

Intelligent Entity Resolution at Scale: A Graph-Based Social Network Framework

Mohammad Sh. Daoud^{1,*}, Hussain Al-Aqrabi^{2,*}, Yousef Q. Saadeh³, Bayan AbuShawar⁴, Samer Aoudi²

¹ College of Engineering, Al Ain University, Abu Dhabi, UAE.

² Higher Colleges of Technology, Department of Computer Information Science, Sharjah, UAE.

³ The Global Centre for AI Excellence (GCAIE), East Ham, London, UK.

⁴ College of Innovation Technology, Zayed University, UAE.

* Corresponding authors: Mohammad Sh. Daoud, Mohammad.daoud@aau.ac.ae (ORCID: 0000-0003-2682-9231); Hussain Al-Aqrabi, halaqrabi@hct.ac.ae (ORCID: 0000-0003-1920-7418)

Co-author contacts: Yousef Q. Saadeh (Yousef.saadeh@gcaie.org); Bayan AbuShawar (Bayan.abushawar@zu.ac.ae, ORCID: 0000-0003-3062-4511); Samer Aoudi (saoudi@hct.ac.ae, ORCID: 0000-0003-3887-0119).

Received: 3 June 2026 **Accepted:** 9 September 2026 **Published:** 15 October 2026

DOI: 10.15849/ijasca.169 **License:** Open access under CC BY 4.0. Authors retain copyright.

Publisher: Al-Zaytoonah University of Jordan (ZUJ), Amman, Jordan.

Cite this article: M. Sh. Daoud, H. Al-Aqrabi, Y. Q. Saadeh, B. AbuShawar, and S. Aoudi, "Intelligent entity resolution at scale: A graph-based social network framework," *Int. J. Advance Soft Compu. Appl.*, vol. 18, no. 3, pp. 394–410, 2026, doi: <https://doi.org/10.15849/ijasca.169>.

ABSTRACT

Entity Resolution (ER) seeks to identify and merge, within one or more datasets, whichever records ultimately correspond to the same real-world entity. As one of the tasks most consequential for data quality and for the reliability of downstream analysis, ER becomes markedly harder under Big Data conditions, given the sharp rise in both the volume of data and the speed of its arrival. This work approaches ER by recasting it as a problem set within a Social Network space, so that Social Network Analysis (SNA) measures and tools can be brought to bear on it directly. A six-phase framework is proposed that represents entity records as nodes and edges of a graph and applies centrality measures to reveal redundancy among them. Four established models were evaluated against datasets of three different sizes (500, 10,000, and 2,000,000 records): k-means, Levenshtein distance, Jaro-Winkler distance, and Soundex. Levenshtein and Soundex achieved the highest measured precision (100%), while Jaro-Winkler followed closely (99.99%) and k-means trailed substantially (91.33%) despite running fastest; precision was computed as TP/(TP+FP) against RLdata10000's known duplicate structure. Because true matches make up under 0.1% of all record pairs in this dataset, these figures behave closer to a non-match specificity score than to a conventional precision measure computed over predicted matches alone, and should be interpreted with this caveat in mind. The findings indicate that the ER problem can be resolved effectively from within an SNA setting, with the Levenshtein algorithm providing the strongest balance of speed against accuracy for the proposed SNA-driven framework.

Keywords: Entity Resolution; Record Linkage; Social Network Analysis; Graph-Based Entity Resolution; String Similarity.

1 Introduction

Across one or many data sources, ER is the task of picking out the several records that in fact refer to a single real-world entity, whether a person, an organisation, an address, an email address, or a telephone number. Nor is the difficulty of this task fixed: as both the sheer amount of data and the pace at which it is produced continue to climb, the ER problem tied to it grows in step, in scale and in complexity [1]. Two broad families of technique are conventionally used to address it. Methods in the first are deterministic, following explicit rules; methods in the second are probabilistic. Either way, each candidate pair of records drawn from a dataset is compared and assigned a value showing whether the pair should be read as a match or as a non-match.

The present study returns to the ER problem from within a Social Network Analysis setting, even though the actual mechanics used to resolve it remain the familiar probabilistic and deterministic ones: Matching Entity Similarity and Graph Representation. Social Network Analysis (SNA) can be described as a family of techniques for depicting networks visually, for characterising the distinctive properties of

a network's overall shape, and for constructing mathematical and statistical models of how such networks are put together and how they change [2]. A social network, in the conventional account, is a set of participants — nodes — each linked to one or more further participants through some category of relationship; those relationships appear visually as the edges running between the nodes. Since every node here denotes either a person or an organisation, the weight given to an edge conveys just how strong the underlying relation of interest between the two really is. Two interlocking layers thus make up a social network, the participants on the one hand and the ties joining them on the other, and it is upon these two layers that most analyses of social networks are ultimately built. A Social Structure, together with the effects that follow from it, can therefore be studied through SNA: doing so makes the interactions between participants clearer, sheds light on how they behave and think, and brings out whichever features of their ties matter most [3].

What this paper contributes is a framework built to strip out redundancy from the linkage stage of Entity Resolution, taking Social Network Analysis (SNA) as its foundation. Four state-of-the-art modelling algorithms are used to gauge how well it performs, judged on both running time and accuracy. As for what follows: Section 2 surveys related work, Section 3 formulates the problem and sets out the proposed six-phase framework together with the algorithms evaluated, Section 4 presents the implementation, results and discussion, and Section 5 closes the paper.

2 Related Work

A thorough survey of Entity Resolution work found in the literature, and drawn upon for the present study, is set out in this section [4].

Which ER strategy is chosen depends on the data being linked. Two approaches may well be combined, but whatever combination is used must share one requirement: every approach folded in has to serve the same end: resolving possible matches, non-matches, and matches. A large portion of current research in the topic leans on a deterministic approach on its own, or on some blend of deterministic and probabilistic approaches; the related work behind the principal approaches is surveyed below.

Relational similarity now enters into a number of recent ER studies. Bhattacharya and Getoor [5] set out collective entity resolution, an approach that weighs attribute similarity together with relational similarity across records and, at a higher computational cost, delivers highly accurate resolution as a result. Take, as an illustration, the records "Jane Aha" and "Jane Lindsay": they are resolved by string and relationship similarities, so that if the two share identical parents they are read as one person regardless of how low the string similarity between the two records happens to be. What distinguishes collective entity resolution is its use of neighbouring nodes as the discriminating signal for ER within the record network graph, a feature that suits it well to data already cast in graph form.

Representation Learning-based Graph Alignment is a method for graph alignment. (REGAL) was put forward by Heimann et al. [34] to find corresponding nodes across separate networks, treating In entity resolution, attribute-based identification serves as a structural cue for identity inference. Three stages make up the procedure: establishing node identity; building an effective, similarity-based representation; and carrying out fast node-recognition alignment. One limitation of that study is worth noting: the experiments were confined to a pair of similar datasets, with the hyper-parameters left fixed throughout. Here, REGAL functions as an auxiliary, graph-theoretic component, with SNA supplying the representation of the datasets, and the algorithm proved highly capable of resolving ER problems by graph-based means.

Koudas et al. [6] set out pairwise-ER, a method centred on the matching step in which two records are compared to establish whether they correspond; the same authors also put forward similarity matrices together with efficient sub-quadratic approximate join algorithms. When the SSN field is one of the identifiers in a high-quality dataset, their matching algorithm returns highly accurate results — a condition not met here, since the dataset used in this work has no SSN identifier, which limits the algorithm's usefulness for the present purpose. Gomatam et al. [7], in a separate contribution, suggested the Stepwise Deterministic Strategy (SDS) for matching, built on an all-or-nothing rule keyed to a

unique identifier (the matching key): comparing, say, the forename and surname of two records, the SDS treats them as a match only where every character agrees across both names, and as a non-match otherwise. Records under the SDS pass through a series of phases, each of which determines a pair's link status based on precise agreement across a selected subset of unique identifiers; duplicates and everything that is still unlinked from both datasets advance to the next step, while unique matches are drawn off at each step, which then works from a different identifier subset. A pair failing the first round of criteria is not discarded but passed on to a second round — a distinguishing feature of the SDS's stepwise design — and adopting this strategy here produced a marked increase in the share of Links and records were successfully matched..

An ER technique that matches records using four personal attributes—name, date of birth (DOB), SSN, and address—was proposed by Wang et al. [8].. The algorithm combines the similarity of these attributes and learns a matching threshold from expert-labelled identity pairs. However, its reliance on these four identifiers limits its flexibility, particularly when some attributes are unavailable or missing from a record. Gravano, Ipeirotis, Koudas and Srivastava [9] took a different route, searching for approximate string matches across records using the cosine similarity metric, which is well-known from information retrieval. The problem with this approach is that it can be expensive to replicate the exact result of a content join, so it works better for finding string matches than for precise join simulation.

Swoosh, a generic match-and-merge approach to entity resolution, was put forward in [10] and comes in three variants: Swoosh G, Swoosh R, and Swoosh F. The most versatile of the three, G-Swoosh, leaves its match and merge functions unconstrained; R-Swoosh instead draws on matching identifiers together with merge functions and compares whole-record specificity; and F-Swoosh likewise uses matching identifiers and merge functions but pushes the comparison down to individual record features. Small datasets are what G-Swoosh copes with, whereas F-Swoosh and R-Swoosh scale to large ones, making F-Swoosh the more suitable choice once the identifiers entering the comparison are taken into account.

A parallel line of work has examined how far pre-trained language models can carry the matching step itself. Zeakis et al. [12] benchmark twelve popular pre-trained language models, including fastText and several BERT variants, across seventeen established entity-resolution datasets, finding that no single model dominates across both blocking and matching and that transferability between the two steps is limited. Such findings are a reminder that embedding-based effectiveness is highly dataset-dependent — part of the motivation for evaluating multiple matching algorithms side by side, as this paper does, rather than committing to a single method in advance.

Attention-based graph models have also been extended to entity resolution. Li et al. [11] propose a Contextual Semantics Graph Attention Network that separates token-level and attribute-level information into a hierarchical heterogeneous graph, allowing the same token to carry different weights under different attribute contexts; the approach improved F1-score over transformer and graph-convolutional baselines on standard product- and beverage-matching benchmarks, illustrating how richer graph structure can compensate for the semantic loss that pre-trained embeddings suffer on domain-specific vocabulary.

At the scale this paper is concerned with, Beraha and Favaro [13] address the statistical cost of large-scale ER directly, proposing a microclustering prior for Bayesian entity resolution that scales linearly in the number of clusters rather than the sample size, paired with a variational-inference scheme reported to run orders of magnitude faster than existing MCMC-based approaches while retaining competitive clustering accuracy. Taken together, this recent work reinforces two premises of the present study: that casting ER as a graph or network problem is an active and productive direction, and that doing so without sacrificing scalability remains an open, live concern for Big Data settings.

3 Problem Formulation and Proposed Method

The ER problem is formulated here as a graph problem: entity records are represented as nodes, and the relations linking them as edges, so that Social Network Analysis measures can be computed directly

over the dataset. The remainder of this section defines the graph representation used, reviews the string-matching and clustering algorithms evaluated, and then sets out the proposed six-phase framework in enough detail to permit independent replication.

3.1 Preliminaries

As a branch of mathematics, graph theory takes networks as its object of study. A graph is a data structure made up of a set of nodes (vertices) and a set of links (edges) that connect the nodes and explain any relationships between the vertices. A network is just a collection of linked items.

Graph theory originates with the eighteenth-century Swiss mathematician Leonhard Euler, who used graphs to resolve the Königsberg Bridge problem; whether the city could be crossed while traversing each of its seven bridges exactly once [14]. A graph admits such a traversal only if every vertex has even degree, or exactly two vertices have odd degree; Euler's own bridge graph has four odd-degree vertices and is therefore not traversable, a result that supplied graph theory's first theorem [15]. The strength of graph theory lies in its abstraction: built from only two components, vertices and edges, it can capture any pairwise relation between objects, whether computer networks, transportation networks, or, as in this work, social networks, and can be classified as undirected or directed according to whether an edge carries a single direction, and as weighted or unweighted according to whether a numerical magnitude is attached to the relation. Social media applies this formulation directly, representing users as vertices and relations such as friendship or following as edges [16], [15].

Graph theory suits the present work better than related data structures such as trees or linked lists. A tree grows from a single root into child nodes carrying sub-trees, with traversal running one way only, from root to leaf, each child having exactly one parent and no cycles permitted; every tree is therefore a graph, though not every graph is a tree, since an edge between two tree nodes runs one way only, whereas a graph permits the connection to run in both directions over time [17]. A linked list is similarly a one-way or two-way chain of nodes and pointers. A singly linked list moves forward only, while a doubly linked list adds a reverse pointer. But even the doubly linked list holds exactly two references per node, unlike a graph node, which may hold any number [18], [19].

3.2 String-Matching and Clustering Algorithms

Four algorithms were implemented and evaluated within the proposed SNA context: Levenshtein distance, Jaro-Winkler distance, k-means clustering, and Soundex.

The Levenshtein Distance, or Edit Distance [20], measures the similarity between two strings by counting the fewest insertion, deletion, and replacement operations needed to make one string equal the other [16], [21]. For strings a and b , the distance $d(a, b)$ is this minimum edit count; the sequence $a =$ "matching", for example, can be turned into $b =$ "meanings" across the five steps below:

1. matching -> metching (substituting a with e).
2. metching -> meaching (substituting t with a).
3. meaching -> meanhing (substituting c with n).
4. meanhing -> meaning (deletion of h).
5. meaning -> meanings (insertion of s at the end of the sequence).

Five operations suffice to turn "matching" into "meanings", giving a minimum edit distance of $d(a, b) = 5$. Read as a dissimilarity measure, the larger this distance grows, the more unlike the two strings are judged to be; two identical strings, such as "hello" paired with "hello", instead return a Levenshtein distance of zero. Equation (1) gives the corresponding formula. Algorithm 1 presents the dynamic-programming computation of the Levenshtein distance."

Algorithm 1. Levenshtein Distance Algorithm

- Set $a = \text{string } 1$ and $b = \text{string } 2$.
- Set i as terminal character position of string a , and j as terminal character position of string b . Therefore, a_i refers to the string character at position i , whereas b_j refers to the string character at position j .
- If $a_i = b_j$ this means that the characters are the same, no modification or edition needed, which in turn does not add 1.
- if $a_i \neq b_j$ add 1 for modification.

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) \\ \min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} \end{cases} \dots(1)$$

The Jaro-Winkler Distance is applied chiefly to duplicate detection and to gauging how similar two strings are [22]. Algorithm 2 summarizes the Jaro-Winkler similarity computation. As a string metric, it computes a similarity score between two strings, taking into account the ordering of shared characters and allowing for the usual alphabetic variation, which is why it is used above all in record linkage [23]. Fig. 1 sets out its implementation steps. Throughout this paper the Jaro-Winkler score is reported as a similarity, not a distance: a value of zero indicates no similarity between two strings and one indicates an exact match; the algorithm itself turns on three elements:

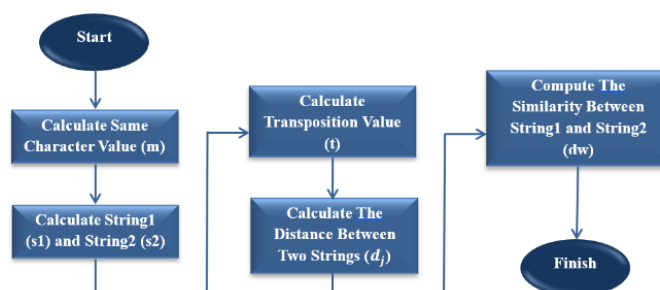


Fig. 1. Jaro-Winkler Distance Algorithm Steps. Source: Created by the authors.

Algorithm 2. Jaro-Winkler Algorithm

1. Compute the string length.
2. For the two strings to be compared, find the same number of characters.
3. Find the number of replacements.
4. Combine matches (m), transpositions (t) and common prefix length (l , capped at 4) with scaling factor $p = 0.1$ into the Jaro and Jaro-Winkler similarity formulas, as detailed in Fig. 1 and [22].

Among the techniques most widely relied upon in exploratory data analysis for making sense of a dataset's structure is clustering. To identify subgroups is, in effect, to arrange objects so that those sharing a subgroup (cluster) are as alike as the chosen similarity measure allows, while those split across different clusters are as unlike as possible — in other words, to isolate homogeneous subgroups within the data [24]. Which similarity measure is fit for the job is a decision that depends on the application at hand. Because no ground truth exists against which the algorithm's output could be checked, clustering counts as an unsupervised method, its performance impossible to gauge directly [24].

K-means [25], adopted here as the clustering method of choice, owes its wide use [26] largely to its simplicity, and finds routine application in data mining and pattern recognition alike. Being iterative, it works by splitting the dataset into K subgroups (clusters), each object assigned to exactly one, with objects inside a cluster kept as alike as possible and the clusters themselves kept as distinct as possible;

the smaller the variation within a cluster, the more homogeneous its data points turn out to be [26]. Algorithm 3 gives the k-means clustering procedure.

Algorithm 3. K-means Algorithm	
1.	<i>The algorithm begins by selecting k (number of clusters) of the dataset randomly to represent the initial centers (centroids) for the clusters.</i>
2.	<i>Assign each of the remaining objects to its closest centroid, measured by the Euclidean distance between the object and the centroid; this stage is called the “cluster assignment” and produces the initial classification. If the classification is unreasonable, the algorithm recalculates each cluster’s new mean value and reassigns the objects accordingly, a stage referred to as “centroid update”. After the centres are recalculated, all objects are re-checked against the new centroids to identify any closer clusters.</i>
3.	<i>The cluster-assignment and centroid-update stages iterate until convergence, where the centroids no longer change and no further reassignment of objects to clusters occurs. In k-means clustering, briefly, k denotes the number of clusters that will be created in the final solution.</i>

Soundex [21] is a phonetic algorithm that encodes names to a shared representation regardless of minor spelling differences, letting two words of like pronunciation but unlike spelling be compared to see whether they in fact sound alike — useful wherever a search touches people's names, such as a database or text file where misspellings are common. Each letter of the input word is converted to a numeric value, so that every word is reduced to a four-element code. Algorithm 4 specifies the Soundex encoding used in this study.

A text search for "Smith", say, would miss someone named "Smythe" — yet Soundex catches the two as alike, since both surnames reduce to the code "S530". Building this four-character code follows a fixed set of rules, given below [21]: (A worked example is provided in Fig. 3)

Soundex Algorithm

Algorithm 4. Soundex Algorithm															
1.	<i>Keep the first letter of the word.</i>														
2.	<i>Change all occurrences of the following vowels to zero: a, e, i, o, u, y (h and w are not coded but, unlike vowels, do not break a run of identical digits — see step 4)</i>														
3.	<i>Replace the digits as follows (after the first letter) as in Fig. 2:</i>														
<table border="1"> <thead> <tr> <th>Letter</th><th>Numeric Code</th></tr> </thead> <tbody> <tr> <td>b, f, p, v</td><td>1</td></tr> <tr> <td>c, g, j, k, q, s, x, z</td><td>2</td></tr> <tr> <td>d, t</td><td>3</td></tr> <tr> <td>l</td><td>4</td></tr> <tr> <td>m, n</td><td>5</td></tr> <tr> <td>r</td><td>6</td></tr> </tbody> </table>		Letter	Numeric Code	b, f, p, v	1	c, g, j, k, q, s, x, z	2	d, t	3	l	4	m, n	5	r	6
Letter	Numeric Code														
b, f, p, v	1														
c, g, j, k, q, s, x, z	2														
d, t	3														
l	4														
m, n	5														
r	6														
4.	<i>Remove all pairs of consecutive digits, which means remove all digits that are the same remove unless a vowel (H, W, and Y) was found between them in the original word.</i>														
5.	<i>Remove all zeros from the result you get.</i>														
6.	<i>Pad the resulting word with extra zeros if less than three digits and return the first three digits, yielding a four-character code (one letter followed by three digits) of this form: <UpperCase> <digit> <digit> <digit>. In other words, if you are not able to assign three digits because you have too few letters in your case, enclose with zeros until three digits are available. You only retain the first three if you have four or more digits.</i>														

Fig. 2. Soundex Letter-to-Numeric-Code Mapping. Source: Created by the authors.

The name: "Herman"	The name: "Hermann"
1. H erman (keep the first letter)	1. H ermann (keep the first letter)
2. H0 rm0n (change vowels to zeros)	2. H0 rm0nn (change vowels to zeros)
3. H06505 (change to numerical value)	3. H065055 (change to numerical value)
4. H06505 (no consecutive digits, remain same)	4. H06505 (remove all consecutive digits)
5. H655 (remove all zeros)	5. H655 (remove all zeros)
6. H655 (four numbers)	6. H655 (four numbers)
The name "Herman" has the same representation of the name "Hermann", which is both equal to the code (H655).	

Fig. 3. Soundex Worked Example: "Herman" vs. "Hermann". Source: Created by the authors.

3.3 Proposed SNA-Based Framework

What is proposed and implemented here is a six-step framework built to handle the Entity Resolution problem while stripping out redundancy; Fig. 4 traces the flow across all six steps, each of which is taken up in turn in the subsections that follow.

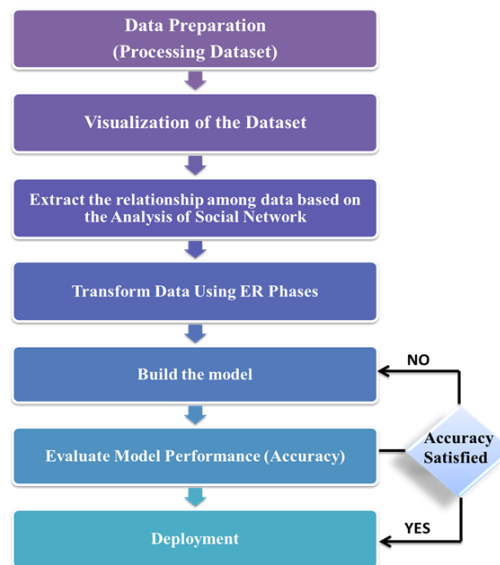


Fig. 4. The Six-Phase Framework. Source: Created by the authors.

Data Preparation

During this stage, two tables are created from the raw dataset: one for the nodes and one for the edges. The data are transformed, in effect, into a directed graph. A NodeID column anchors the node table, which may add optional columns such as label, shape, node name, size or degree, each standing for a node attribute; the edge table, meanwhile, carries two columns, From and To, fixing the direction between nodes, alongside whichever optional columns — edge type, edge label, width, edge level — the case requires.

Fig. 5 offers an illustration: a sample dataset, the North Carolina Voters Dataset, is shown being converted into node and edge tables, with a node-id field appended to each record so that edges linking the different ids can be built.

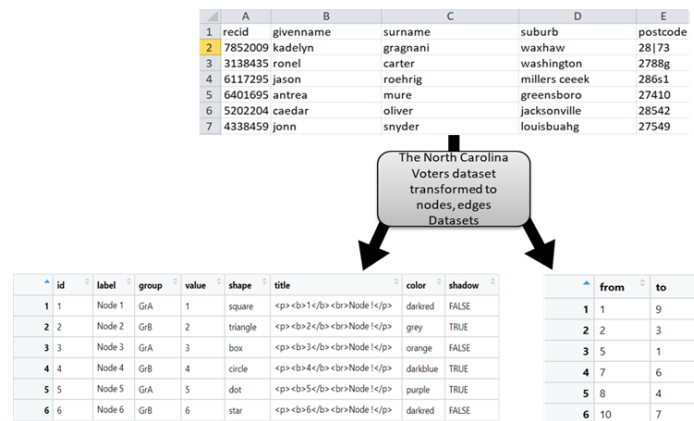


Fig. 5. Illustration of the Data-Preparation Step. Source: Created by the authors.

Visualizing the Dataset

Once node and edge tables have been derived, the data lend themselves to display as a network graph of connected entities, shown in Fig. 6. Whatever capable network-visualisation package is to hand will serve — VisNetwork among them, an R package built for interactive visualisation that is straightforward to apply and offers controls spanning collapsed nodes, zoom, selection, physics, node movement, events and tooltips [27]. Through network visualisation the dynamics of a network — provider and seeker nodes, transmitter and receiver nodes, among others — become explorable by eye, with no statistical analysis required, the node itself standing as the network's most important element; VisNetwork further sorts the data into clusters keyed to connection type.

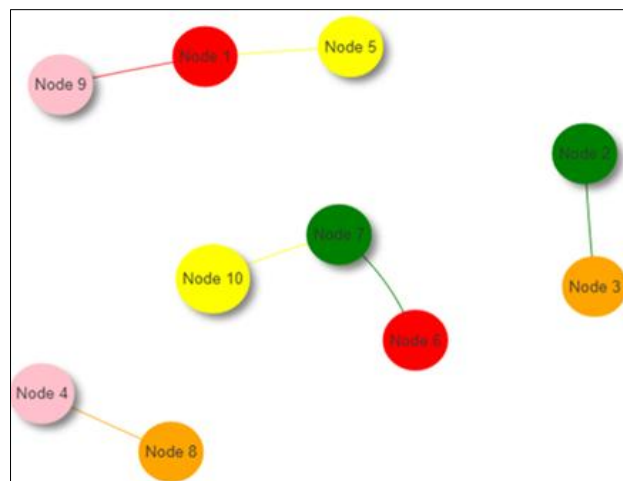


Fig. 6. Example of Network Visualization. Source: Created by the authors.

Analyzing the Social Network

Understanding a network relies heavily on social-network-analytics measures. Visualising the structure formed by the graph's elements calls, in turn, for a further set of straightforward network calculations known as centrality measurements, which lean on graph theory to work out how important any given node is. Each such measure comes with its own definition and its own behaviour; the principal measures used in social-network analysis are set out below:

Degree Centrality measure

A node's connectivity is what Degree Centrality (CD) simply measures — that is, however many connections the node itself holds. Read in terms of the nodes any given node links to, it shows how many connections that node has and flags whichever node carries the most; the higher a node's degree, the more central it stands, which is why the measure serves so well in uncovering well-connected or popular individuals, along with whoever is well placed to carry most of a network's data or to reach its wider reaches with ease.

Item	Description
Graph	$G = (V, E)$, where V is the set of nodes (vertices) and E is the set of edges.
Degree Centrality (DC)	Measures how strongly a node is connected to other nodes in the network [28].
Equation (2)	$D_C(v) = \frac{\deg(v)}{n - 1}$
$D_C(v)$	Degree centrality of node v .
$\deg(v)$	Number of edges connected to node v .
n	Total number of nodes in the network.
Fig. 7 example	Node v has 8 edges and the network contains 9 nodes.
Calculation	$D_C(v) = \frac{8}{9 - 1} = 1$
Interpretation	$D_C(v) = 1$ means that node v is connected to every other node and therefore has the highest degree possible centrality.
Meaning of higher DC	A higher DC value indicates that the node is more highly connected within the network.

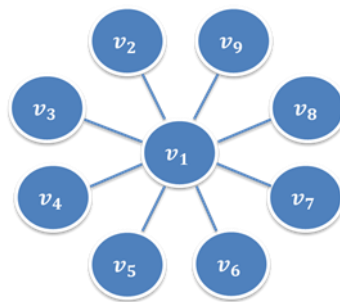


Fig. 7. Degree Centrality. Source: Created by the authors.

Betweenness Centrality measure

What this measure records is how far a given node sits along the shortest paths linking other nodes, thereby marking out which nodes function as "bridges" within the network; it works out, specifically, the share of shortest paths passing through a node — a centrality read in terms of how a node connects others — and so shows how deep into the graph's sub-components a node sits, governing the flow of information as every shorter path is traced and every instance of a node falling on one is tallied. Selecting individuals who mediate between communities is one use to which it is put. Below follow the equation and its parameters:

$$C_B(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v_i)}{\sigma_{st}} \quad (3)$$

Closeness Centrality measure

Because central nodes hold, on average, the shortest paths — their closeness — to every other node in a graph, they reach the rest of the network quickly. Closeness Centrality accordingly expresses how near a given node lies to the others, a centrality read in terms of speed of access [29]; it ranges over the shortest paths linking every pair of nodes and rules that each node's standing depends on the aggregate length of those paths, capturing, too, how long information would take to spread from v out to every other node in succession (rapid spread) [30] — a

property useful for picking out whichever individuals are best placed to sway the whole network swiftly. Equation (4) gives the formula for Closeness Centrality.

$$C_C(v_i) = \frac{n-1}{\sum_{j \neq i} d(v_i, v_j)} \quad (4)$$

Eigenvector Centrality measure

How important a node is by virtue of its ties to other highly influential nodes is what eigenvector centrality measures, such that a node scoring high on eigenvector centrality ought itself to be tied to further high-scoring nodes. It extends degree centrality by folding in a node's neighbours' importance indirectly — much as a person's standing depends partly on how important their own friends are, so having well-connected neighbours strengthens the signal. What Eigenvector Centrality can pick out are nodes that sway the network as a whole, not merely those directly attached to a given node, weighing both how well-connected a node is and how many links its own connections carry in turn. Understanding human social networks benefits from this measure, a valuable product of SNA, and the method indeed sits close in spirit to the scheme Google uses to rank web pages by importance [31]. Equation (5) gives the formula for EC.

$$x = \frac{1}{\lambda} Ax \rightarrow C_E(v_i) = \frac{1}{\lambda} \sum_{v_j \in N_i} A_{ij} C_E(v_j) \quad (5)$$

Transform Data Using ER Phases

Within this framework phase, the finalised datasets are further traversed through six standard ER phases adapted from the entity-resolution literature — distinct from the six phases of the overall framework introduced above — before the ER problem is resolved. Data Preparation comes first, made up of three mini-stages: encoding, in which input data are converted from one encoding scheme into another; conversion, wherein the data format itself shifts between several representations; standardization, in which the represented data are normalised to a shared, common format. Blocking follows as the second phase, achieved by splitting the dataset's records into several blocks so that pair-wise comparison touches only records sharing a block — a step taken to sharpen the ER approach's efficiency. Third comes Data Representation, where the dataset is mapped onto an existing representation, a network graph in the present case. Similarity computation, the fourth phase, settles whether entity pairs sharing a block in fact co-refer. Clustering, fifth, gathers the duplicate entities into clusters, each one standing for a single unique object; and Merging, the sixth and final phase, fuses every cluster's entities into one represented object.

Building the Model

During this stage, a deterministic (rule-based) solution to address the ER problem is developed; the String Similarity Function, long used for ER on this basis, takes here the form of the Sequence-based String Similarity Function, whose best-known instance is the string edit distance. Converting *s* into *t* carries an edit distance of five, and the String Similarity Example sets out the steps by which that distance is reached, Fig. 8. The algorithm calculates the distance between prefixes of length *i* and *j*, where *c*(*s*[*i*], *t*[*j*]) indicates the cost of replacing the *i*-th character of *s* with the *j*-th character of *t* (set to zero, or to some negative value, once the two characters match), and *c*(*s*[*i*], *h*) and *c*(*o*, *t*[*j*]) indicate the costs of removing the *i*-th character of *s* and adding the *j*-th character of *t*, respectively.

Insert "t"	"12 8 Street" → "12 8t Street"
Insert "h"	"12 8t Street" → "12 8th Street"
Substitute "e" with "."	"12 8th Street" → "12 8th Str.et"
Delete "e"	"12 8th Str.et" → "12 8th Str.t"
Delete "t"	"12 8th Str.t" → "12 8th Str."

Fig. 8. String-Similarity Edit-Distance Example. Source: Created by the authors.

Figure 9 sets the scene: The clustering outputs A1 and A2 from systems S1 and S2 are stored in panels (a) and (b), the ground-truth cluster is shown in panel (c), and the outcome is displayed in panel (d) following the application of the blocking strategy from stage 4. as summarised in Fig. 10. Within the S1/S2 systems, cluster A1 groups nodes A, B and C together, so the edges linking them take the value +1 (matching); by contrast, the edges A–D, B–D and C–D each take the value –1, node D sitting in a separate cluster (non-match).

Edge	Decision Features (dj1, dj2)
A-B	(+1, -1)
A-C	(+1, -1)
A-D	(-1, -1)
B-C	(+1, +1)
B-D	(-1, +1)
C-D	(-1, +1)

Fig. 9. ER Using Graph Representation (clustering outputs and blocking). Source: Created by the authors.

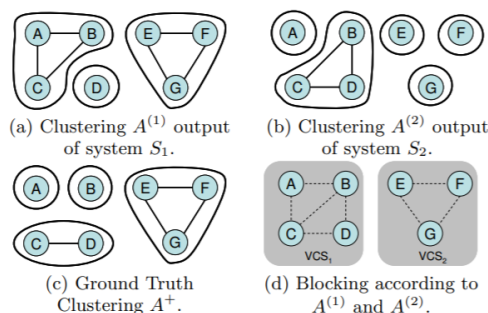


Fig. 10. ER Using Graph Representation (network view of matched vs. non-matched clusters). Source: Created by the authors.

Accuracy and Deployment

Evaluating the model is a step central to model development, since it points to whichever model best represents the data and to how well the chosen model is likely to perform going forward. Where the accuracy proves satisfactory, the process moves on to the next stage — deployment or publishing; where it does not, the process loops back to model-building for a further check.

Judging how well a given model suits the data falls to accuracy and error measures. MAE, RMSE, MSE and MAPE are general-purpose measures considered during the design of the evaluation framework [32]; for the categorical match/non-match outcome produced by entity resolution, however, the results in Section 4 are reported using precision, computed as $TP/(TP+FP)$, which is the standard measure for this classification-style task.

Model performance is evaluated using four common error measures: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Square Error (MSE), and Mean Absolute Percentage Error (MAPE). Once the model has been evaluated, the final stage involves its deployment.

A demonstration reflecting the work's results follows once the model is built and its performance checked, built with a tool R studio supplies — R Shiny, an interactive web application constructed directly in R [33].

4 Results and Discussion

4.1 Experimental Data and Setup

What the proposed framework's implementation looks like is set out in the section that follows.

RLdata10000, supplied as part of R's RecordLinkage package, is the dataset drawn on in this work. Record linkage itself denotes identifying, across a variety of sources, records that stand for the same entity; as Table 1 shows, this particular dataset is built from a real person's records together with generated duplicates that supply further records for a given entity — reflecting, say, how that person's name happened to be recorded — and serves here to check record matching, non-matching, and the removal of duplicates.

Table 1. Dataset Description.

The RL10000 Dataset	
Domain	Persons
Attributes	Id, firstname_c1, firstname_c2, lastname_c1, lastname_c2, Birth Year, Birth Month, Birth Day
Number of Entities	10,000 (10 Thousand)

Ten thousand record entities make up RLdata10000, described across eight variables: Id, firstname_c1, firstname_c2, lastname_c1, lastname_c2, Birth Year, Birth Month and Birth Day. Once the various algorithms are applied to it, this dataset yields the matching and non-matching counts reported in Table 2.

Table 2. RLdata10000 Sample Records.

	fname_c1	fname_c2	lname_c1	lname_c2	by	bm	bd
1	FRANK	NA	MUELLER	NA	1967	9	27
2	MARTIN	NA	SCHWARZ	NA	1967	2	17
3	HERBERT	NA	ZIMMERMANN	NA	1961	11	6
4	HANS	NA	SCHMITT	NA	1945	8	14
5	UWE	NA	KELLER	NA	2000	7	5
6	DANIEL	NA	HEINRICH	NA	1967	5	6
7	MARTIN	NA	ZIMMERMANN	NA	1982	11	2
8	ANDREAS	BENJAMIN	BERGMANN	NA	1989	9	13

Ten thousand rows and seven variables make up the dataset noted above, each row a single record, as Table 2 shows.

A function assembles record pairs that are subsequently classified as a link (match) or a non-link (no match), operating on a single dataset earmarked for de-duplication; among the arguments it takes, the first names the dataset itself. Four arguments in total are accepted: the dataset; blockfld, fixing the blocking step for use with a large dataset; strcmp, setting whether a string metric applies; and strcmpfun, naming the particular string-similarity algorithm used here.

K-means, an unsupervised classification technique, is the last method drawn on in this work; well known and widely used as a clustering algorithm, it operates by iteratively minimising within-cluster variance and belongs among the iterative-descent family of clustering methods. Merging every record into a single clean table free of duplicates is entity resolution's final step; which record makes it into the merged table is left to the user's discretion, and R supplies several functions both for the merge itself and for confirming that no duplicates remain.

4.2 Model Performance

Only the output from the four models used to solve ER — Levenshtein, Jaro-Winkler, K-means and Soundex — is presented in this subsection's results, each model tested across three differently sized datasets: 500, 10,000, and two million recordings. The construction and ground-truth labelling of the 500- and 2,000,000-record datasets used for these timing comparisons are not detailed in this study; a full description is left for a future extended version.

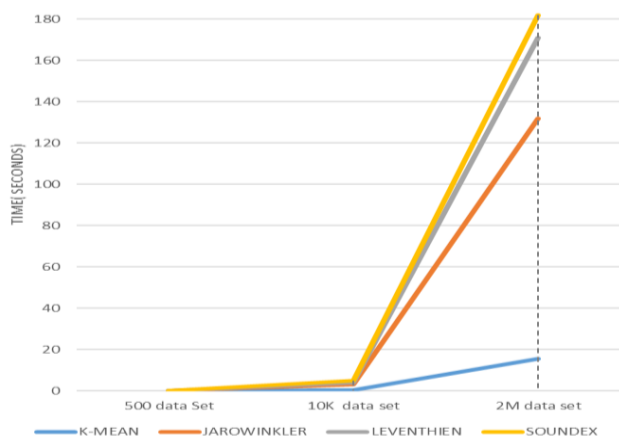


Fig. 11. Model Performance Across Dataset Sizes. Source: Created by the authors.

Fig. 11 reports, for each model, the execution time recorded, with the x-axis giving the record count per dataset and the y-axis the execution time in seconds. k-means' execution time ran to 0.04 seconds on the 500-record dataset, 0.17 seconds on the 10k dataset, and 15.34 seconds on the 2M dataset; Levenshtein's ran to 0.11 seconds, 3.81 seconds, and 171.07 seconds across the same three; Jaro-Winkler's to 0.08 seconds, 2.90 seconds and 132.00 seconds; and Soundex's to 0.14 seconds, 4.87 seconds and 182.04 seconds — a summary of how the four models performed. The measured execution times are summarised in Table 3.

Table 3. Execution Time Comparison (seconds).

Matching Algorithm	500 Dataset	10K Dataset	2M Dataset
Levenshtein	0.11	3.81	171.07
Jaro-Winkler	0.08	2.90	132.00
k-means	0.04	0.17	15.34
Soundex	0.14	4.87	182.04

What the results make clear is that k-means needs far less time than the remaining three models, and that Jaro-Winkler outpaces both Levenshtein and Soundex, whose own performances sit very close together. From fastest to slowest, the models rank as follows:

1. K-MEANS
2. JARO WINKLER
3. LEVENSHTein
4. SOUNDEX.

Certain models turn out, in the sections below, to combine very high performance with very low accuracy — a combination worth examining further.

4.3 Accuracy and Results Comparison

By counting the modifications needed to turn one word into another so that the two match exactly, the Levenshtein Distance model measures how similar two strings are. Table 4's results, for instance, link NAME_ID1 = "MICHAEL" with NAME_ID2 = "MICHKAEL", the model judging the pair similar.

Table 4. Levenshtein Distance Matching Results.

id1	id2	weight	links	NAME_ID1	DOB_ID1	NAME_ID2	DOB_ID2
1099.00	3825.00	38.40	L	MICHAEL WEBER	29 / 9 / 1955	MICHKAEL WEBER	29 / 9 / 1955
1193.00	3277.00	38.40	L	MARKUS NEUMANN	31 / 7 / 1948	MARKGUS NEUMANN	31 / 7 / 1948
1281.00	2672.00	38.40	L	GERHARGD KOERTIG	5 / 3 / 1973	GERHARD KOERTIG	5 / 3 / 1973
1659.00	1735.00	38.40	L	STEFAN MUELLER	26 / 3 / 1938	STEFAND MUELLER	26 / 3 / 1938
1671.00	1734.00	38.40	L	WILHELM KRAUS	26 / 11 / 1973	WILHELIM KRAUS	26 / 11 / 1973

A single inserted "K" is all that separates "MICHAEL" from "MICHKAEL", and the two remain similar despite it. Because the Levenshtein Distance is simply a number capturing how far apart two strings sit, this pair's distance comes to 1, and, generally, the larger that number grows, the further apart the two strings (names) are judged to be. Matching records here total 344. Levenshtein's own accuracy is high, yet its performance trails the other models, since it runs slowly — a longer running time translating directly into weaker performance.

Jaro-Winkler matched 349 records (its highest weight, 38.52, again narrowly exceeding Levenshtein's), while k-means matched 41,532 records by forename alone, regardless of surname or date of birth — the behaviour, detailed in Table 6, that accounts for its low precision.

k-means returns 433,388 non-matching records overall; FRANK MUELLER and FRANK PETERS, for instance, are found unlinked by k-means — evidence enough that k-means, working as it does in an illogical fashion, is not a model fit for solving ER.

Table 5. Soundex Matching Results.

id1	id2	weight	links	NAME_ID1	DOB_ID1	NAME_ID2	DOB_ID2
1099	3825	38.40	L	MICHAEL WEBER	29 / 9 / 1955	MICHKAELE WEBER	29 / 9 / 1955
1193	3277	38.40	L	MARKUS NEUMANN	31 / 7 / 1948	MARKGUS NEUMANN	31 / 7 / 1948
1281	2672	38.40	L	GERHARD KOERTIG	5 / 3 / 1973	GERHARD KOERTIG	5 / 3 / 1973
1659	1735	38.40	L	STEFAN MUELLER	26 / 3 / 1938	STEFAND MUELLER	26 / 3 / 1938
1671	1734	38.40	L	WILHELM KRAUS	26 / 11 / 1973	WILHELM KRAUS	26 / 11 / 1973

Table 5 shows the Soundex model linking two records wherever forename and surname are close (if not identical) and DOB matches exactly, DOB being treated throughout as an identical variable. MICHAEL WEBER and MICHKAELE WEBER, for instance, differ only by an extra K in the second record's forename, and share the same DOB, 29/9/1955; Soundex accordingly reads the pair as the same real-world entity, assigning a very high weight of 38.40. Matching records under Soundex total 344, matching Levenshtein's own count, a result that speaks to Soundex's high accuracy and its place among the models judged efficient for ER.

Across all four models, non-matching records numbered in the hundreds of thousands (e.g. 474,576 for Levenshtein and Soundex, 474,571 for Jaro-Winkler, 433,388 for k-means); in every case the negative weight assigned correctly signalled dissimilar name/DOB pairs such as FRANK MUELLER vs. FRANK WEBER, confirming that each model avoids false links as well as finding true ones.

Table 6 reports the measured accuracy across the four tested models. A True Positive (TP) denotes a record pair correctly classified as a match against the dataset's known duplicate structure; a False Positive (FP) denotes a record pair incorrectly classified as a match. What the results show plainly is that Levenshtein and Soundex stand as the two most accurate models for ER, with TP reaching 474,920 and FP reaching 41,188. Recall and F1-score could not be computed without an independent enumeration of all true matching pairs beyond RLdata10000's embedded duplicate labels; we report precision alone here and identify the full precision–recall trade-off as a direction for future evaluation.

Table 6. Accuracy Performance.

Matching Model	Matching Records	Non-matching Records	True Positive	False Positive	Precision
Levenshtein	344	474,576	474,920	0	100 %
Soundex	344	474,576	474,920	0	100 %
Jaro-winkler	349	474,571	474,916	4	99.99%
k-means	41532	433,388	433,732	41,188	91.33%

Table 6 makes plain that k-means, high-speed as it is, suffers from low accuracy and produces illogical results; the present work, however, seeks a model striking a balance between speed and accuracy rather than maximising either alone. Set against Jaro-Winkler, Levenshtein returns a TP of 474,916 against only four FP records, from which Jaro-Winkler emerges as an efficient model of middling speed and low FP.

Set against Soundex, finally, Levenshtein returns a TP of 474,920 and an FP of zero, marking both out as the two strongest models overall; Levenshtein nonetheless edges ahead of Soundex on speed, needing 3.81 s against Soundex's 4.87 s — a narrow gap, yet one that would matter considerably at big-data scale. Weighing all these results together, Levenshtein stands out as the dependable choice for ER, combining strong accuracy with good speed and logical outcomes throughout.

recid.1	recid.2	NAME_ID1	DOB_ID1	NAME_ID2	DOB_ID2	levenshtein	soundex	jarowinkler	k.mean
469	4291	WERNER SCHMIDT	19 / 10 / 1965	MARTIN SCHMID	19 / 10 / 1965	N	N	L	L
2238	4847	DIETER RICHTER	29 / 9 / 1965	KARL RICTER	29 / 9 / 1965	N	N	L	L
4086	5037	ULRICH HOFMANN	8 / 2 / 1964	BARBARA HOFFMANN	8 / 2 / 1964	N	N	L	L
8375	7581	FRANK MUELLER	27 / 9 / 1967	BARBARA SCHMIDT	5 / 2 / 1966	N	N	L	L
1	950	FRANK MUELLER	27 / 9 / 1967	FRANK BECKER	2 / 9 / 1969	N	N	N	L
1	1075	FRANK MUELLER	27 / 9 / 1967	FRANK SCHROEDER	14 / 9 / 1994	N	N	N	L
1	1948	FRANK MUELLER	27 / 9 / 1967	FRANK KLEIN	25 / 9 / 1980	N	N	N	L
1	2524	FRANK MUELLER	27 / 9 / 1967	FRANK RICHTER	5 / 9 / 1954	N	N	N	L
4	1957	HANS SCHMITT	14 / 8 / 1945	HRANS SCHMITT	14 / 8 / 1945	L	L	L	L
38	3165	HORSTX MUELLER	7 / 9 / 1991	HORST MUELLER	7 / 9 / 1991	L	L	L	L
85	2623	HARALD FUCHS	20 / 11 / 1954	HARAJLD FUCHS	20 / 11 / 1954	L	L	L	L
75	2377	HERMAANN MUELLER	2 / 10 / 1990	HERMANN MUELLER	2 / 10 / 1990	L	L	L	L

4.4 Visualization of Results

SNA and visualisation together serve, in this paper, to examine the relationships holding among the records, as Fig. 12-a shows, with the ER problem rendered as an SNA through network visualisation. Zooming into the network reveals links between nodes, each colour marking a different record, with a given edge capable of encoding one or several variables: first name, last name, DOB. Fig. 12-(b) shows the two blue nodes joined by first name (THORSTEN) and last name (SCHMIDT), while Fig. 12-(c) shows three nodes joined by first name (MONIKA) and last name (HOFFMANN). Through SNA, then, it becomes easier to show how records connect and to reveal which variables render them matching or non-matching. Read against the centrality measures defined in Section 3.3, such clusters of matched records correspond to nodes of elevated Degree Centrality (2), since each shares a direct edge with every other record in its cluster; a node bridging two otherwise disconnected clusters would register a raised Betweenness Centrality (3); the short average path length within a matched cluster corresponds to high Closeness Centrality (4); and a record's centrality is further reinforced under Eigenvector Centrality (5) whenever its neighbours are themselves well-connected duplicates. These measures are illustrated qualitatively through the visualisation in Fig. 12 rather than computed numerically over the full RLdata10000 graph, which we identify as a direction for future quantitative evaluation.

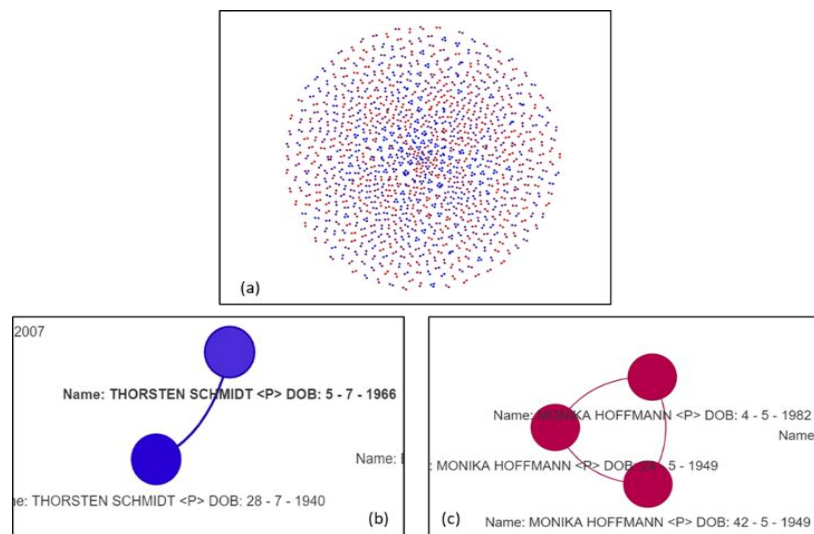


Fig. 12. SNA Visualization of Matched Records. Source: Created by the authors.

5 Conclusion

This study proposes a framework for resolving the entity resolution problem while maintaining record redundancy and high accuracy across datasets. It is based on recasting the dataset as a Social Network graph so that it may

access state-of-the-art analytical measurements; the framework spans several phases that drive entity resolution while stripping out redundancy, so that high accuracy is achieved even over large datasets. Tested on the RLdata10000 dataset, the framework draws on several SNA criteria — Degree Centrality, Betweenness Centrality, Closeness Centrality and Eigenvector — to characterise, qualitatively, the relationships holding among matched data records (Section 4.4); the performance and accuracy of several modelling algorithms — Levenshtein, Jaro-Winkler, K-means and Soundex — were likewise assessed, implemented in R-studio, with the framework evaluated across three datasets (of 500, 10k and 2M records) using all four algorithms. What the results show is k-means running markedly faster than the rest yet trailing badly on accuracy, while, among the remaining three, Jaro-Winkler offers the best execution speed alongside performance sitting very close to the optimal solutions reached by Levenshtein and Soundex. Recasting the data as an SNA further opens up SNA's network-visualisation tools, letting the data be inspected and evaluated visually simply by observing how the records connect.

DECLARATIONS

Funding. The authors received no external funding for this research.

Conflicts of interest. The authors declare that they have no competing interests.

Ethics approval. This study did not involve human participants, human data, or animal subjects requiring ethics-committee approval. The dataset used (RLdata10000) is a publicly available synthetic benchmark distributed with the RecordLinkage package for R.

Informed consent. Not applicable, as the study did not involve human participants.

Data availability. The RLdata10000 dataset used in this study is publicly available within the RecordLinkage package for R.

Author contributions. Mohammad Sh. Daoud conceived and led the study, developing the methodology and software and drafting the original manuscript; he also supervised the project and administered the work. Hussain Al-Aqrabi contributed to supervision, formal analysis, and writing—review and editing. Yousef Q. Saadeh carried out validation of the results. Bayan AbuShawar contributed to investigation and writing—review and editing. Samer Aoudi contributed to data curation and writing—review and editing. All authors read and approved the final manuscript.

Use of artificial intelligence. Generative AI (Claude, Anthropic) assisted with template restructuring, and reference formatting during manuscript preparation. All conception, methodology, data, analysis, and conclusions are the authors' own, who reviewed and approved all AI-assisted text before submission.

Acknowledgements. none apply.

References

- [1] B. H. Li, Y. Liu, A. M. Zhang, et al., “A survey on blocking technology of entity resolution,” *J. Comput. Sci. Technol.*, vol. 35, pp. 769–793, 2020, doi: 10.1007/s11390-020-0350-4.
- [2] D. A. Luke, “Introducing Network Analysis in R,” in *A User’s Guide to Network Analysis in R*, 2015.
- [3] Z. Farkas, “Analysing social structure: A chapter for the outline of social theory,” *Working Paper*, 2017.
- [4] G. Papadakis, E. Ioannou, and T. Palpanas, “Entity resolution: Past, present and yet-to-come,” in *EDBT*, 2020, pp. 647–650.
- [5] I. Bhattacharya and L. Getoor, “Collective entity resolution in relational data,” *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, art. 5, 2007, doi: 10.1145/1217299.1217304.
- [6] N. Koudas, S. Sarawagi, and D. Srivastava, “Record linkage: Similarity measures and algorithms,” *Proc. ACM SIGMOD Int. Conf. Manag. Data*, pp. 802–803, 2006, doi: 10.1145/1142473.1142599.
- [7] S. Gomatam, R. Carter, M. Ariet, and G. Mitchell, “An empirical comparison of record linkage procedures,” *Stat. Med.*, vol. 21, pp. 1485–1496, 2002, doi: 10.1002/sim.1147.
- [8] G. Wang, H. Chen, and H. Atabakhsh, “Automatically detecting deceptive criminal identities,” *Commun. ACM*, vol. 47, no. 3, pp. 70–76, 2004, doi: 10.1145/971617.971618.
- [9] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava, “Text joins in an RDBMS for web data integration,” *p. 90*, 2003, doi: 10.1145/775165.775166.
- [10] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E. Whang, and J. Widom, “Swoosh: A generic approach to entity resolution,” *VLDB J.*, vol. 18, no. 1, pp. 255–276, 2009, doi: 10.1007/s00778-008-0098-x.
- [11] X. Li, S. Fan, J. Yao, and H. Sun, “Contextual semantics graph attention network model for entity resolution,” *Sci. Rep.*, vol. 15, art. 27093, 2025, doi: 10.1038/s41598-025-11932-9.
- [12] A. Zeakis, G. Papadakis, D. Skoutas, and M. Koubarakis, “An in-depth analysis of pre-trained embeddings for entity resolution,” *VLDB J.*, vol. 34, no. 1, art. 5, 2025, doi: 10.1007/s00778-024-00879-4.
- [13] M. Beraha and S. Favaro, “Large-scale entity resolution via microclustering Ewens-Pitman random partitions,” *arXiv preprint arXiv:2507.18101*, 2025.
- [14] V. Adamchik, “*Graphs_1_Print*,” pp. 1–11, 2005.

- [15] S. C. Carlson, "Graph theory: Problems & applications," Encyclopaedia Britannica. [Online]. Available: <https://www.britannica.com/topic/graph-theory>
- [16] A. P. U. Siahaan, S. Aryza, E. Hariyanto, R. Lubis, A. H. Ikhwan, and P. L. E. Kan, "Combination of Levenshtein distance and Rabin-Karp to improve the accuracy of document equivalence level," *Int. J. Eng. Technol. (UAE)*, vol. 7, no. 2 Special Issue 27, pp. 17–21, 2018.
- [17] V. Joshi, "A gentle introduction to graph theory," *basecs*, Medium, Mar. 20, 2017. [Online]. Available: <https://medium.com/basecs/a-gentle-introduction-to-graph-theory-77969829ead8>
- [18] J. L. Antonakos and K. C. Mansfield, Jr., *Practical Data Structures Using C/C++*. Prentice-Hall, 1999, pp. 165–190.
- [19] W. J. Collins, *Data Structures and the Java Collections Framework*. New York: McGraw Hill, 2005 [2002], pp. 239–303.
- [20] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, Feb. 1966.
- [21] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, no. 1, pp. 31–88, 2001.
- [22] W. E. Winkler, "String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage," in *Proc. Section on Survey Research Methods, American Statistical Association*, 1990, pp. 354–359.
- [23] B. Leonardo and S. Hansun, "Text documents plagiarism detection using Rabin-Karp and Jaro-Winkler distance algorithms," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 5, no. 2, pp. 462–471, 2017, doi: 10.11591/ijeecs.v5.i2.pp462-471.
- [24] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, 2003.
- [25] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *J. R. Stat. Soc. Ser. C (Appl. Stat.)*, vol. 28, no. 1, pp. 100–108, 1979.
- [26] Y. Li and H. Wu, "A clustering method based on K-means algorithm," *Phys. Procedia*, vol. 25, pp. 1104–1109, 2012, doi: 10.1016/j.phpro.2012.03.206.
- [27] B. Thieurmél and Almende B.V., "visNetwork: Network visualization using 'vis.js' library," R package, 2022. [Online]. Available: <https://CRAN.R-project.org/package=visNetwork>
- [28] K. Das, S. Samanta, and M. Pal, "Study on centrality measures in social networks: A survey," *Soc. Netw. Anal. Min.*, vol. 8, no. 1, pp. 1–11, 2018.
- [29] J. Golbeck, "Analyzing networks," in *Introduction to Social Media Investigation*, Elsevier, 2015, pp. 221–235.
- [30] Y. Cheng and D. Suthers, "Social network analysis: Centrality measures," 2011.
- [31] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, no. 1–7, pp. 107–117, 1998, doi: 10.1016/S0169-7552(98)00110-X.
- [32] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecast.*, vol. 22, no. 4, pp. 679–688, 2006, doi: 10.1016/j.ijforecast.2006.03.001.
- [33] W. Chang, J. Cheng, J. J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges, "shiny: Web application framework for R," R package, 2021. [Online]. Available: <https://CRAN.R-project.org/package=shiny>
- [34] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "REGAL: Representation Learning-based Graph Alignment," in *Proc. 27th ACM Int. Conf. Information and Knowledge Management (CIKM)*, 2018, pp. 117–126, doi: 10.1145/3269206.3271788.