The Four Generations of Entity Resolution

Synthesis Lectures on Data Management

Synthesis Lectures on Data Management is edited by H.V. Jagadish of the University of Michigan. The series publishes 80–150 page publications on topics pertaining to data management. Topics include query languages, database system architectures, transaction management, data warehousing, XML and databases, data stream systems, wide scale data distribution, multimedia data management, data mining, and related subjects.

Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments
Daniel C.M. de Oliveira, Ji Liu, and Esther Pacitti
2019

Answering Queries Using Views, Second Edition
Foto Afrati and Rada Chirkova
2019

Transaction Processing on Modern Hardware
Mohammad Sadoghi and Spyros Blanas
2019

Data Management in Machine Learning Systems
Matthias Boehm, Arun Kumar, and Jun Yang
2019

Non-Volatile Memory Database Management Systems
Joy Arulraj and Andrew Pavlo
2019

Scalable Processing of Spatial-Keyword Queries
Ahmed R. Mahmood and Walid G. Aref
2019

Data Exploration Using Example-Based Methods
Matteo Lissandrini, Davide Mottin, Themis Palpanas, and Yannis Velegrakis
2018

Data Profiling
Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock
2018

Querying Graphs
Angela Bonifati, George Fletcher, Hannes Voigt, and Nikolay Yakovets
2018

Query Processing over Incomplete Databases
Yunjun Gao and Xiaoye Miao
2018

Natural Language Data Management and Interfaces
Yunyao Li and Davood Rafiei
2018

Human Interaction with Graphs: A Visual Querying Perspective
Sourav S. Bhowmick, Byron Choi, and Chengkai Li
2018

Probabilistic Ranking Techniques in Relational Databases
Ihab F. Ilyas and Mohamed A. Soliman
2011

Uncertain Schema Matching
Avigdor Gal
2011

Fundamentals of Object Databases: Object-Oriented and Object-Relational Design
Suzanne W. Dietrich and Susan D. Urban
2010

Advanced Metasearch Engine Technology
Weiyi Meng and Clement T. Yu
2010

Web Page Recommendation Models: Theory and Algorithms
Sule Gündüz-Ögüdücü
2010

Multidimensional Databases and Data Warehousing
Christian S. Jensen, Torben Bach Pedersen, and Christian Thomsen
2010

Database Replication
Bettina Kemme, Ricardo Jimenez-Peris, and Marta Patino-Martinez
2010

Relational and XML Data Exchange
Marcelo Arenas, Pablo Barcelo, Leonid Libkin, and Filip Murlak
2010

User-Centered Data Management
Tiziana Catarci, Alan Dix, Stephen Kimani, and Giuseppe Santucci
2010

Data Stream Management
Lukasz Golab and M. Tamer Özsu
2010

Access Control in Data Management Systems
Elena Ferrari
2010

An Introduction to Duplicate Detection
Felix Naumann and Melanie Herschel
2010

The Four Generations of Entity Resolution

George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas

www.morganclaypool.com

ISBN: 9781636390567 paperback
ISBN: 9781636390574 ebook
ISBN: 9781636390581 hardcover

DOI 10.2200/S01067ED1V01Y202012DTM064

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON DATA MANAGEMENT

Lecture #65
Series Editor: H.V. Jagadish, *University of Michigan*
Founding Editor: M. Tamer Özsu, *University of Waterloo*
Series ISSN
Print 2153-5418 Electronic 2153-5426

The Four Generations of Entity Resolution

George Papadakis
National and Kapodistrian University of Athens, Greece

Ekaterini Ioannou
Tilburg University, Netherlands

Emanouil Thanos
Katholieke Universiteit Leuven, Belgium

Themis Palpanas
University of Paris, France & French University Institute (IUF), France

SYNTHESIS LECTURES ON DATA MANAGEMENT #65

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

Entity Resolution (ER) lies at the core of data integration and cleaning and, thus, a bulk of the research examines ways for improving its effectiveness and time efficiency. The initial ER methods primarily target Veracity in the context of structured (relational) data that are described by a schema of well-known quality and meaning. To achieve high effectiveness, they leverage schema, expert, and/or external knowledge. Part of these methods are extended to address Volume, processing large datasets through multi-core or massive parallelization approaches, such as the MapReduce paradigm. However, these early schema-based approaches are inapplicable to Web Data, which abound in voluminous, noisy, semi-structured, and highly heterogeneous information. To address the additional challenge of Variety, recent works on ER adopt a novel, loosely schema-aware functionality that emphasizes scalability and robustness to noise. Another line of present research focuses on the additional challenge of Velocity, aiming to process data collections of a continuously increasing volume. The latest works, though, take advantage of the significant breakthroughs in Deep Learning and Crowdsourcing, incorporating external knowledge to enhance the existing words to a significant extent.

This synthesis lecture organizes ER methods into four generations based on the challenges posed by these four Vs. For each generation, we outline the corresponding ER workflow, discuss the state-of-the-art methods per workflow step, and present current research directions. The discussion of these methods takes into account a historical perspective, explaining the evolution of the methods over time along with their similarities and differences. The lecture also discusses the available ER tools and benchmark datasets that allow expert as well as novice users to make use of the available solutions.

KEYWORDS

entity resolution, entity matching, blocking, clustering, data integration, data cleaning, data evolution

Contents

Preface

Entity Resolution (ER) lies at the core of data integration, cleaning, and querying. As a result, a bulk of the research examines ways for improving its effectiveness and time efficiency. Initially, the relevant methods were primarily crafted for addressing *Veracity* in the context of structured (relational) data that are described by a schema of well-known quality and meaning. To achieve high effectiveness, they typically relied on expert and/or external knowledge. Some of these methods were later extended to address *Volume*, processing large datasets through multi-core or massive parallelization approaches, such as the MapReduce paradigm. In the context of Web data, though, these schema-based approaches were inapplicable, as the scope of ER gradually moved toward voluminous, noisy, semi-structured, and highly heterogeneous data collections. To address the additional challenge of *Variety*, recent works on ER adopt a novel, loosely schema-aware functionality that emphasizes scalability and robustness to noise. Another line of recent approaches focuses on the additional challenge of *Velocity*, aiming to process data collections of continuously increasing volume through the ingestion of new information. The latest works take advantage of the significant breakthroughs in Deep Learning and Crowdsourcing, incorporating external knowledge to further enhance the performance of earlier approaches.

This synthesis lecture provides a generation-centric overview of ER. For each generation, we outline the corresponding ER workflow, discuss the state-of-the-art methods per workflow step, and present current and open research directions. The focus is not only on traditional solutions but also on the latest developments in the areas of non-structured, crowdsourced, incremental, progressive, and deep-learned solutions. In the discussion of these methods, we take into account a historical perspective, explaining the evolution of the methods over time, and pinpointing their similarities and differences. This book also discusses the available ER tools and benchmark datasets that allow experienced, as well as lay, practitioners to apply and assess the available solutions. In this way, we provide readers with a deep understanding of the broad field of ER, highlighting the landmarks in this active research domain.

George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas
February 2021

Acknowledgments

We would like to thank all our collaborators (students and colleagues alike) for everything they have taught us.

George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas
February 2021

CHAPTER 1

Entity Resolution: Past, Present, and Yet-to-Come

The core organizational unit in many applications is the *profile*, i.e., the collection of information that pertains to a particular real-world entity. Profiles are used to organize data of any structuredness, be it structured (e.g., relational databases), semi-structured (e.g., knowledge bases), or even unstructured (e.g., free text). For instance, the database of a bank uses profiles that describe clients, the knowledge graph of an on-line social network uses profiles about persons, locations, and events, and a recommendation application uses profiles about products described in plain text.

The semantics of profiles and the connections between them play a crucial role in the performance of a wide variety of applications that range from data analytics to query answering, object-oriented searching, and data evolution. Invariably, though, profiles are dirty, containing noisy, incorrect, redundant, or simply incomplete information. It is of paramount importance to integrate different profiles that actually describe the same real-world entity. For example, two banks that merge need to combine their customer databases so as to detect their shared clients along with the unique ones. This essential task is called *Entity Resolution* (**ER**).

Putting ER into practice entails a number of challenges that arise from the application settings, i.e., the data and system characteristics as well as the resource and time restrictions. The evolving nature of the settings implies a corresponding evolution of the challenges. This explains the plethora of methodologies that have been proposed to successfully apply ER in various domains. For this book, we studied how ER challenges evolved over the years and organized the work in this area into four generations of ER methods. Each generation focuses on addressing particular challenges, which we describe below. Figure 1.1 illustrates these four generations of ER methods on the horizontal axis along with the respective main challenges and their level of difficulty on the vertical axis.

The 1st generation comprises the initial ER methods that deal with textual differences in database records [Christen, 2012a]. These records involve various forms of inconsistencies, noise, or sparsity in profiles, which are introduced during the manual data entry, or by the limitations of the automatic extraction techniques. They are described by schemata of known semantics and quality, their size is moderate, and they are mostly static, i.e., their evolution is typically considered slow and, thus, irrelevant. In this context, the main challenge for ER is *Veracity*, i.e., achieving high accuracy despite the high profile noise. Veracity is addressed by leveraging

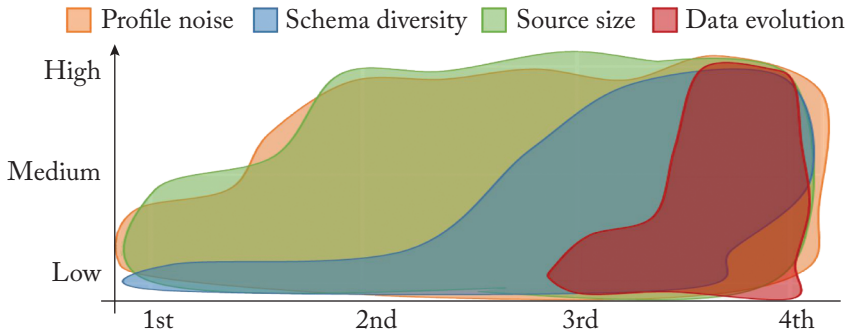


Figure 1.1: Illustration of the challenges along with their corresponding level that are addressed by the four generations of ER.

schema and domain knowledge and/or labeled instances to automatically learn matching rules that simultaneously maximize precision and recall. A workflow consisting of two core steps is typically used: first, the obviously unrelated profiles are quickly discarded, and then the related profiles are detected based on the *similarity assumption*, i.e., the more similar two profiles are, the more likely they are to refer to the same entity, and vice versa.

The 2nd generation focuses on the combined effect of Veracity and *Volume*, as the size of the input now raises up to several million profiles. Given that the rest of the challenges remain similar, there is no significant change in the logic of ER methods. Emphasis is placed, though, on increasing its throughput. This is achieved through parallelization, from multicore to GPU parallel processing. Most efforts, though, focus on the recent paradigm for *massive parallelization*, namely Map/Reduce. In this context, special care is taken to avoid the underutilization of computational resources through Load Balancing techniques.

In the 3rd generation, Veracity and Volume persist and are accompanied by extreme schema diversity, a challenge called *Variety*. The scope of ER has moved to semi-structured or even unstructured Web data, which is dominated by user-generated content. Instead of a database-like schema, there is an unprecedented level of schema noise and heterogeneity as well as loose schema bindings of unclear semantics. For example, there are ∼2,600 different vocabularies in the Link Open Data cloud,[1] but only 109 from them are shared by more than one dataset [Efthymiou et al., 2019]. Inevitably, the schema-aware methods of the previous generations are inapplicable in these settings. Instead, schema-agnostic methods have been amalgamated into a comprehensive end-to-end workflow. This waives the need for expert and domain knowledge, yet it is quite effective and time efficient, especially when leveraging the various parallelization schemes for higher scalability.

[1] https://lod-cloud.net

More recently, the 4th generation tackles the additional challenge of Velocity, which emanates from the continuously increasing volume of input data. To address this challenge, novel progressive methods and workflows produce useful results in a pay-as-you-go manner, before the full completion of ER. Velocity also stems from constraints related to response time, which needs to be low. Query-driven ER resolves profiles while answering to an incoming query profile, whereas Incremental ER refines existing results as new, possibly conflicting evidence becomes available by focusing on a portion of the overall datasets—rather than repeating the entire ER process from scratch.

Even though these generations are more or less numbered by order of appearance, none of them has become obsolete so far. Generation 4 is not the sole focus of research. Instead, all generations are actively being investigated at the moment, especially in relation to the recent, promising developments in Deep Learning and Crowdsourcing. Deep Learning (DL) incorporates complex classification models and powerful pre-trained representations of textual evidence in profiles, while crowdsourcing devises efficient and effective techniques for leveraging human feedback. These two forms of external knowledge can be applied to any workflow step in any generation, improving their performance to a significant extent. These directions are actively being investigated at the moment.

Existing Surveys and Our Contributions. Several surveys and books have already examined various aspects of ER. They focus, though, on a particular generation and/or a particular workflow step. For example, for Generation 1, the main Blocking methods are surveyed in Christen [2012b], the main Matching methods in Getoor and Machanavajjhala [2012], Köpcke and Rahm [2010], Köpcke et al. [2010], and Koudas et al. [2006], while the entire ER workflow for this generation is covered in Christen [2012a] and Elmagarmid et al. [2007], Blocking across all generations is outlined in Papadakis et al. [2020b], whereas Christophides et al. [2015, 2020] and Naumann and Herschel [2010] discuss mostly methods from Generation 3, and Dong and Srivastava [2015] looks into the main methods for Generations 2 and 3.

This work has several differences to the existing ones. First, it groups ER methods into generations, a novel categorization that is based on the typically used aspects of Big Data [Dong and Srivastava, 2015]. This allows for faster navigation to the ER method(s) that can be used for a particular situation or setting. Second, this book is the first one to examine the recent developments in DL and crowdsourcing for ER in a systematic and thorough way. Special care is also taken to present the open-source ER tools along with the established benchmark datasets, facilitating researchers and practitioners to put the main ER approaches into practice.

Outline of the book. The remaining manuscript is organized as follows. Chapter 2 defines the particular task, while Chapters 3–6 elaborate on the four generations, analyzing the respective ER workflow along with the main approaches per workflow step. Chapter 7 discusses the recent advances in deep learning and crowdsourced ER, and Chapter 8 gives an overview of related resources. We conclude in Chapter 9 with a list of possible directions for future work.

CHAPTER 2

Preliminaries

This chapter formally defines the ER task and its core notions along with the measures used for evaluating the generated results. The following definitions and notations are generic and can capture the variations of this problem across the different domains (e.g., in structured and semi-structured data).

The fundamental component of the ER data model is the *profile*, also known as *instance*, *entity description*, or *reference*. Each profile provides information about a particular real-world object, such as an event, location, organization, or person. More formally:

Definition 2.1 A **profile** p_i is a subset of a data source DS, i.e., $p_i \subset DS$, providing information about a real-world object. An **entity** e_k is a set of profiles, i.e., $e_k = \{p_1, \ldots, p_n\}$, where all $p_i, \ldots, p_n$ pertain to the same real-world object. ■

Existing methods use different profile definitions, depending on the data format in the given data source. For example, in the case of a relational database, we could have $p_i = R(a_1, ..., a_k)$, where R denotes the name of the relation and $a_1, ..., a_k$ its attributes. In the case of semi-structured data, we could have profiles corresponding to a set of attribute-value pairs $p_i = \{(a_l, v_m)\}$. The latter definition is more flexible with respect to the schema describing a profile, as it supports a heterogeneous set of attribute names, multiple values for the same attribute, as well as tag-style values, which lack an attribute.

Any pair of profiles from an entity e_k, i.e., (p_i, p_j) s.t. $p_i, p_j \in e_k$, is commutative, transitive, symmetric, and reflexive. To ease notation, in the remaining text, such a pair is denoted with $\mathbf{p_i \equiv p_j}$, and we refer to it using interchangeably the terms **match** and **duplicate**.

The data sources are typically classified according to the number of profiles in each entity. A data source containing at most one profile per entity is called *Clean DS*, whereas one containing multiple profiles per entity is called *Dirty DS*. More formally:

Definition 2.2 A data source is a **Dirty DS** if $\exists\ e_k = \{p_i, \ldots, p_j\}$ s.t. $p_i, \ldots, p_j \in DS$, or a **Clean DS** if $\nexists\ e_k = \{p_i, \ldots, p_j\}$ s.t. $p_i, \ldots, p_j \in DS$. ■

In this context, ER is formally defined as follows [Christophides et al., 2015]:

Problem Statement 2.3 The **Entity Resolution** task aims to generate entities from the profiles included in a set of data sources S. The task is called **Dirty ER** if S includes a single Dirty DS, **Clean-Clean ER** if S encompasses exactly two Clean DSs, and **Multi-source ER** if S comprises multiple DSs, i.e., $|S| \geq 2$.

Example 2.4 Figure 2.1 illustrates a Clean-Clean ER task over structured data. Figure 2.1a depicts the data source D, which conveys author data that is described by three textual attributes of high quality; Figure 2.1b shows the data source W, which involves census data that is described by four textual attributes and a numeric one ("Age")—three of them with missing values; and Figure 2.1c shows the ER result, with every entity including at most one profile from each data source.

A Dirty ER task over structured data is illustrated in Figure 2.2. The original set of profiles in the input data source G is shown in Figure 2.2a. Each profile involves five attributes, three textual, and two numeric ones. The final set of entities appears in Figure 2.2b. Unlike Clean-Clean ER, the size of every entity e_k is arbitrary, $|e_k| \geq 1$.

Finally, a multi-source ER task over structured data is illustrated in Figure 2.3. It includes three different data sources combining data from the two previous cases, as shown in Figure 2.3a. The final set of entities appears in Figure 2.3b.

Typically, ER methods depend on the characteristics of input data. In case a given DS is described by a well-defined schema (e.g., a relational database), its profiles convey schema information that can be used during resolution. Techniques exploiting such information are known as **schema-aware** or **schema-based** ER methods. The opposite situation involves data sources lacking a schema or described by a noisy and heterogeneous schema. Such a schema may be ignored by ER techniques, which are thus called **schema-agnostic** or **schema-independent**.

In some environments, expert knowledge or mere heuristics are available. Such information can become useful by being incorporated into **non-learning** ER methods. In contrast, **learning-based** ER methods use machine learning techniques to train models for solving the problem. These techniques are further distinguished into **unsupervised** methods, which require no labeled data, and **supervised** ones, which employ labeled data in the form of matching and non-matching profiles, called *positive* and *negative instances*, respectively. Arguably, the biggest limitation of supervised approaches is the requirement for an *a priori* labeled dataset in order to train the selected learning algorithm to classify new instances, similar to the ones met in the training set.

Regardless of the input DSs and their characteristics, ER relies on matching functions, which estimate the *similarity* between two profiles, p_i and p_j. The similarity, which is also referred to as *resemblance* and *matching probability*, is used to decide whether two profiles should be included in the same entity, i.e., if $p_i \equiv p_j$ and $p_i \in e_k$ then $e_k = e_k \cup \{p_j\}$. More formally:

Definition 2.5 Given two profiles p_i and p_j, their *similarity* is determined by a **Matching Function**, denoted as $c_{i,j}|p_i, p_j \mapsto [0, 1]$. An **Oracle** is a matching function that decides with 100% accuracy whether p_i and p_j are matching or not. ■

Having an Oracle is of course ideal, but usually unrealistic. In practice, the majority of ER methods approximate the Oracle by using one or more similarity measures along with a threshold

id	Name	Surname	Organization
D1	Robert	Smith	University of California
D2	Joan	Clarke	University of Buenos Aires
D3	Anthony H.	Kane	City, University of London
D4	Joe	Green	PSL University, Paris
D5	Serge	Lenglet	New York University
D6	Jack	Smyth	New York University
D7	Ann	Green	University of Athens
D8	David	York	University of Münster
D9	Luc	Vander Bollen	ULB, Brussels
D10	Argyrios	Samaris	UPV, Valencia

(a)

id	Name	Surname	Age	City	Country
Wl	Antony	Kane		London	UK
W2	Dave	York	33		
W3	Bob	Smith			USA
W4	Jack	Smith	56	New York City	USA
W5	Joanne	Clark	68		
W6	Joe	Green	59	Paris	France
W7	Annabelles	Greenwood	49		Canada
W8	Luke	van der Bollen		Brussels	Belgium

(b)

Entity 1	Entity 2	Entity 3	Entity 4	Entity 5
D1 W3	D2 W5	D3 W1	D4 W6	D5

Entity 6	Entity 7	Entity 8	Entity 9	Entity 10	Entity 11
D6 W4	D7	D8 W2	D9 W8	D10	W7

(c)

Figure 2.1: An example of Clean-Clean ER over *structured* data: (a)–(b) the input DSs and (c) the corresponding entities.

id	Name	Affiliation	Areas of Interest	#Articles	#Citations
G1	Robert Smith	University of California	Artificial Intelligence, Text Mining	25	1602
G2	Joan Clarke	University of Buenos Aires	Entomology	12	441
G3	Anthony H. Kane	City, University of London	Database	9	41
G4	Joe Green	PSL University, Paris	Computer Science, Algorithms	149	6221
G5	Joanne Clark	University of Buenos Aires	Entomology	12	429
G6	Annabelles Greenwood	University of Toronto	Algorithms	2	1
G7	Robert Smith	University of California	Database, Text Mining	26	1610
G8	Antony Kane	Unknown	Biological Databases	9	39
G9	Serge Lenglet	New York University	Entomology	22	2291
G10	Antony Kane	City, University of London	Bioinformatics	5	26

(a)

Entity 1	Entity 2	Entity 3	Entity 4	Entity 5	Entity 6
G1 G7	G2 G5	G3 G8 G10	G4	G6	G9

(b)

Figure 2.2: An example of Dirty ER over *structured* data: (a) the input DS and (b) the corresponding entities.

θ such that $c_{i,j} \geq \theta \Rightarrow p_i \equiv p_j$. A $c_{i,j}$ with a result that does not distinguish between a match or non-match is called **uncertain match**. In these cases, the ER method typically leverages additional information to reach a decision.

2.1 COMPUTATIONAL COST

In the worst case, ER needs to compare every profile with all others, i.e., to apply the Matching function to every possible pair of profiles. The time complexity of this naive, brute-force solution to ER is quadratic with respect to the size of the input. As a result, it cannot scale to large data sources [Christen, 2012a]. To reduce the computational cost of ER to manageable and scalable levels, **Blocking** is typically used. Its goal is to group together similar profiles into clusters, called *blocks*, such that comparisons are executed only inside each block. In this way, the time complexity is now quadratic to the size of the blocks, which is much smaller than the size of the input DSs.

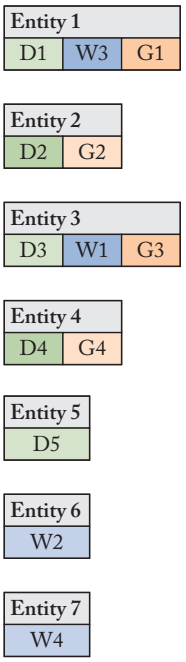
The relative cost of these approaches is illustrated in Figure 2.4, which pertains to a Dirty ER task with N profiles as input. The brute-force approach corresponds to the grey area ($N \cdot (N-1)/2$), while the red circle indicates the minimum possible computational cost, in the ideal

Profile	Name	Surname	Organization
D1	Robert	Smith	University of California
D2	Joan	Clarke	University of Buenos Aires
D3	Anthony H.	Kane	City, University of London
D4	Joe	Green	PSL University, Paris
D5	Serge	Lenglet	New York University

Profile	Name	Surname	Age	City	Country
W1	Antony	Kane		London	UK
W2	Dave	Johnson	33		
W3	Bob	Smith			USA
W4	Jack	Smith	56	New York City	USA

Profile	Name	Affiliation	#Articles	#Citations
G1	Robert Smith	University of California	25	1602
G2	Joan Clarke	University of Buenos Aires	12	441
G3	Anthony H. Kane	City, University of London	9	41
G4	Joe Green	PSL University, Paris	149	6221

(a)



(b)

Figure 2.3: An example of Multi-source ER over *structured* data: (a) the input DSs and (b) the corresponding entities.

case that we exclusively compare the matching profiles. The green circle represents the cost of Blocking, which covers the largest part of duplicate pairs, but not all of them, as it typically constitutes an *approximate procedure* based on heuristics (i.e., it cannot guarantee to detect all duplicates). The closer the two circles are, the more effective is Blocking.

Blocking represents each profile by one or more signatures, which are called *blocking keys*. Then, it places into blocks all profiles having the same or similar blocking keys. More formally:

Definition 2.6 Given a set of sources, i.e., $DS_1, \ldots, DS_k$, a **blocking scheme** B^{key} generates a set of blocks $\{b_1^{key}, \ldots, b_n^{key}\}$ through the **blocking key** *key*. Each **block** b_i^{key} is a subset of the profiles from the given data sources that have the same (or similar) value for *key*, i.e., $b_i^{key} \subset \bigcup_{j=1}^{k} DS_j$. ■

Example 2.7 Figure 2.5 illustrates a simple blocking scheme applied to the Clean-Clean DSs of Figure 2.1. The values of attribute "Surname" are used as blocking keys to form the resulting

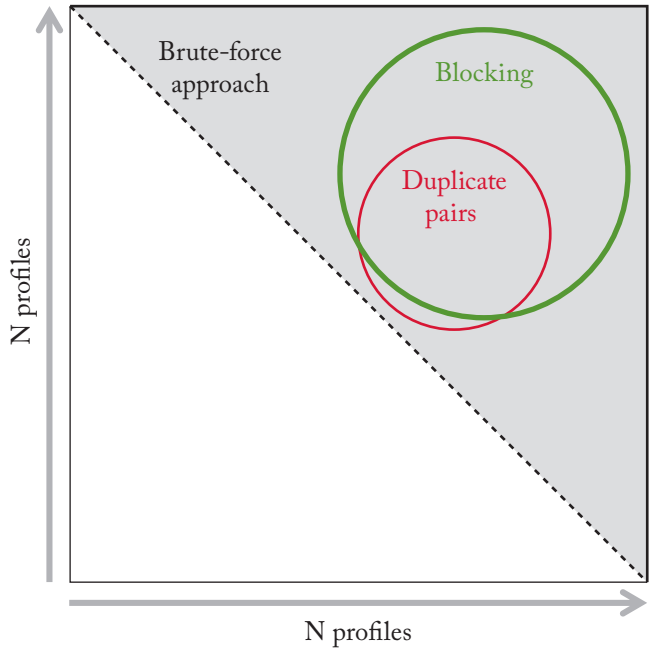


Figure 2.4: The relative computational cost for the brute-force approach, Blocking and the ideal solution (Duplicate Pairs) in the case of Dirty ER, based on Papadakis et al. [2020b].

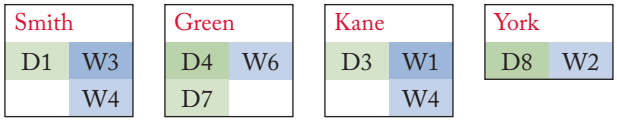


Figure 2.5: A simple blocking scheme applied to the Clean DSs of Figure 2.1.

blocks. The resulting blocks involve just $(2 + 1 + 1 + 1 =)$ 5 comparisons, a computational cost that is significantly lower than the $(10 \times 8 =)$ 80 comparisons of the brute-force approach.

Existing ER methods apply different Blocking mechanisms based on the challenges of the given data. As we will discuss in following sections, we have ER methods with mechanisms for generating non-overlapping blocks (i.e., $\forall b_i^{key}, b_j^{key} \in B^{key} : b_i^{key} \cap b_j^{key} \neq \emptyset$), selecting the most beneficial order for processing blocks, deciding when to stop processing blocks, etc.

2.2 PERFORMANCE EVALUATION

To assess the **effectiveness** of ER methods, the following measures are typically considered.

- *True Positive* (*TP*) gives the number of matched profiles that indeed correspond to the same entity.
- *False Positive* (*FP*) gives the number of matched profiles that actually correspond to different entities.
- *False Negative* (*FN*) gives the number of non-matched profiles that actually correspond to the same entity.

On this basis, the following three measures are defined in the interval [0, 1], with higher values indicating higher effectiveness.

1. *Precision* measures the portion of correctly matched profiles: $Pr = \frac{TP}{TP + FP}$.
2. *Recall* measures the portion of detected matches: $Re = \frac{TP}{TP + FN}$.
3. *F-Measure* is the harmonic mean of Precision and Recall: $F1 = \frac{2 \cdot Pr \cdot Re}{Pr + Re}$.

F-Measure is typically used as the single measure summarizing all aspects of the effectiveness of an ER approach. This practice, though, has been challenged in Hand and Christen [2018], which argues that the F-Measure is also equivalent to an arithmetic mean of recall and precision, with the actual weights between the two depending on the method at hand. An alternative measure, called *Generalized Merge Distance*, has been proposed in Menestrina et al. [2010]. Inspired from the edit distance of strings, it essentially measures the merge and split operations that are required to transform the detected into the perfect results.

Note also that evaluating the actual effectiveness of an ER pipeline or system with respect to the above measures might be challenging. Due to the heavy imbalance between matching and non-matching pairs of profiles (in favor of the latter), a statistically sound approach is required for sampling the ER output. This is offerred by *OASIS* [Marchant and Rubinstein, 2017], an open-source system that implements a principled and efficient approach based on a stratified Bayesian generative model.

The **time efficiency** of ER methods is typically measured through their *overall run-time*, i.e., the time that intervenes between receiving the input profiles and returning the detected entities as output. In practice, this run-time depends largely on the effectiveness of the blocking method that is employed.

To assess the quality of a blocking scheme B^{key} independently of the performance of the subsequent steps like Matching, it is generally assumed that a pair of duplicates is detected as long as they co-occur in at least one block [Christen, 2012b, Dong and Srivastava, 2015, Papadakis et al., 2013, 2014a, Stefanidis et al., 2017]. Given a block b_i, the set of **candidate matches** (i.e., pairs of profiles to be compared) is denoted by $MC(b_i) = \{c_{k,l} : p_k \in b_i \wedge p_l \in b_i\}$, while the set of true entities, to which the given profiles belong, is denoted by $E(b_i) =$

$\{p_i \equiv p_j : p_k \in b_i \wedge p_l \in b_i\}$ [Christen, 2012b, Dong and Srivastava, 2015, Stefanidis et al., 2017].

The size of $E(b_i)$ relates to effectiveness, since it bounds the number of entities that are correctly detected by the overall ER process. The size of $MC(b_i)$ relates to time efficiency, as it determines the number of executed matching functions. Intuitively, a larger $MC(b_i)$ yields a larger $E(b_i)$, though at a higher computational cost, due to the additional comparisons that should be executed. Given that a mere subset of the executed matching functions in MC correspond to valid entities in E, a blocking scheme is considered successful as long as it achieves a good balance between the two sets. This trade-off is commonly captured by the following three measures, which are all defined in the interval [0, 1], with higher values indicating higher effectiveness [Bilenko et al., 2006, de Vries et al., 2009, Michelson and Knoblock, 2006, Papadakis et al., 2011b].

1. *Pair Completeness*, adapted from recall, denotes the portion of existing entities that are detected in the blocks of blocking scheme B^{key}:
   $PC(B^{key}) = \frac{|E \cap MC|}{|E|}$, where $E = \bigcup_{b_i \in B^{key}} E(b_i)$, $MC = \bigcup_{b_i \in B^{key}} MC(b_i)$.

2. *Pairs Quality*, adapted from precision, denotes the portion of candidate matches in the blocks of B^{key} that correspond to valid entities:
   $PQ(B^{key}) = \frac{|E \cap MC|}{|MC|}$.

3. *Reduction Ratio* expresses the reduction in the number of executed comparisons in the blocks of B^{key} with respect to the brute-force approach:
   $RR(B^{key}) = 1 - \frac{|MC|}{|BF|}$, where $|BF| = |DS_1| \times |DS_2|$ in case of Clean-Clean ER over the data sources DS_1 and DS_2, or $|BF| = \frac{|DS| \times (|DS|-1)}{2}$ in case of Dirty ER over the data source DS.

Example 2.8 In the example blocks of Figure 2.5, the set of candidate matches comprises the following pairs: $MC = \{(D_1, W_3), (D_1, W_4), (D_7, W_6), (D_8, W_2), (D_3, W_1)\}$. The corresponding set of entities, which appears in Figure 2.1c, comprises the following profile pairs: $E = \{(D_1, W_3), (D_2, W_5), (D_3, W_1), (D_4, W_6), (D_6, W_4), (D_8, W_2), (D_9, W_8)\}$. Note that E by definition, excludes the *singleton entities* that consist of a single profile—every entity that is not compared to any other is considered as a singleton entity. The set of common pairs is $E \cap MC = \{(D_1, W_3), (D_8, W_2)\}$ and the performance measures of Blocking take the following values:

$PC(\text{B}) = |E \cap MC|/|E| = 2/7 = 0.2857$,
$PQ(\text{B}) = |E \cap MC|/|MC| = 2/5 = 0.40$,
$RR(\text{B}) = 1 - |MC|/(|DS_1| \times |DS_2|) = 1 - 5/(10 \times 8) = 1 - 1/16 = 0.9375$.

We observe that the selected blocking scheme performs a deep pruning of the search space (very high RR) at the cost of very low recall (PC) and moderate precision (PQ).

It should be stressed at this point that ER is typically preceded by *Data Cleaning*, a preprocessing step that improves the quality of the input DSs, by reducing their noise. ER is also succeeded by *Data Fusion*, i.e., the task of merging duplicate profiles into a clean entity by resolving the conflicting information and synthesizing the complementary one. Both tasks lie out of the scope of this book. For more details on Data Cleaning and Data Fusion, please refer to Ilyas and Chu [2019] and to Bleiholder and Naumann [2008] and Dong and Naumann [2009], respectively.

CHAPTER 3

Generation 1: Addressing Veracity

The target of the ER methods in this generation is *Veracity*. They focus on transforming the input profiles into an accurate set of entities that is as close as possible to the corresponding real-world objects. Their input typically comprises structured data that are *homogeneous* or involve low levels of schema diversity. In the latter case, the attributes can be homogenized through a manual or automatic process. Hence, this generation primarily targets noise in the attribute values of profiles, operating in a schema-aware fashion. Depending on the input, this task is usually called **Record Linkage** or **Deduplication** [Christen, 2012a,b]. These terms are synonyms with Clean-Clean and Dirty ER, respectively, and can be used interchangeably with them.

Methods of this generation have been in use since the infancy of ER [Fellegi and Sunter, 1969], but remain at the core of recent methods (e.g., Reyes-Galaviz et al. [2017]) and cutting edge tools [Konda et al., 2016]. We can organize them according to the end-to-end ER workflow in Figure 3.1, which consists of the following steps.

1. *Schema Alignment* is an optional step needed only for Record Linkage over disparate schemata. Its goal is to create alignments between the attributes of the given DSs based on their relatedness, which is inferred from the similarity of their structure, name, and/or included values [Bernstein et al., 2011, Madhavan et al., 2001]. Identifying semantically equivalent attributes (e.g., "profession"-"job") enables the schema-aware operation of the next steps.

2. *Blocking* is necessary for reducing the quadratic time complexity of the naive, brute-force ER approach that compares every profile with all others—a process that cannot scale to large data sources [Christen, 2012a]. It restricts the candidate matches to pairs of profiles that are similar according to some criterion (e.g., common zip code for personal addresses). As a result, it boosts the overall time efficiency at the cost of an approximate solution—the more comparisons it filters out, the more duplicates are likely to be missed.

3. *Matching* performs the comparisons determined by Blocking, applying a Matching Function to the candidate matches [Elmagarmid et al., 2007]. The resulting degrees of similarity along with contextual information (e.g., the match decision of related profiles) are used to assign profile pairs into one of three possible categories, i.e., `match`, `non-match`, and `uncertain` (cf. Section 2).

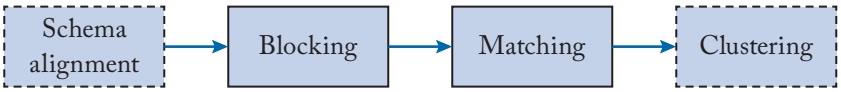


Figure 3.1: Illustration of the 1st generation end-to-end ER workflow. Dashed contours indicate optional steps.

4. *Clustering* is an optional step that goes beyond the local evidence encapsulated in the individual decisions or similarity scores. It leverages their global, collective information to infer indirect matching relations and to discard unlikely matches in favor of pairs with higher matching likelihood.

Next, we delve into the methods corresponding to each workflow component.

3.1 SCHEMA ALIGNMENT

The methods of this step leverage the attribute values in profiles as well as the available, if any, schema knowledge. The goal is to learn and use *attribute mappings* between the data sources, also called *transformations*, *correspondences*, or *rules* [Tejada et al., 2002, Yan et al., 2001]. To this end, *Active Atlas* [Tejada et al., 2002] starts with a collection of generic transformations (e.g., *abbreviation* for transforming "3rd" to "third," *acronym* for transforming "United Kingdom" to "UK") and learns the weight of each transformation given a particular application domain.

Example 3.1 Consider the following simple schema alignment rule: "*Two attributes are aligned if their names are identical*." When applied to the Clean DSs of Figure 2.1, it identifies the attributes pairs: $< DS1.Name, DS2.Name >$ and
$< DS1.Surname, DS2.Surname >$ as semantically equivalent.

Another option is finding similarities between the profile values and relationships among these values [Melnik et al., 2002, Zhang et al., 2011]. For instance, the *similarity flooding* algorithm [Melnik et al., 2002] converts the given data into directed labeled graphs, where the nodes correspond to the schema elements and the edges to the relations between them. The algorithm then detects if the nodes in one graph are similar to the nodes in the second graph, while also propagating the found similarities to the respective neighbor nodes.

There are also methods going beyond schema and value processing. One such direction is the combination of different schema alignment methods, i.e., merging the results from previous alignment operations. This can be done using machine-based approaches, such as Doan et al. [2002], or generic approaches, such as Do and Rahm [2002] and Raffio et al. [2008].

Another direction is based on mapping composition. For example, the work in Madhavan and Halevy [2003] models the relationships between schemata. The work in Fuxman et al. [2006] creates mappings in a recursive manner: first, it defines how the top components of

a schema relate, then it defines how their sub-components relate, and so forth. The resulting mappings are referred to as *nested mappings*.

Other methods are also based on machine learning mechanisms. Ehrig et al. [2005] introduces a machine learning approach that explores the user validation of initial alignments for optimizing alignment methods. *GLUE* [Doan et al., 2002] creates alignments in a semi-automatic way between the schemata, whereas *LSD* [Doan et al., 2001] creates alignments between the schema and a mediated schema. GLUE first applies a set of learners and then it combines the results they generate. User assistance is leveraged in Lee et al. [2007] in order to improve the quality of the outcomes generated by the automatic part of the solution.

Other methods focused on automated or semi-automated tools for creating schema mappings. One such example is the *Clio* system [Hernández et al., 2001], which operates without making assumptions about the relationship between the schemata or how they were created. The automatic tuning of the schema matching of the *eTuner* system [Lee et al., 2007] also belongs to this category. Finally, the *Valentine* suite[1] implements the main methods of this generation. It is used in Koutras et al. [2020] to perform and analyze a detailed comparative experimental study.

3.2 BLOCKING

This step receives as input the original DS(s) along with the output of Schema Alignment (if applicable) and clusters together similar profiles into blocks, which are returned as output. In this way, Matching suffices to compare only the candidate matches inside the blocks.

Internally, Blocking operates in a *schema-aware* fashion, assuming that the input data adheres to a known schema or to aligned schemata. Based on this assumption and respective domain knowledge, the most suitable attributes are used for extracting one or more representative signatures from each profile. These signatures are called *blocking keys* and are composed of (combinations of) parts of values from the most informative attributes. Assuming that these keys reflect the overall similarity of profile pairs, profiles with identical or similar keys are placed into the same block to be compared by Matching [Christen, 2012b].

Depending on how they define blocking keys, Blocking methods are distinguished into *local* and *global* ones. The former rely exclusively on the content of individual profiles, whereas the latter consider collective information from the entire input DS(s) or even from external sources. Each category involves several subcategories, as shown in the taxonomy of Figure 3.2.

3.2.1 LOCAL BLOCKING METHODS

Most methods of this type operate as hash functions: they associate every profile with one or more blocking keys and for every distinct key they create a separate block that contains all cor-

[1] https://delftdata.github.io/valentine

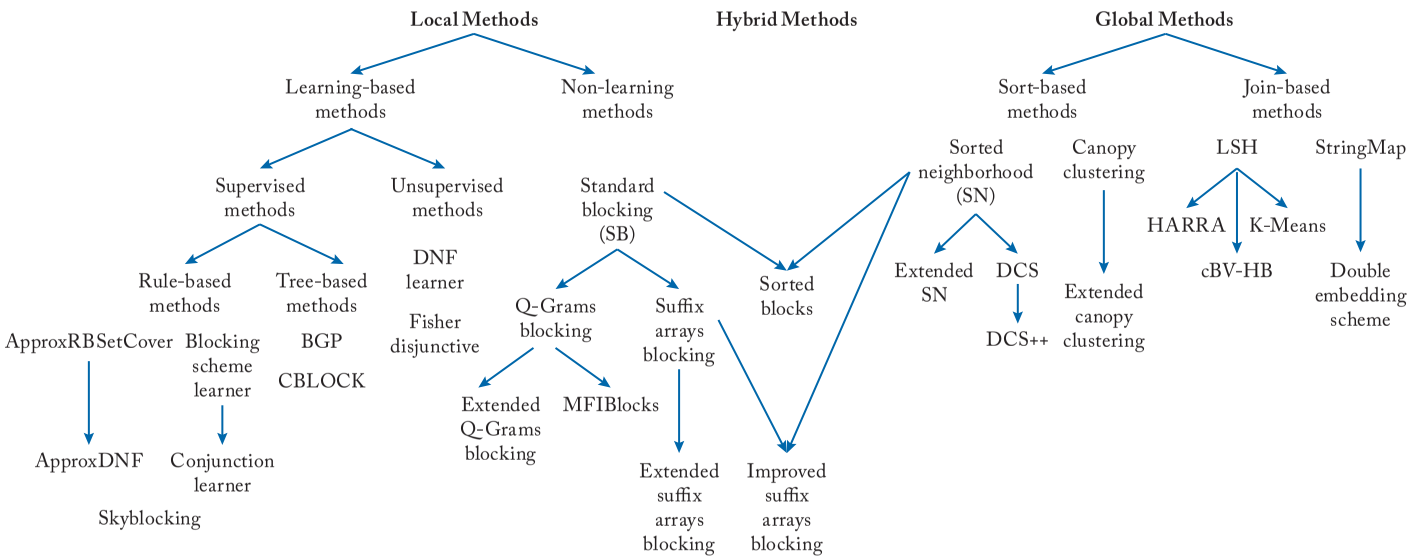


Figure 3.2: The taxonomy of 1st-generation Blocking methods, with the arrows pointing from the original method to its improvement.

responding profiles. We distinguish them into non-learning and learning-based methods, with the latter further categorized into supervised and unsupervised ones.

Non-learning Methods. The cornerstone method here is *Standard Blocking* (SB) [Fellegi and Sunter, 1969], which requires an expert to manually define a part or a transformation of one or more attribute values as the single *blocking key* of each profile. Every profile is then placed in the block corresponding to its blocking key. This hash-based functionality results in *disjoint blocks*, thus being quite sensitive to noise in blocking keys—duplicates with the slightest difference in their keys share no block. To increase its robustness, a multi-pass functionality is applied in practice, i.e., SB is combined with several different definitions of blocking keys.

SB's sensitivity is also addressed by two families of local methods.

1. *Suffix Arrays Blocking* [Aizawa and Oyama, 2005] converts each SB key into the suffixes that are longer than a specific minimum length $l_{\min}$. Then, it defines a block for every suffix that does not exceed a predetermined frequency threshold $f_{\max}$, which specifies the maximum block size.

   Extended Suffix Arrays Blocking [Christen, 2012b, Papadakis et al., 2015] considers all substrings (not just the suffixes) with more than $l_{\min}$ characters so as to support noise at the end of SB keys (e.g., "JohnSnith" and "JohnSmith").

2. *Q-grams Blocking* [Christen, 2012b, Papadakis et al., 2015] converts SB keys into subsequences of q characters (*q-grams*) and defines a block for every distinct q-gram. *Extended Q-Grams Blocking* [Baxter et al., 2003, Christen, 2012b, Papadakis et al., 2015] concatenates multiple q-grams to form more distinctive blocking keys that yield higher precision (PQ) for similar recall (PC). *MFIBlocks* [Kenig and Gal, 2013] concatenates q-grams into itemsets and uses a maximal frequent itemset algorithm to detect those exceeding a predetermined support threshold. These itemsets are then used as the new blocking keys.

Example 3.2 An example applying two local, non-learning blocking methods to the Dirty DS in Figure 2.2 is shown in Figure 3.3. Standard Blocking defines as blocking key the concatenation of the following three pieces of information: (i) {"Name," Last2Characters}, (ii) {"Areas of Interest," First3Characters}, and (iii) {"#Citations," FirstCharacter}. The information extracted from each profile is highlighted in blue in Figure 3.3a. The resulting SB keys appear in Figure 3.3b, while the keys of Q-grams Blocking with $q = 4$ are shown in Figure 3.3c. Each key that is shared by more than two profiles produces a block. However, all Standard Blocking keys are unique, due to differences in the considered attribute values, thus yielding no block at all. In contrast, 4-Grams Blocking yields the blocks in Figure 3.3d, which involve three pair-wise comparisons and two duplicates. Hence, their precision is $PQ = 2/3 = 0.67$, while their recall is $PC = 2/5 = 0.4$.

id	Name	Affiliation	Areas of Interest	#Articles	#Citations
G1	Robert Smith	University of California	Artificial Intelligence, Text Mining	25	1602
G2	Joan Clarke	University of Buenos Aires	Entomology	12	441
G3	Anthony H. Kane	City, University of London	Database	9	41
G4	Joe Green	PSL University, Paris	Computer Science, Algorithms	149	6221
G5	Joanne Clark	University of Buenos Aires	Entomology	12	429
G6	Annabelles Greenwood	University of Toronto	Algorithms	2	1
G7	Robert Smith	University of California	Database, Text Mining	26	1610
G8	Antony Kane	Unknown	Biological Databases	9	39
G9	Serge Lenglet	New York University	Entomology	22	2291
G10	Antony Kane	City, University of London	Bioinformatics	5	26

(a)

id	Key
G1	thArt1
G2	keEnt4
G3	neDat4
G4	enCom6
G5	rkEnt4
G6	odAlg1
G7	thDat1
G8	neBio3
G9	etEnt2
G10	neBio2

(b)

id	Key
G1	thAr, hArt, Art1
G2	keEn, eEnt, **Ent4**
G3	neDa, eDat, Dat4
G4	enCo, nCom, Com6
G5	rkEn, kEnt, **Ent4**
G6	odAl, dAlg, Alg1
G7	thDa, hDat, Dat1
G8	**neBi**, **eBio**, Bio3
G9	etEn, tEnt, Ent2
G10	**neBi**, **eBio**, Bio2

(c)

Ent4
G2
G5

neBi
G8
G10

eBio
G8
G10

(d)

Figure 3.3: Applying Standard and 4-grams Blocking to the Dirty DS of Figure 2.2: (a) the input DS with highlighted the information used in blocking keys, (b) the blocking keys of Standard Blocking per profile, (c) the blocking keys of 4-grams Blocking per profile, and (d) the blocks of 4-grams Blocking—Standard Blocking yields no blocks.

Supervised Learning-based Methods. These methods yield *blocking schemes* composed of complex *predicates*, i.e., combinations of transformation functions that extract a blocking key from the value of a specific attribute, e.g., {title, First3Characters}. Since there are numerous alternative schemes, these methods try to learn the optimal one based on an objective func-