IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 14, 2021, accepted October 13, 2021, date of publication October 25, 2021, date of current version November 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3122451

Generating Name-Like Vectors for Testing Large-Scale Entity Resolution

SAMUDRA HERATH[1,2], MATTHEW ROUGHAN[1,2], (Fellow, IEEE), AND GARY GLONEK[1]
[1]School of Mathematical Sciences, The University of Adelaide, Adelaide, SA 5005, Australia
[2]Australian Research Council (ARC) Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS), Parkville, VIC 3010, Australia

Corresponding author: Samudra Herath (samudra.herath@adelaide.edu.au)

This work was supported in part by the Data to Decisions (D2D) CRC, in part by the Cooperative Research Centres Programme, and in part by the ARC Center of Excellence for Mathematical and Statistical Frontiers (ACEMS).

ABSTRACT Entity resolution (ER), the problem of identifying and linking records that belong to the same real-world entities in structured and unstructured data, is a primary task in data integration. Accurate and efficient ER has a major practical impact on various applications across commercial, security and scientific domains. Recently, scalable ER techniques have received enormous attention with the increasing need to combine large-scale datasets. The shortage of training and ground truth data impedes the development and testing of ER algorithms. Good public datasets, especially those containing personal information, are restricted in this area and usually small in size. Due to privacy and confidential issues, testing algorithms or techniques with real datasets is challenging in ER research. Simulation is one technique for generating synthetic datasets that have characteristics similar to those of real data for testing algorithms. Many existing simulation tools in ER lack support for generating large-scale data and have problems in complexity, scalability, and limitations of resampling. In our work, we propose a simple, inexpensive, and fast synthetic data generation tool. Our tool only generates entity names in the first stage, but these are commonly used as identification keys in ER algorithms. We avoid the detail-level simulation of entity names using a simple vector representation that delivers simplicity and efficiency. In this paper, we discuss how to simulate simple vectors that approximate the properties of entity names. We describe the overall construction of the tool based on data analysis of a namespace that contains entity names collected from the actual environment.

INDEX TERMS Entity resolution, data integration, data linkage, data matching, information systems, large-scale synthetic data, record linkage.

I. INTRODUCTION

Data integration plays a vital role in data analysis and mining projects by combining data from different sources into meaningful information. Entity resolution (ER), a core step in data integration, detects entity records across multiple databases that correspond to the same real-world entity. It applies to various data types, from structured relational databases to unstructured entities extracted from free text [1], [2]. ER has also been known as the object identity problem, record linkage, the merge/purge problem, deduplication, duplicate record detection, and data matching in different domains.

ER has been widely recognised in academic and statistical research since research data are gathered from multiple data sources that store data in different formats. This process is also of increasing importance in commercial and government practice. ER appears as a problem in a range of applications, e.g., medical or epidemiological research, crime detection and national security, tax and other fraud detection, education research, e-commerce applications, and customer relationship management [1].

The problem of combining two or more separately recorded pieces that belong to a particular individual dates back to the early 1950s [3]. Despite extensive research studies over several decades, ER remains a challenging problem in practice, especially for applications in big data. Big data create new challenges such as high scalability, complex similarity metrics, and advanced data quality evaluation requirements for ER. Hence, high-performing scalable ER techniques are required to facilitate the integration of

The associate editor coordinating the review of this manuscript and approving it for publication was Vijay Mago.

large-scale data collections. Simultaneously, big data ER applications require large-scale testing data when developing suitable ER solutions.

We explain a simple illustrative instance of the ER problem through the contact tracing process in the novel coronavirus (COVID-19) pandemic. Health and surveillance systems currently use contact tracing to determine where patients have caught the virus and whom they might have infected. The process tracks patient movements back to the potential person from whom they caught the virus to identify people who may have been in contact with the infected person. It requires data matching between different databases, especially when only partial information is available about a person. Most countries have manual contact tracing systems that could easily break with a few hundred cases during a massive outbreak. Contract tracing could involve comparing hundreds of personal information items against a large population within a day; it is impossible to accomplish this task with a manual system. Hence, developing and applying suitable big data ER techniques can be a promising strategy to accommodate efficient and effective contact tracing.

Many ER applications have databases that contain records of individual entities such as patients, customers, travellers, etc. Often, there are no unique identifiers for entities that would make record matching easy in those databases. Therefore, record matching relies on the entity attributes that are common across databases. For instance, the above example of contact tracing strongly depends on the information available about a person. Examples of entity identifying attributes based on personal information are name, address, and date of birth. However, obtaining such real datasets is difficult for research with growing concerns over data privacy and identity theft. Even if we are allowed to use such datasets for testing our algorithms, they cannot be published for replication studies. As a result, developing novel algorithms that process and integrate data that contain personal identifying information is often difficult. This lack of data is a fundamental problem of ER research.

There are several large-scale, real RDF (resource description framework) web datasets that could be used as ER benchmarks [4]. However, only a few real datasets that are available for research contain personal information, and these are small in size [1], [5]. Even with the available real datasets, the correct status of two records is unknown to many applications. The main reason is the lack of ground truth or ''gold-standard'' data specifying whether two records correspond to the same entity. Therefore, validating matched and linked results in the testing of newly developed ER algorithms is very difficult. Data simulation is an alternative approach in the absence of suitable datasets. Our motivation is to mimic the characteristics of large-scale data containing personal information that are useful in deduplication, fraud detection, cloud computing, and health informatics in our simulation model to generate large-scale datasets [1].

There are many advantages in using simulations as part of the development process, where algorithms will ultimately be tested on real datasets [6]. A key benefit of simulated data is that they contain ground truth data critical to evaluating ER algorithms. We can also control the properties such as size, linking rate, types of errors, and error rates in data simulation. Moreover, simulated datasets can be published with their source programs/tools, allowing others to adapt them in their application domains.

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

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

Our method holds several attractive properties compared to traditional simulation tools. These simulation tools resample from real databases or datasets that contain information about entities. Thus, the size of the existing databases limits the size of the datasets a particular tool can generate effectively. In contrast, our solution relies on a vector representation that defines records, where the capacity of resampling from vector space is unlimited. A vector can be uniquely located in this space, and the distance computations between them can be done efficiently using L^p norms. Due to the availability of operations such as vector addition and subtraction in this space, new vectors can be constructed from prior vectors equally easily [7].

Error modelling is an essential and complex component of test data simulation in ER since errors are inherent in real-world databases. The quality of entity identifying information in real data can be low for incorrect, incomplete or outdated details that could lead attribute values to be erroneous. Generating those errors in data simulation requires decisions such as types of errors (e.g., spelling or phonetic errors) and the probabilities of introducing and positioning errors. However, the most common type of errors considered

in many simulation tools is typographical errors. Our simulation model generates typographical errors of names using only one parameter: the variance of errors in name-like vectors that approximates the underlying namespace. We introduce small typographical errors with small distances away from name-like vectors, and the same could be applied to generate large errors to simplify the error simulation process.

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

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

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

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

Three surname datasets are used as inputs for the experimental data analysis. The results closely mirrored each other (details in Section V). The proposed numerical simulation model uses a normal distribution to generate name-like vectors in a low-dimensional Euclidean space (6 to 8 dimensions).

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

The contributions of this paper are as follows:

- A name approximation method that maps actual names into vectors of a Euclidean space approximating the dissimilarities or distances between name stings. The technique can be applied to any set of strings, not just names.
- A numerical simulation model to generate name-like vectors and their errors based on name dissimilarities of an actual namespace. We avoid the detail-level simulation of entity names using a simple vector representation that delivers simplicity and efficiency.
- A workflow to construct, test and evaluate the proposed simulation model following data analysis of a namespace. We provide a simple workflow of our simulator that is easy to follow, where users can generate data vectors using their input datasets. We recommend using names or similar strings as input data. We also explain how a user can easily extend this tool to different features.

II. BACKGROUND AND RELATED WORK

The problem of entity resolution using computers started in the early 50s when Newcombe *et al.* [3] proposed the automation of ER. Fellegi and Sunter [8] described a framework for probabilistic ER in their seminal paper in 1969. Later, many practical applications followed this framework instead of manual or ad hoc linking techniques. Christen [1], in his book, provided an overview, one of the comprehensive sources of entity matching.

Some of the top surveys found in the literature discuss the pressing issues of ER, e.g., privacy concerns, big data, and existing ER solutions. Vatsalan *et al.* [9] presented a survey of techniques that match and link databases between organisations considering the privacy aspects of the data. Christophides *et al.* [2] reviewed ER techniques in the context of big data, whereas Barlaug and Gulla [10] provided a survey of deep neural networks in entity matching. Several other surveys discuss narrower aspects of ER, such as specific techniques or subtasks. In this section, we survey only a few relevant works that align with the focus of our work.

There are frequently used datasets in the ER literature: a health data set containing midwife data records [11], the North Carolina Voter database (NCVR) [12] and CORA,[1] a citation network. Besides, only a few 'real' datasets that contain personal information are available for research. However, each of these has a limited size, problems in resampling, and limited control over errors.

Arehart and Miller [5] developed a ground truth dataset that contains approximately 70,000 culturally diverse Roman names. The dataset includes personal names that were drawn from different sources and manually incorporated name variations.

Several simulation tools have been developed in ER literature. These tools aim to simulate realistic records

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

and common errors. Hernandez and Stolfo [13] presented the first work on data generation with duplicate records based on real-world error distributions. Bertolazzi *et al.* [14] addressed some limitations of Hernandez's method, including missing values and various corruption functions.

Christen and Vatsalan [6] presented a tool that generates records with entity attributes such as names, addresses, and dates of birth, which has been widely used in ER research. However, it does not support large-scale data generation. Talburt *et al.* [15] presented a tool for generating records that represent people's residential occupancy histories. Several other works on simulation studies were proposed by Tromp *et al.* [16] and Bachteler and Reiher [17]. These simulation tools are often application-specific, where the researchers proposed various ad hoc methods that generate datasets with specific characteristics.

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

The frequency and look-up tables used in the record linkage system, FEBRL (freely extensible biomedical record linkage) [18], have been regularly used by other simulation studies to generate records. Although this is a broadly used open-source approach, it is not scalable enough to explore ER algorithms that work with very large datasets.

Our work is motivated by the vector-based representation of words, which has a thriving history in the applications of information retrieval, computational semantics, and natural language processing. The underlying property is that words with similar contextual significance tend to have similar embeddings when mapped to a continuous vector space [19]. Word2vec [20], in particular, has been used extensively in natural language processing applications. However, it is less useful in proper-noun databases since *names* do not have semantic relationships [21].

Several methods that embed a set of strings in a metric-space have been used to explore the problems in ER [22], [23]. These methods focus on matching similar records efficiently using the properties of the metric-space. However, to the best of our knowledge, none of them addressed the data simulation using a metric-space, particularly a vector space.

Entity alignment (EA) models represent entities as low-dimensional vectors in knowledge graphs according to semantic or structural information. While EA and ER have similar goals to identify entity records of the same entity, EA differs from ER as it operates on graph-structured data.

Several mapping algorithms, including StringMap [22], FastMap [24], SparseMap [23], and MetricMap [25], are proposed for embedding a set of objects in a metric-space. We choose multidimensional scaling (MDS) because of its ability to work with the nonlinear and nonmetric nature of our data and because it includes a distance preservation capability for large-scale data with a small amount of extra effort.

The outcome of a general MDS algorithm is a map that conveys spatial relationships among input objects, which are usually dissimilarities between objects. This map is expected to preserve the original proximities such that similar items are close by and dissimilar items are further apart by their relative locations [26]. Different variations of the generic MDS algorithm are found in the literature. For an overview, see [27], [28]. We will discuss the technical details of the MDS algorithms in Section IV.

III. PRELIMINARIES

This section introduces relevant definitions, concepts and notations used throughout the paper. First, we present key definitions of the entity resolution (ER) problem. Next, we introduce preliminary concepts of domain-specific distance metrics used in attribute-level ER matching.

An *entity* can be a person, place, product, organisation or any object with a unique identity that distinguishes it from all other entities of the same type. A collection of name-value pairs that describe a particular entity is known as an *entity profile* [2]. A pair of similar entity profiles are called *duplicates*. A duplicate of an entity can be either an exact copy of the original entity profile or an entity profile that contains an error (e.g., typographical error). A database representation of an entity profile is usually referred to as a *record*.

A set of entity profiles (or simply, records) is called *entity collection*, denoted by E. Given E, we say two records r_1, r_2 *match* if they refer to the same unique real-world object. We denote this as $r_1 \equiv r_2$. The goal of ER is to link different records that describe the same entity within an entity collection or across two or more entity collections.

Matching records of an entity are determined by pairwise comparisons between records in entity collections. Given a pair of records (r_i, r_j) and a set of attributes $a_1, a_2, \ldots, a_x$ that describe them, similarities $s_1, \ldots, s_x$ between attribute values are determined by applying a set of similarity functions $sim_k(r_i.a_k, r_j.a_k)$, with $1 \leq k \leq x$. The total similarity between the records is given by the similarity score $S = \sum_{i=1}^{x} s$. The record pairs can be classified as a match or a non-match using the similarity scores and a matching threshold [23].

Several comparison methods such as edit distance, a.k.a. Levenshtein distance, Jaro distance, and q-gram distance are found in the domain of strings [29]. In this work, we mainly used the Levenshtein distance to measure the similarity at the attribute level. It calculates the minimum number of character insertions, deletions, and replacements necessary to transform a string s_1 into a string s_2. Minkowski metrics based on L^p norms, $\| x \|_p = (\sum |x_i|^p)^{1/p}$, with $p \geqslant 1$ are popular in the metric domains. In metric-space calculations, we used the most common Minkowski metric, Euclidean distances $d_E(p = 2)$.

IV. METHODOLOGY

Our methodology is organised into three main sections. In each section, we answer the questions defined in Section I.

A. CONVERTING NAMES TO MULTIDIMENSIONAL VECTORS

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

We cast the problem as an embedding problem, which refers to constructing coordinates from distances. In domains that use computationally expensive distance functions, significant speed-ups can be obtained by embedding objects into a coordinate space and applying an efficient distance function, such as the L^p norm [7].

Assume R is a collection of objects, δ measures the distances between R objects, X represents the coordinates matrix for the R objects in the Euclidean space, and d measures the distances between coordinates. Embedding of a metric or non-metric space (R, δ) into a coordinate space (X, d) is a mapping $\phi : R \rightarrow X$. In this work, (R, δ) will always be a finite space (i.e., R is a finite set) and (X, d) will always be a Euclidean space.

The most commonly used technique for embedding a set of distances (or dissimilarities) into a coordinate space is multidimensional scaling (MDS) [30]. MDS is a non-linear optimisation problem for finding the best mapping of ϕ [30]. Using MDS, we can project names into a lower-dimensional Euclidean space by approximating their dissimilarities. Then the pairwise comparisons between names become Euclidean distances between vectors. This lower-dimensional representation of namespace allows us to study the structure and explain the relationships of the data mathematically.

An MDS algorithm takes the input of N points as a $N \times N$ matrix Δ, a distance or affinity matrix. An entry in the i^{th} row and j^{th} column of Δ is the dissimilarity δ_{ij}, between object i and object j, where $\delta_{ii} = 0$, and $\delta_{ij} > 0, i \neq j$. Then it attempts to construct a configuration of N data points $x_1, \ldots, x_N$ in p dimensions, such that if d_{ij} denotes the Euclidean distance between x_i and x_j and D is their distance matrix, then D is similar to Δ [31].

One of the issues in some uses of MDS is that some subsequent actions might not be invariant to orientation or position, and MDS does not guarantee a particular orientation. However, we are mainly concerned about relative distances in entity matching; therefore, our tools are invariant to orientation and position.

MDS is a standard embedding technique we used as the basis of our simulation model. Unlike many embedding techniques, MDS algorithms can handle both metric and non-metric data. MDS has been used for non-metric data embedding in the application areas of clinical psychology, sociology, marketing, ecology, biology, and image processing [26], [28], [32]. Most of our input data is not Euclidean. For instance, string dissimilarities between names are not always distances, i.e., metrics. Among the variants of MDS, we use the least-squares multidimensional scaling (LSMDS) approach, as it worked best with the nonlinear and non-metric nature of our data.

1) LEAST-SQUARES MULTIDIMENSIONAL SCALING

The standard method in LSMDS is to minimize the raw stress (σ_{raw}) of a configuration. The input is a dissimilarity matrix $\Delta = [\delta_{ij}]$, where i and j are the indices of two data points in a high-dimensional space and δ_{ij} is the distance between them. In the context of names, we refer to *dissimilarities* because some measures are not strictly distance metrics. For a given dissimilarity matrix Δ, a configuration matrix $\mathbf{X} = [x_1, x_2, .., x_N]$, where $x_i \in \mathbb{R}^p$ is constructed by minimizing

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

The Euclidean distance between the i^{th} and j^{th} points in the lower-dimensional space is given by d_{ij}, and w_{ij} denotes the possible weights for each pair of points. These nonnegative weights indicate the importance of the residuals $d_{ij}(\mathbf{X}) - \delta_{ij}$ of object pair ij. Weights are useful in handling missing values, and the default values are $w_{ij} = 0$ if δ_{ij} is missing and $w_{ij} = 1$ otherwise [33]. We do not apply weights in this work; hence, $w_{ij} = 1$ always.

One can use squared distances or a normalised raw stress criterion as well. We prefer the normalised stress (σ) in our experiments since it is popular and theoretically justified [34]. The normalized stress σ, is obtained by

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

In general, LSMDS has been used to find a visual representation of high-dimensional data in 2-3 dimensions. However, we use LSMDS to determine a simple vector representation in a lower dimension for approximating the characteristics of a namespace. We call this an approximated Euclidean namespace that contains *name-like vectors*.

To that end, we need to define an appropriate dimension p, in which distances are maintained at a suitable level of accuracy. We used two approaches to discover a reasonable lower-dimension that fits our data. First, we analysed the rate of change of the normalised stress σ against p-dimensions, where a lower σ value indicates a better approximation. Second, we used a Shepard diagram [35] to determine the goodness-of-fit of LSMDS results. A Shepard diagram is a popular way of assessing the goodness of fit of data reduction or embedding techniques such as MDS and t-SNE (t-distributed stochastic neighbour embedding) [35].

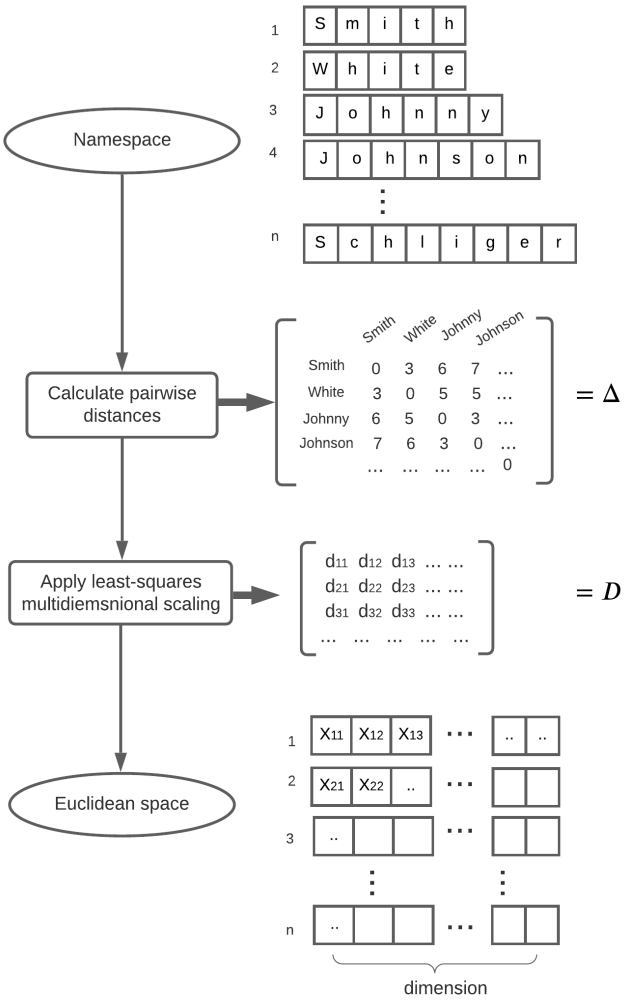

FIGURE 1. **The basic workflow of our numerical simulation model. Names come from a complicated high dimensional space, and we can represent them in a lower-dimensional Euclidean space by applying LSMDS to name dissimilarities. For instance, Δ represents the dissimilarity matrix containing the Levenshtein distance between the given names. By getting Δ as the input, LSMDS project the names into a lower-dimensional space, approximating their initial Levenshtein distances by Euclidean distances, i.e., $\Delta \approx D$. Then, the names become vectors in p dimensional space, where p must be chosen.**

It compares how far apart our data points are before and after the transformation using a scatter plot. We present the results in Section V.

Fig 1. explains the basic workflow that creates the foundation of the proposed simulation model using an example of how textual similarities of strings relates to vector similarities. For a given set of names, Δ represents the dissimilarity matrix of Levenshtein distances. By applying LSMDS, we project the name strings into a lower-dimensional space such that $\Delta \approx D$. We choose the dimension p of the lower-dimensional space that provides the best compromise for our data using the stress function (2).

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

B. SIMULATING NAME-LIKE VECTORS

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

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

1) MULTIVARIATE NORMAL CHARACTERISTICS

Many parametric multivariate statistical methods, such as linear discriminant analysis, require the data to hold the multivariate normality (MVN) assumption. These methods produce more reliable results if the data are exactly or even approximately multivariate normal.

Many analytical and graphical methods test the goodness-of-fit of a dataset to the multivariate normal distribution. However, choosing which test in practice is difficult since different approaches may give different conclusions about the MVN of a dataset. Therefore, it is usually recommended to perform several tests while examining the results produced by graphical methods [36], [37].

We applied Mardia's skewness and kurtosis statistical tests to check the multivariate normality of our data. Additionally, we used graphical approaches such as histograms, density plots, and quantile-quantile plots (Q-Q plots) to visually assess the MVN characteristics of the data [37]. Moreover, we applied a multivariate outlier detection method based on the Mahalanobis distance. It demonstrates how far an individual data point is from the centroid of all points for the underlying variables. An observation is considered an outlier when the distance is great [38].

In addition to MVN, we applied univariate normality tests and plots to diagnose any deviation from normality. Investigating the univariate normality of the underlying variables builds a foundation for a complete understanding of MVN. In the univariate setting, normality tests such as Kolmogorov-Smirnov (K-S) and Shapiro-Wilk are extensively applied in practice [37]. Furthermore, we used histograms and Q-Q plots to assess univariate normality visually.

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

distribution is acceptable as an adequate approximation to simulate name-like vectors.

2) PARAMETER SELECTION AND SIMULATION

Name-like vector simulation is a two-step procedure. The first step is estimating the parameters (mean and variance) for the normal distribution that we would choose as our simulation model. Since our data are multivariate, we calculated the sample covariance matrix (Σ) [39]. All these parameters are estimated from an approximated namespace that contains the name-like vectors.

The second step is the simulation of name-like vectors using the estimated parameters of the normal distribution $N(\bar{x}, \Sigma)$, where $\bar{x} = 0$. In this way, we can generate many name-like vectors within seconds. Hence, our simulation model is a normal distribution-based numerical vector generator that can produce large-scale data.

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

C. ERROR SIMULATION IN NAME-LIKE VECTORS

Real-world datasets have data quality issues, including incompleteness, incompatible formats, and errors [1]. Hence, a namespace can contain errors such as typographical, OCR, and phonetic errors. We wish to include some of these errors in our simulated namespace. Consequently, we come to the subquestion: ''How do we model errors in a simulated namespace?''

When simulating real name strings, the typical process would be to build a complicated error generating model. In contrast, we create errors of simulated name-like vectors by adding some noise to the simulated namespace.

Errors are variations of original names. Distances between a name and its variants are typically small compared to the distances between distinct names. Similarly, the typical distance between a name-like vector and that containing an error should be small. In our error model, we capture the variance of vectors that represent original names and their errors. Then, we use it to simulate errors of name-like vectors with a small Euclidean distance away from their initially simulated name-like vectors. Our prime interest is in edit distance errors such as insertions, substitutions, deletions, and transpositions since they are the most common errors in real datasets [40].

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

1) MAXIMAL RATIO OF COVARIANCES

The maximal ratio of the covariance test is a useful way to report the differences between two variance-covariance structures [41]. The value of the ratio quantifies the two covariances. We compare the covariance of the two groups: approximated errors of name-like vectors and name-like vectors.

Let Σ_s be the $N \times N$ covariance matrix of name-like vectors and Σ_e be that of errors of name-like vectors, and assume both matrices are invertible. Given that $\Sigma_s^{-1/2}$ is the inverse square root matrix, the ratio can be written as a product of one covariance matrix and the inverse of the other covariance matrix ($\Sigma_s^{-1/2}\Sigma_e\Sigma_s^{-1/2}$). Relative eigen analysis or relative principal component analysis determines the direction along which the ratio of variances between two groups is a maximum [41]. This direction is the first eigenvector of $\Sigma_s^{-1/2}\Sigma_e\Sigma_s^{-1/2}$, also known as the first relative eigenvector of Σ_e with respect to Σ_s. The eigenvalues of $\Sigma_s^{-1/2}\Sigma_e\Sigma_s^{-1/2}$ are called the relative eigenvalues of Σ_e with respect to Σ_s. Thus, the first relative eigenvalue γ_1 is equal to the maximal ratio of variances. Small γ_1 values indicate that even in the worst-case scenario, the errors have small variances.

If we can find a small γ_1 for approximated name-like vectors and their errors, it implies that we can add Gaussian noise to simulate random errors in a simulated namespace. However, our experimental results supported this expectation. We calculated the Gaussian noise through a second normal (Gaussian) distribution $N(0, \Sigma_e)$, different from the initially estimated normal distribution. We will discuss the results in the next section.

V. EXPERIMENTAL RESULTS

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

A. THE DATASETS

We used three datasets of surnames to explore namespaces and their properties, e.g., simple relationships such as the distributions of names and the similarities between name strings. In general, all datasets exhibit the same characteristics. For instance, they follow a Zipf's distribution [42].

The first dataset is from Ancestry.com [43], which contains the 250k most commonly occurring surnames. It is the largest unique surname repository we found for this study.

The second dataset is derived from the frequency files included in the dataset generator program in FEBRL [18]. The frequency table contains 9000 unique surnames extracted from telephone directories in Australia.

The third data is from the US census [44] that contains all surnames occurring 100 or more times in the 2010 census. It includes 150,000 distinct surnames.

B. EXPERIMENTAL SETUP

This section presents the results of the ideas introduced in the previous section to real data, particularly the results based on the first dataset. Since our data analysis observations and the simulation results were similar across the three datasets, we will not discuss them separately.

In this paper, we used 5000 randomly selected surnames for testing, which allowed us to perform resampling to evaluate the results for consistency. The outcome of the proposed simulation model does not depend on the individual observations of the selected random sample. Hence the sample selection is flexible. Initially, we started with a distinct set of names, as we first attempted to simulate name-like vectors without errors. However, later on, the simulation of errors for name-like vectors uses a dataset that contains real-world errors.

We implemented LSMDS by applying iterative gradient descent [45] on the stress function (2). Some MDS applications have used other implementations for LSMDS based on the SMACOF (stress majorization of a complicated function) algorithm. For MDS, majorization was introduced by De Leeuw and Mair [46].

We compared the stochastic gradient-descent (SGD) algorithm and the SMACOF algorithm. Both algorithms produced similar results with the same σ values for different configurations and input data sizes. However, we found SMACOF is comparatively slow for our data. Hence, we use SGD-based LSMDS in the rest of the experiments.

One can use a range of standard string distance measures to calculate the pairwise dissimilarities between surnames. We used the STRINGDIST R package [29] to calculate the dissimilarities between name strings. Initially, we considered five frequently used string dissimilarity measures [47], [48]: Levenshtein distances (LV), longest common subsequent (LCS), Jaccard dissimilarity, Q-grams, and Jaro-Winkler distances.

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

C. STRESS ANALYSIS

Fig. 2 shows the stress-σ against dimension-p. The stress σ (2) tends towards a small but nonzero asymptote for most dissimilarities, reflecting the non-Euclidean nature of the original data. The stress values of the LV and LCS distances are overlaid, and it is hard to visualise them separately.

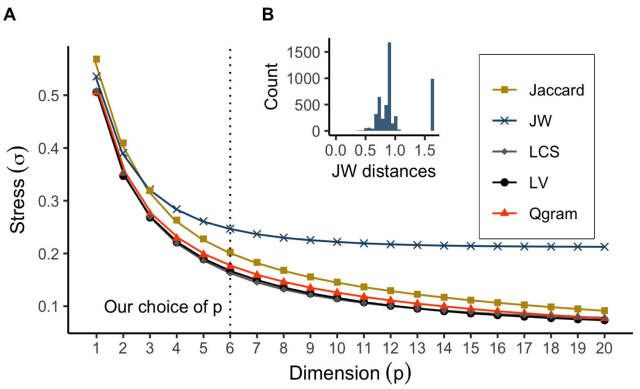

FIGURE 2. A) The trade-off between *stress* and the *dimension* for JaroWinkler (JW), Jaccard coefficient, Longest common subsequence (LCS), Levenshtein (LV) and Qgrams. B) The distribution of JW distances. Compared to the other distributions, JW distances exhibit a spike where all the dissimilar surnames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

The JW distances illustrated in the inset of Fig. 2 are different because JW tends to rate all dissimilar names as being the same distance apart, leading to large nonlinearity in LSMDS. Hence, we argue that the approximation of a namespace in the Euclidean space using LSMDS works for all but the JW dissimilarities.

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

However, when we have non-zero σ, we should keep in mind that the distances among vectors are imperfect and distorted representations of the relationships given by our data. The LMDS transformation does not need to have zero σ to be useful in our application since we are willing to tolerate a certain degree of distortion. There are different standards regarding the amount of σ to tolerate [30]. Our rule is that any σ value under 0.1 is excellent, and anything over 0.2 is unacceptable.

D. SHEPARD DIAGRAM ANALYSIS

An accurate dimension reduction or embedding technique will produce a straight line that goes through the origin in a Shepard diagram [35]. However, in practice, Shepard diagrams rarely look straight due to information loss during data reduction.

Fig. 3 compares the Euclidean distances against the LV distances in 6-p. There is a strong linear relationship between the two distance distributions. Hence, we confirmed that LSMDS successfully approximated the LV distances between real names into the Euclidean distances among the name-like vectors. Since the σ is not zero, there are some errors in the approximation.

Considering the trade-off between the simplicity of the model and the accuracy of the reproduced distances, we selected the 6-p Euclidean space for our simulation model.

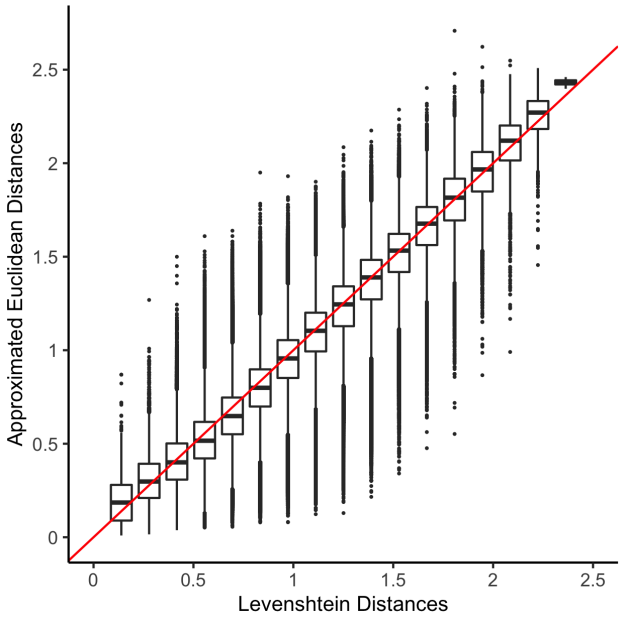

FIGURE 3. The Shepard diagram [35] of the LV distances vs Euclidean distances in 6-*p*. There is a strong linear relationship between the two distance values as desired.

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

E. MULTIVARIATE NORMAL ANALYSIS

We first evaluated the multivariate normality (MVN) of the name-like vectors in Euclidean space. The combined test results of the multivariate skewness and kurtosis tests failed to indicate MVN for the name-like vectors. We used a Q-Q plot (see Fig. 4) to understand the failures of the tests. It depicts several large Mahalanobis distances (outliers) that imply possible departures from the MVN distribution, particularly in the upper tail. However, the body of the distribution mainly lies on the $y = x$ line, suggesting that a normal distribution is a reasonable model for most of the data.

To diagnose the reason for the deviation from MVN, one can perform univariate tests on each variable distribution. In addition, checking univariate plots is also very useful. However, Kolmogorov–Smirnov (K-S) and Shapiro–Wilk tests [37] rejected the normality assumptions for each of the variables in the name-like vectors.

Fig. 5 shows the histograms corresponding to the distribution of each variable in the approximated name-like vectors. The relevant Q-Q plots are not included due to space limitations. However, the histograms and Q-Q plots suggested that the normal distribution reasonably well approximates variables 5 and 6. The other variables have slightly skewed distributions where the body still follows a moderately normal distribution.

These results suggested that the problems with MVN seem to be relatively minor for name-like numbers. Even though

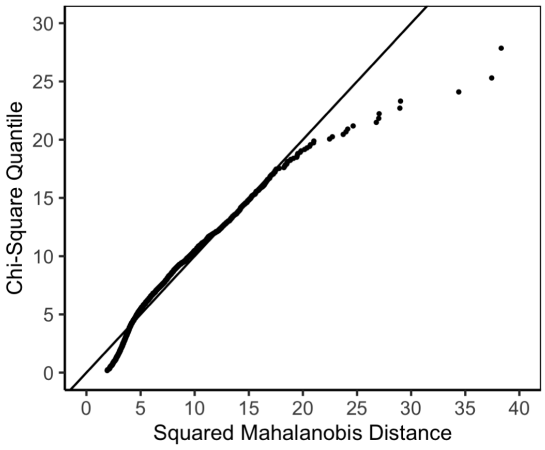

FIGURE 4. The chi-square Q-Q plot for the distribution of all 6-dimensional vectors. There are some deviations from the straight line, indicating possible departures from a multivariate normal distribution.

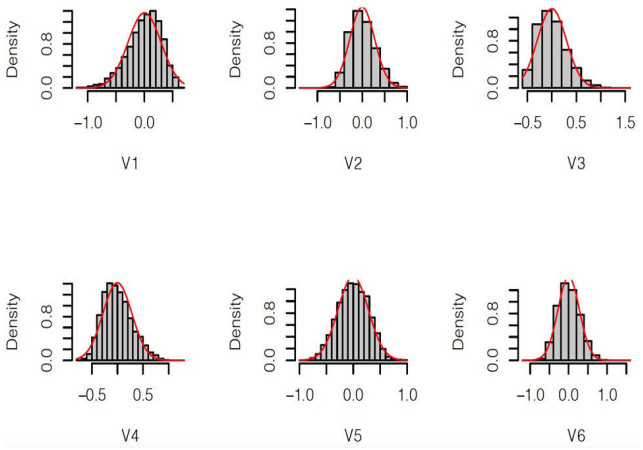

FIGURE 5. Histograms of the approximated coordinate vectors. The variables *V*5 and *V*6 of the coordinates show an approximately normal distribution, while the others have slightly skewed distributions.

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

F. NAME-LIKE VECTOR SIMULATION

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

The calculated covariance matrix has values very near zero on off-diagonals. We tested for independence between each variable using a correlation test and Hoeffding's D statistics [49]. Both approaches verify that we can assume independence between the 6-p variables.

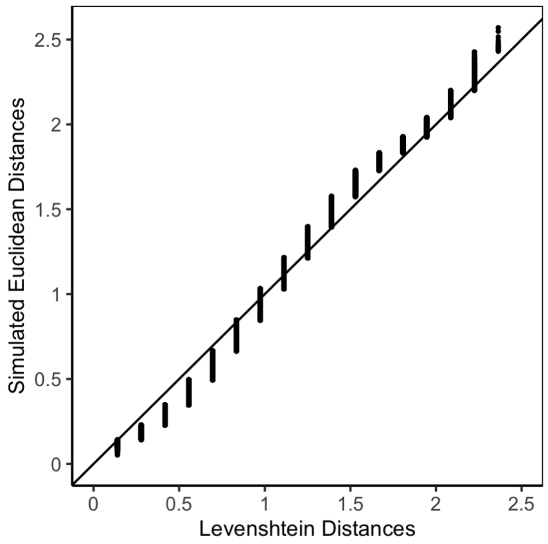

FIGURE 6. Q-Q plot comparing the distributions of LV distances with the Euclidean distances of simulated vectors. Points along the straight line indicate a similarity agreement between the two distributions.

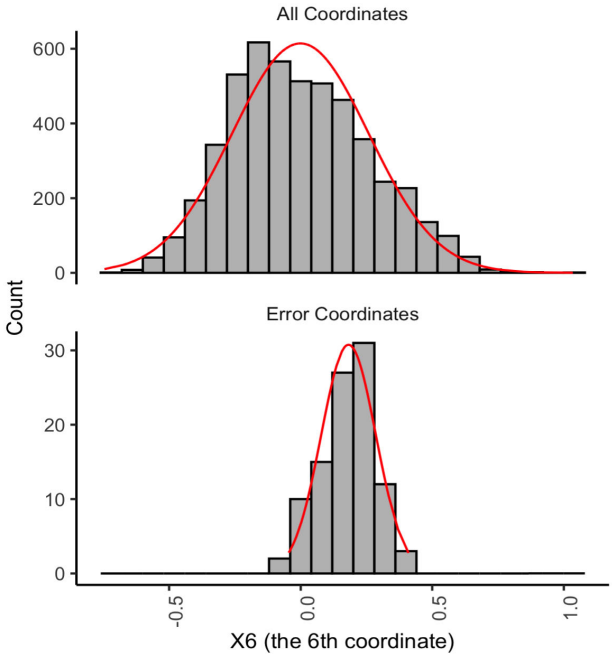

FIGURE 7. The top histogram shows the spread for a single coordinate ($X6$: the 6^{th} coordinate of a name-like vector) in the whole sample (all the surnames and their errors) in 6-p. The bottom row shows the spread of the same coordinate that considers a single surname and its errors. The spread of errors is much smaller compared to the spread of surnames and their errors as desired.

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

Fig. 6 shows a Q-Q plot that compares distributions of LV distances between surnames and Euclidean distances between simulated name-like vectors. The Q-Q plot exhibits a reasonable similarity between the distributions even if they do not precisely fall along the $y = x$ line. We can see the discrete nature of the LV distances and the continuous Euclidean distances along with the points of the small vertical lines. There is a distortion created through the approximation process, but apart from this, the simulated vectors behave very similarly.

G. ERROR SIMULATION

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

A random sample of 25 surnames was selected from the initial dataset of surnames, and 100 variations of edit distance errors were generated for each of them. The errors for the original surnames are generated using FEBRL [18]. Then, we applied LSMDS on a total of 5000 surnames. The dataset includes 2500 distinct surnames along with 2500 errored names. We used these approximated name-like vectors to explain the variance of the errors in the Euclidean space.

We examined the variance of erroneous vectors relative to their correct version. Fig. 7 illustrates a comparison between the distribution of a whole sample of name-like vectors (all the surnames and their errors) with a distribution of errors that belong to a single name-like vector (errors of a single surname). We have only shown results from one coordinate since the other coordinates exhibited similar results. The top histograms represent the distribution of a single variable ($X6$: the 6^{th} coordinate) in name-like vectors and their errors. In contrast, the bottom histogram represents the distribution of the same variable considering a single name-like vector and its errors. The small spread of the histogram that contains only the error distribution implies that we can add appropriate noise by adding a small relative variance to a name-like vector to simulate its errors in the Euclidean space.

Finally, we investigated the variance-covariance matrices corresponding to each of the erroneous surnames and their errors. These values were small compared to the variance-covariance matrix of the whole sample, indicating that these small edit distance errors are closer to their real surname values than other names. By applying the relative eigenvalue analysis to the variance-covariance matrices, we found the relevant eigenvalues and eigenvectors. The first eigenvalue γ_1 refers to the maximal covariance ratio between the errors and the name-like vectors, obtained a value closer to 0.1. Therefore, even in the worst-case scenario, the errors have a small variance compared to the name-like vectors in the Euclidean space.

Hence, our model can simulate simple edit distance errors by adding Gaussian noise to an existing Euclidean space. The noise is generated from a normal (Gaussian)

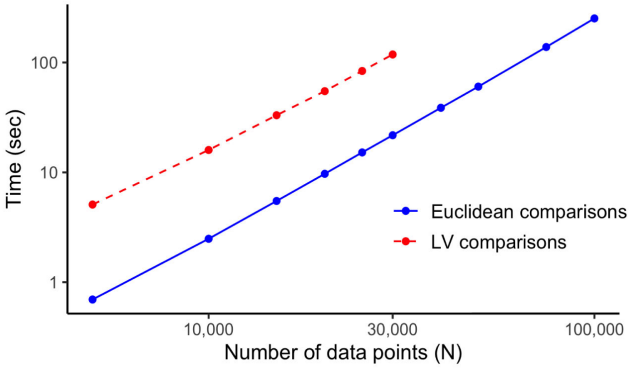

FIGURE 8. The computational time to calculate pairwise distances between name strings and name-like vectors. Calculating the Euclidean distances is much faster than calculating LV distances.

distribution $N(0, \Sigma_e)$. We estimated the unbiased estimator of the covariance matrix (Σ_e) using the pooled covariance estimate $\hat{\Sigma}_p$ [41]. Therefore, we can simulate errors of simulated name-like vectors such that they preserve small distances to their initial name-like vectors.

H. COMPUTATIONAL COMPLEXITY

In this section, we discuss the practical use of our simulator when developing big data ER algorithms. These algorithms require large-scale test data and efficient pairwise calculations for testing. We mainly focused on the computational time of generating large-scale name-like vectors and pairwise comparisons between them to measure the performance.

Pairwise comparisons between records are challenging when the majority of the entity records contain strings. In contrast, name-like vector comparisons of a complete dataset would only take much lower computational time. Fig. 8 compares the computational time of the pairwise comparisons between a dataset containing real surnames and a simulated set of name-like vectors. The surname comparisons are based on the LV distances, whereas the name-like vector comparisons are measured using Euclidean distances. The time grows quadratically for both methods, but the results of our approach are nearly ten times faster. This efficiency is beneficial for the rapid testing of algorithms that apply to large-scale data.

We also measured the computational time to simulate datasets of different sizes. The results showed that the time grew linearly, and it took less than a minute to generate 1 billion name-like vectors.

I. COMPARING DIFFERENT DATASETS

We used three datasets for the analysis (see Section V.A for details). We tested all three of them as the input to our model in the data analysis, and the results were substantially the same.

We also used these data to understand the universality of the results. Fig. 9 compares the distributions of pairwise LV distances of the random samples of 5000 surnames for each dataset and the distribution of pairwise Euclidean distances

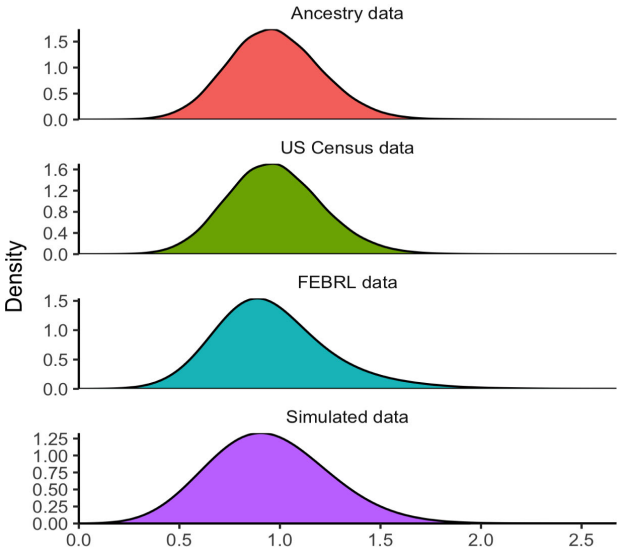

FIGURE 9. The distributions of pairwise LV distances of three datasets that contain unique surnames and the Euclidean distances of a simulated namespace. The plots are quite similar in shape, sharing similar statistical properties.

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

VI. DISCUSSION AND FUTURE WORK

We can synthesise large-scale datasets of name-like vectors using our simulation model, but the model has limitations. To obtain the full benefit of the proposed simulator, we still need to model other variables required for linking, such as addresses and gender. As a result, the dimensions of the vectors need to be extended accordingly to facilitate complete records. However, numerical or categorical data such as age or gender are easier to simulate.

Surnames (or names) follow a Zipf distribution [42], where few high-frequency names account for most of the population with many low-frequency names. We can easily modify our model to capture the frequencies of names in simulating name-like vectors by repeated sampling. Additionally, the nature of the data allows an arbitrarily large number of synthetic data points. For very large datasets, one can also increase the dimension of the space to reduce the density in which it is populated. The volume of such a space increases exponentially with the dimension.

Traditional simulation tools are often biased. For instance, existing entity identification keys such as names are typically biased towards English, Welsh, Scottish and Irish for many

datasets. Resampling from these datasets to simulate records repeats those biases in the names. We attempt to avoid this bias using a vector representation in our simulation model.

The results discussed in this paper based on the Ancestry.com [43] dataset also include biased names towards Americans and Europeans. We realized that this could distort the results and include some bias in the simulation model. Hence, we constructed the model using multiple datasets and compared the results for consistency. However, these datasets also have some biased names, which is unavoidable in many publicly available real datasets of names.

We wanted to construct and test our model based on a diverse set of names that are not biased towards English names with Roman characters, e.g., Chinese, Indian, or Arabic. However, there are extensive difficulties in Asian or Arabic names. For instance, the character set differs for these datasets; hence many string dissimilarity measures become less meaningful or have no meaning. There is an inherent problem of incorporating datasets that are not based on Roman characters in ER. In future work, we expect to investigate this problem to extend the model construction based on culturally diverse data and possibly artificially created unbiased datasets.

We are interested in extending this work to solve the global matching problem for large datasets by mainly addressing the limitations. The idea is to avoid comparisons of unnecessary detail in the pairwise comparisons (global matching) and make the ER process more scalable for big data. Hence, simulation of the many details of identification keys is not required for test data when considering the global matching problem.

VII. CONCLUSION

Our goal here was to generate synthetic data to develop and test ER algorithms appropriate for big data. We proposed a simple, inexpensive, and fast simulation model that can generate name-like vectors, including simple errors. In this paper, we discussed how to simulate simple vectors in a space that approximates the properties of names as one step towards being able to generate large simulated datasets for large-scale testing of global matching techniques.

The proposed simulation model is developed based on a simple normal distribution. This model outputs a set of abstract vectors that approximates a real namespace. Hence, generating test data for name matching has become easy. The L^p norm distance can quickly compute the similarity between numerical values, unlike the similarity between string values. Therefore, pairwise comparisons between name-like vectors are efficient when testing large-scale algorithms.

REFERENCES

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

[2] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis, "An overview of end-to-end entity resolution for big data," *ACM Comput. Surv.*, vol. 53, no. 6, pp. 1–42, Feb. 2021, doi: 10.1145/3418896.

[3] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James, "Automatic linkage of vital records," *Science*, vol. 130, pp. 954–959, Oct. 1959.

[4] J.-M. Herrera, A. Hogan, and T. Käfer, "BTC-2019: The 2019 billion triple challenge dataset," in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, 2019, pp. 163–180.

[5] M. Arehart and K. J. Miller, "A ground truth dataset for matching culturally diverse romanized person names," in *Proc. LREC*, 2008, pp. 1–4.

[6] P. Christen and D. Vatsalan, "Flexible and extensible generation and corruption of personal data," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2013, p. 1165.

[7] P. Zezula, G. Amato, V. Dohnal, and M. Batko, *Foundations of Metric Space Searching*. Boston, MA, USA: Springer, 2006, pp. 5–66.

[8] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," *J. Amer. Stat. Assoc.*, vol. 64, no. 328, pp. 1183–1210, 1969.

[9] D. Vatsalan, P. Christen, and V. S. Verykios, "A taxonomy of privacy-preserving record linkage techniques," *Inf. Syst.*, vol. 38, no. 6, pp. 946–969, 2013.

[10] N. Barlaug and J. A. Gulla, "Neural networks for entity matching: A survey," *ACM Trans. Knowl. Discovery From Data*, vol. 15, no. 3, pp. 1–37, Apr. 2021.

[11] *NSW Department of Health. New South Wales Mothers and Babies 2002*, C Epidemiol. Res., NSW Public Health Bull, NSW, Australia, 2002, p. s-3, vol. 14.

[12] P. Christen, "Preparation of a real temporal voter data set for record linkage and duplicate detection research," ANU, Tech. Rep., 2014. [Online]. Available: http://users.cecs.anu.edu.au/~Peter.Christen/publications/ncvoter-report-29june2014.pdf

[13] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," *ACM SIGMOD Rec.*, vol. 24, no. 2, pp. 127–138, May 1995.

[14] P. Bertolazzi, L. Santis, and M. Scannapieco, "Automatic record matching in cooperative information systems," in *Proc. ICDT Int. Workshop Data Qual. Cooperat. Inf. Syst. (DQCIS)*, 2003, p. 9.

[15] J. R. Talburt, Y. Zhou, and S. Y. Shivaiah, "SOG: A synthetic occupancy generator to support entity resolution instruction and research," in *Proc. ICIQ*, vol. 9, 2009, pp. 91–105.

[16] M. Tromp, A. C. Ravelli, G. J. Bonsel, A. Hasman, and J. B. Reitsma, "Results from simulated data sets: Probabilistic record linkage outperforms deterministic record linkage," *J. Clin. Epidemiol.*, vol. 64, no. 5, pp. 565–572, May 2011.

[17] T. Bachteler and J. Reiher, "TDGen: A test data generator for evaluating record linkage methods," Univ. Duisburg-Essen, Duisburg, Germany, Tech. Rep. WP-GRLC-2012-01, 2012. [Online]. Available: https://ssrn.com/abstract=3549240

[18] P. Christen, "Febrl: An open source data cleaning, deduplication and record linkage system with a graphical user interface," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 1065–1068.

[19] S. K. Sienčnik, "Adapting word2vec to named entity recognition," in *Proc. 20th Nordic Conf. Comput. Linguistics (NODALIDA)*. Linköping, Sweden: Linköping Univ. Electronic Press, May 2015, pp. 239–243.

[20] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, pp. 1–12, Sep. 2013.

[21] A. Mazeika and M. H. Böhlen, "Cleansing databases of misspelled proper nouns," in *Proc. CleanDB Workshop*, 2006, pp. 63–70.

[22] C. Li, L. Jin, and S. Mehrotra, "Supporting efficient record linkage for large data sets using mapping techniques," *World Wide Web*, vol. 9, no. 4, pp. 557–584, Dec. 2006.

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

[24] C. Faloutsos and K.-I. Lin, "FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*. New York, NY, USA: Association for Computing Machinery, 1995, p. 163–174.

[25] J. T.-L. Wang, X. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro, and K. Zhang, "Evaluating a class of distance-mapping algorithms for data mining and clustering," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. New York, NY, USA: Association for Computing Machinery, 1999, p. 307.

[26] M. C. Hout, M. H. Papesh, and S. D. Goldinger, "Multidimensional scaling," *WIREs Cognit. Sci.*, vol. 4, no. 1, pp. 93–103, 2013.

[27] T. Cox and M. Cox, *Multidimensional Scaling* (Chapman & Hall/CRC Monographs on Statistics & Applied Probability). New York, NY, USA: Taylor & Francis, 1994.

[28] N. Saeed, H. Nam, M. I. U. Haq, and D. B. M. Saqib, "A survey on multidimensional scaling," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 1–25, Jul. 2018, doi: 10.1145/3178155.
[29] M. P. J. Van der Loo, "The STRINGDIST package for approximate string matching," *R J.*, vol. 6, no. 1, pp. 111–122, 2014.
[30] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, Mar. 1964.
[31] A. Ghodsi, "Dimensionality reduction a short tutorial," Dept. Statist. Actuarial Sci., Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep., Jan. 2006, p. 38.
[32] P. J. Groenen and I. Borg, "The past, present, and future of multidimensional scaling," Erasmus School Econ., Econometric Inst., Erasmus Univ. Rotterdam, Rotterdam, The Netherlands, Econ. Inst. Res. Papers EI 2013-07, 2013, pp. 1–25. [Online]. Available: https://ideas.repec.org/p/ems/eureir/39177.html
[33] P. J. F. Groenen and M. van de Velden, "Multidimensional scaling by majorization: A review," *J. Stat. Softw.*, vol. 73, no. 8, pp. 1–26, Sep. 2016.
[34] S.-H. Bae, J. Y. Choi, J. Qiu, and G. C. Fox, "Dimension reduction and visualization of large high-dimensional data via interpolation," in *Proc. 19th ACM Int. Symp. High Perform. Distrib. Comput. (HPDC)*, New York, NY, USA, 2010, pp. 203–214.
[35] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html
[36] C. J. Mecklin and D. Mundfrom, "On using asymptotic critical values in testing for multivariate normality," InterStat, Dept. Math. Statist., Murray State Univ., Murray, Kentucky, Tech. Rep., 2003.
[37] S. Korkmaz, D. Goksuluk, and G. Zararsiz, "MVN: An R package for assessing multivariate normality," *R J.*, vol. 6, no. 2, pp. 151–162, 2014.
[38] T. Burdenski, "Evaluating univariate, bivariate, and multivariate normality using graphical and statistical procedures," *Multiple Linear Regression Viewpoints*, vol. 26, pp. 15–28, Jan. 2000.
[39] C. Chatfield and A. Collins, *Introduction to Multivariate Analysis*. Boca Raton, FL, USA: CRC Press, 1981.
[40] P. Christen, "Probabilistic data generation for deduplication and data linkage," in *Intelligent Data Engineering and Automated Learning—IDEAL 2005*. Berlin, Germany: Springer, 2005, pp. 109–116.
[41] F. Bookstein, "Comparing covariance matrices by relative eigenanalysis, with applications to organismal biology," *Evol. Biol.*, vol. 41, no. 2, pp. 336–350, 2014.
[42] M. Cristelli, M. Batty, and L. Pietronero, "There is more than a power law in Zipf," *Sci. Rep.*, vol. 2, p. 812, Nov. 2012.
[43] J. Sukharev, L. Zhukov, and A. Popescul, "Parallel corpus approach for name matching in record linkage," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 995–1000.
[44] J. Comenetz, "Frequently occurring surnames in the 2010 census," Bureau Census, Suitland-Silver Hill, MD, USA, Tech. Rep., 2016. [Online]. Available: https://www.census.gov/topics/population/genealogy/data/2010_surnames.html, doi: 10.13140/RG.2.2.30041.83043.
[45] J. Melville. (2019). *Mize: Unconstrained Numerical Optimization Algorithms R Package Version 0.2*. [Online]. Available: https://cran.r-project.org/web/packages/mize/mize.pdf
[46] J. de Leeuw and P. Mair, "Multidimensional scaling using majorization: SMACOF in R," *J. Stat. Softw.*, vol. 31, no. 3, pp. 1–30, Aug. 2009.
[47] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string metrics for matching names and records," in *Proc. KDD Workshop Data Cleaning Object Consolidation*, vol. 3, 2003, pp. 73–78.
[48] P. Christen, "A comparison of personal name matching: Techniques and practical issues," in *Proc. 6th IEEE Int. Conf. Data Mining-Workshops (ICDMW)*, 2006, pp. 290–294.
[49] W. Hoeffding, *A Non-Parametric Test of Independence*. New York, NY, USA: Springer, 1994, pp. 214–226.

SAMUDRA HERATH received the B.S. degree (Hons.) in computer science from the University of Colombo School of Computing, Sri Lanka, in 2015. She is currently pursuing the Ph.D. degree in mathematical and computer sciences with The University of Adelaide, Australia. Her research interests include the large-scale entity resolution, data integration, data mining, information retrieval, and statistical modeling.

MATTHEW ROUGHAN (Fellow, IEEE) received the Ph.D. degree in applied mathematics from The University of Adelaide, in 1994. He is currently a Professor with the School of Mathematical Sciences, The University of Adelaide, and the Chief Investigator with the Australian Research Council (ARC) Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS). His research interests include measurement/estimation, modeling and control of computer networks and social networks, graph theory, stochastic modeling, statistics, abstract algebra, formal methods, and signal processing. He is a fellow of ACM.

GARY GLONEK received the Ph.D. degree in statistics from Flinders University, SA, Australia. He is currently an Associate Professor and the Former Head of the School of Mathematical Sciences, The University of Adelaide. His research interests include statistics, especially with applications in bioinformatics. He is also interested in applied statistics and has undertaken consultancies across a wide range of areas, including road safety, wine quality, and healthcare policy.