mar. 2004. [Online]. Available: https://doi.org/10.1145/971617.971618

[187] P. Christen, *Further topics and research directions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 209–228. [Online]. Available: https://doi.org/10.1007/978-3-642-31164-2_9

[188] Ö. Akgün, A. Dearle, G. Kirby, and P. Christen, "Using metric space indexing for complete and efficient record linkage," in *Advances in Knowledge Discovery and Data Mining*. Cham: Springer International Publishing, 2018, pp. 89–101.

[189] Z. Sehili and E. Rahm, "Speeding up privacy preserving record linkage for metric space similarity measures," *Datenbank-Spektrum*, vol. 16, no. 3, pp. 227–236, 2016.

[190] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, 1989.

[191] U. Draisbach and F. Naumann, "A generalization of blocking and windowing algorithms for duplicate detection," in *Proceedings - 2011 International Conference on Data and Knowledge Engineering, ICDKE 2011*, 10 2011, pp. 18 – 24.

[192] A. Mazeika and M. Böhlen, "Cleansing databases of misspelled proper nouns," in *CleanDB, Seoul, Korea*, 09 2006.

[193] A. Saeedi, E. Peukert, and E. Rahm, "Comparative evaluation of distributed clustering schemes for multi-source entity resolution," in *Advances in Databases and Information Systems*. Cham: Springer International Publishing, 2017, pp. 278–293.

[194] P. Christen and K. Goiser, *Quality and complexity measures for data linkage and deduplication*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 127–151. [Online]. Available: https://doi.org/10.1007/978-3-540-44918-8_6

[195] H. Liang, Y. Wang, P. Christen, and R. Gayler, "Noise-tolerant approximate blocking for dynamic real-time entity resolution," in *Advances in Knowledge Discovery and Data Mining*, V. S. Tseng, T. B. Ho, Z.-H. Zhou, A. L. P. Chen, and H.-Y. Kao, Eds. Cham: Springer International Publishing, 2014, pp. 449–460.

[196] Y. Bengio, J. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet, "Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering," *Advances in Neural Information Processing Systems*, pp. 177–184, 2003.

[197] M. J. Anderson and J. Robinson, "Generalized discriminant analysis based on distances," *Australian & New Zealand Journal of Statistics*, vol. 45, no. 3, pp. 301–318, 2003. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-842X.00285

[198] J. C. Gower, "Adding a point to vector diagrams in multivariate analysis," *Biometrika*, vol. 55, no. 3, pp. 582–585, 1968. [Online]. Available: http://www.jstor.org/stable/2334268

[199] Q. Dong and Z. Wu, "A linear-time algorithm for solving the molecular distance geometry problem with exact inter-atomic distances," *Journal of Global Optimization*, vol. 22, no. 1, pp. 365–375, Jan 2002. [Online]. Available: https://doi.org/10.1023/A:1013857218127

[200] Y. Mito, M. A. M. Ismail, and T. Yamamoto, "Multidimensional scaling and inverse distance weighting transform for image processing of hydrogeological structure in rock mass," *Journal of Hydrology*, vol. 411, no. 1, pp. 25 – 36, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022169411006536

[201] Z. Wang, S. Zheng, Y. Ye, and S. Boyd, "Further relaxations of the semidefinite programming approach to sensor network localization," *SIAM Journal on Optimization*, vol. 19, pp. 655–673, 06 2008.

[202] T. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, 2002, pp. 170–179 vol.1.

[203] A. Jansen, G. Sell, and V. Lyzinski, "Scalable out-of-sample extension of graph embeddings using deep neural networks," *Pattern Recognition Letters*, vol. 94, pp. 1 – 6, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016786551730137X

[204] L. van der Maaten, "Learning a parametric embedding by preserving local structure," in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 5. PMLR, 16–18 Apr 2009, pp. 384–391. [Online]. Available: http://proceedings.mlr.press/v5/maaten09a.html

[205] K. Bunte, M. Biehl, and B. Hammer, "A general framework for dimensionality-reducing data visualization mapping," *Neural computation*, vol. 24, pp. 771–804, 12 2011.

[206] N. Ketkar, *Feed forward neural networks.* Berkeley, CA: Apress, 2017, pp. 17–33.

[207] B. Pang, E. Nijkamp, and Y. N. Wu, "Deep learning with tensorflow: A review," *Journal of Educational and Behavioral Statistics*, vol. 45, no. 2, pp. 227–248, 2020.

[208] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[209] D. E. Rumelhart and R. J. Hinton, G. E. & Williams, "Learning representations by back-propagating errors," *Nature (London)*, vol. 323, p. 533–536, 1986.

[210] S. Xu, M. Zheng, and X. Li, "String comparators for chinese-characters-based record linkages," *IEEE Access*, vol. 9, pp. 3735–3743, 2021.